Document Information

Analyzed document	Machine Learning.pdf (D165969181)
Submitted	5/5/2023 10:23:00 AM
Submitted by	Mumtaz B
Submitter email	mumtaz@code.dbuniversity.ac.in
Similarity	17%
Analysis address	mumtaz.dbuni@analysis.urkund.com

Sources included in the report

W URL: https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf
Fetched: 8/29/2021 3:25:58 PM 189

Entire Document

Machine Learning

98% **MATCHING BLOCK 1/189** W

100% **MATCHING BLOCK 2/189** W

INTRODUCTION Artificial Intelligence and Machine Learning have been governing the information technology sector over the past two decades. With the ever increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress. Today technology has made its roots in the day to day life of humans. The computers are overpowering humans in every way possible. Being an application of Artificial Intelligence (AI), Machine learning aims at computers to learn like humans. With the help of machine learning, computers can learn automatically without human interference or assistance. To simplify learning, the study material covers seventeen units partitioned in four modules. For effortless learning, the reader should have some basic knowledge of probability basic and Engineering mathematics. The overall objective of the course is to gain knowledge about basic concepts of Machine Learning, identify machine learning techniques suitable for a given problem, solve the problems using various machine learning techniques, design and implement machine learning solutions to classification, regression, and clustering problems; and be able to evaluate and interpret the results of the algorithms. Module-1 reveals the basic concept of Machine Learning. The evolution of learning methods, basic components of learning systems, and the applications of Machine Learning in distinct fields are illustrated in the first Unit. Unit-2 elaborates on the various types of Machine Learning methods. Unit-3 and 4 illustrate the designing of a learning system and various Machine learning frameworks. Module-2 gives an extended outlook on Supervised Learning. Unit- 5, 6, 7, 8, and 9 depict the Decision Trees, Classification and Regression Trees, Bayesian and Nearest- Neighbor Classifiers, Neural Networks, and Support Vector Machines respectively. All these topics are discussed in detail. Module-3 elaborates the basic concept of Unsupervised Learning, Clustering, Clustering Methods, Clustering Validation, and Dimensionality Reduction. Module-4 focuses on the third type of Machine Learning, i.e. Reinforcement Learning, Genetic Algorithms, Analytical and Ensemble Learning, and Design and Analysis of Machine Learning Experiments.
This content is designed comprehensively and follows a simple approach, keeping in mind the syllabus of the program. It exhilarates interest and is sure to stimulate knowledge among the readers. The purpose is to acquaint the readers with the principle and design of computer organization and architecture. Numerous figures and tables, key terms help in simplifying learning about the subject. The 'Check Your Progress' section intends the readers to test their knowledge. It is hoped that the language and the content demonstration is coherent to the readers and will enhance their learning in the best way possible.
Happy Learning!
Introduction, Basics and Types of Machine Learning MODULE: I INTRODUCTION, BASICS, AND TYPES OF MACHINE LEARNING Introduction, Basics and Types of Machine Learning

Introduction, Basics and Types of Machine Learning Unit 1 – Introduction Structure 1.0 Introduction 1.1 Unit Objectives 1.2 Definition of Machine Learning 1.3 Basic Components of learning process 1.4 Phases of Machine Learning 1.5 Importance of Machine Learning 1.6 Applications of Machine Learning 1.6.1 Examples of Machine Learning Applications 1.7 Issues in Machine Learning 1.8 Summary 1.9 Key Terms 1.10 Check Your Progress 1.0 Introduction In the past 20 years, machine learning has become an important component of the growing field of data science. With the increase in the amount of available data, there are good reasons to believe that the analysis of data will become more common and become an essential part of technological progress. Being an application of Artificial Intelligence (AI), Machine learning aims at computers to learn like humans. With the help of machine learning, computers can learn automatically without human interference or assistance. In simple terms, learning is defined as the process of converting experiences into expertise or knowledge. Humans can learn with their past experiences. Similarly, a machine can also learn with the help of past data and algorithms. A learning algorithm considers the training data (experiences) as input and delivers expertise as output. This expertise usually takes the form of another computer program performing some task. Let's begin with an example. Assume that John likes to listen to music a lot. He makes his choice according to the intensity and tempo of the music. On analyzing the kind of songs he likes, it can be concluded that he prefers songs with both high intensity and tempo. Now, if John listens to a new song, then it can be predicted with the help of his
Introduction, Basics and Types of Machine Learning previous song choice that he will like the song or not. This implies that there is a certain algorithm on which John's song choice works. By following this algorithm anyone can predict whether John will like the new song or not. This example can simply explain the basis of machine learning. To learn how a machine can learn to behave and think like humans, we should first discuss an overview, definition, importance, and framework of machine learning. 1.1 Unit Objectives After completing this unit, the reader will be able to: ● Gain knowledge about the basic overview
of machine learning. ● Illustrate the basic components of the learning process. ● Describe
the phases of machine learning lifecycle. ● Learn about various applications of machine learning. ● Discuss the importance and issues in machine learning. 1.2 Definition of Machine Learning The term Machine Learning was coined by Arthur Samuel, a pioneer in the field of artificial intelligence, in 1959 at IBM. According to him, "

91% **MATCHING BLOCK 11/189** **W**

Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed." 1 1.

100% **MATCHING BLOCK 3/189** **W**

Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data or both.

Different authors have different perspectives on machine learning. However, there is no universal definition for machine learning. Some other definitions of machine learning are given below: 2 1 Arthur L Samuel. "Some studies in machine learning using the game of checkers". In: IBM Journal of research and development 3.3 (1959), pp. 210–229. 2
Ethem Alpaydin, Introduction to Machine Learning, The MIT Press, Cambridge, Massachusetts, 2004.
The field of study known as machine learning is concerned with the question of how to
Introduction, Basics and Types of Machine Learning 2. construct computer programs that automatically improve with experience. 3 3. Machine learning is the hot new thing. — John L. Hennessy, President of Stanford (2000–2016) 4. A breakthrough in machine learning would be worth ten Microsofts. — Bill Gates, Microsoft Co-Founder 5.

100% **MATCHING BLOCK 4/189** **W**

A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks T, as measured by P, improves with experience E. 3

Table 1.1 provides some situations as examples and depict the corresponding performance, task, and experience. Table 1.1 Situational examples for tasks, performance, and experience Situation Details

100% **MATCHING BLOCK 5/189** **W**

Handwriting recognition learning problem ● Task (T): Recognising and classifying handwritten words within images ● Performance

measure (

98%	MATCHING BLOCK 6/189	W

P): Percent of words correctly classified ● Training experience (E): A dataset of handwritten words with given classifications A robot driving learning problem ● Task (T): Driving on highways using vision sensors ● Performance measure (P): Average distance traveled before an error ● Training experience (

E):

93%	MATCHING BLOCK 7/189	W

A sequence of images and steering commands recorded while observing a human driver A chess learning problem ● Task (T): Playing chess ● Performance measure (P): Percent of games won against opponents ● Training experience (E): Playing practice games against itself A computer program that learns from experience is called a machine learning program or simply a learning program. Sometimes such a program is also referred to as a learner.

Figure 1.1 represents the difference between traditional programming and machine learning. 3 Mitchell T., Machine Learning, McGraw Hill, 1997.
Introduction, Basics and Types of Machine Learning Figure 1.1 Machine learning vs Traditional programming 1.3

44%	MATCHING BLOCK 8/189	W

Basic Components of Learning Process Either for humans or a machine, the learning process includes four components, data storage, abstraction, generalization, and evaluation. Figure 1.2 represents the

basic components of the learning process.
Figure 1.2 Components of

56%	MATCHING BLOCK 9/189	W

Learning Process 1. Data storage: Storing and retrieving large amounts of data are important components of the learning process. Both humans and computers utilize data storage as a foundation for advanced reasoning. Humans store the data in the brain and retrieve it using electrochemical signals.

On the other hand,

100%	MATCHING BLOCK 10/189	W

computers use hard disk drives, flash memory, random access memory, and similar devices to store data and use cables and other technology to retrieve data. 2. Abstraction:

Abstraction being the second component of the learning process, is
the process
Introduction, Basics and Types of Machine Learning

77%	MATCHING BLOCK 12/189	W

of extracting knowledge about already stored data. This involves creating general concepts about the data with the help of known models and new models. Training is the process of fitting a model to a dataset. Once the model is trained, the data is transformed into an abstract form that summarizes the original information. 3. Generalization: The third component generalization of the learning process describes the process of turning the knowledge about stored data into a form that can be utilized for future action. These actions are to be carried out on tasks that are similar, but not identical, to those

that

have been seen before. In generalization, the goal is to discover those properties of the data that will be most relevant to future tasks. 4. Evaluation: Evaluation, being the last component of the learning process, is the process of giving feedback to the user to measure the utility of the learned knowledge. This feedback is then utilized to effect improvements in the whole learning process. 1.4 Phases of Machine Learning

We are already aware of the definition of machine learning. In simple terms, machine learning provides an automatic learning platform for computer systems without being explicitly programmed. But inquisitively, it becomes interesting to know how a machine learning system works. This can be analyzed by defining the seven phases of machine learning. The cyclic process of machine learning is capable of finding an appropriate solution to problems or projects. Before proceeding with a machine learning process, It is important to understand the problem completely. This helps in analyzing the process better and provides good results. The machine learning system is based on a model and this model is created by providing training. The seven phases of a machine learning system include: ➔ Data Collection ➔ Data Preparation ➔ Data Wrangling ➔ Data Analyzation ➔ Training the model
Introduction, Basics and Types of Machine Learning ➔ Testing the model ➔ Deployment Data Collection Collecting or gathering the data is the initial phase of the system. Data collection aims at identifying and obtaining all data-related problems. The data sources for collecting the data include files, databases, the internet, or mobile devices. The efficiency of the data is determined by the quality and quantity of the data collected. More is the data, more accurate will be the prediction. The data obtained after collection is termed as a dataset. Data Preparation After gathering the data, the next step is to prepare the data according to the requirement. The data is put into a suitable place and prepared to be used in machine learning training. Initially, all the data is put together and the order of the data is randomized. The further process is divided into: ● Data Exploration: In this step, the nature of the data is analyzed. This includes understanding the characteristics, format, and quality of the data. The data is analyzed using correlations, general trends, and outliers. ● Data Pre-processing: In this step, the data is pre-processed for its analysis. Data Wrangling Data wrangling is one of the most important phases of the machine learning lifecycle. It is the process of cleaning and converting raw data into a usable format. This again contributes to making the data suitable for analysis. Cleaning of data is needed for addressing the quality issues. The issues concerned with data collection are missing values, invalid data, duplicate data, or noise. It is necessary to detect and remove these issues, ultimately cleaning the data using various filtering techniques, to improve the quality of the outcome. Data Analyzation The data analysis phase aims at building a machine learning model to analyze the data using distinct analytical techniques and review the output. The steps involved in
Introduction, Basics and Types of Machine Learning this phase are: ● Selection of analytical techniques ● Building models ● Review the outcome Firstly, the type of issue is determined and then the suitable machine learning technique is selected. These techniques can be classification, regression, cluster analysis, association, etc. Secondly, the model is built using the prepared data. Lastly, the model is evaluated. Training the Model In this phase, the model is trained to improve its performance for a better outcome of the problem. Datasets are used for this purpose using various machine learning algorithms. It is essential to train the model so that it can understand various patterns, rules, and features. Testing the Model Once the machine learning model has been trained on a given dataset, then testing of the model is performed. In this phase, the accuracy of the model is checked by providing a test dataset to it. Testing the model determines the percentage accuracy of the model as per the requirement of the project or problem. Deployment This is the last phase of the machine learning lifecycle, where the model is deployed in the real-world environment. If the model produces accurate results as per the requirements with desired speed, then the model is deployed in the real system. Before deploying the project, it is also checked whether the performance of the project is improving or not with the available data. 1.5 Importance of Machine Learning At an early age, only companies stocked their data. The data was stored and processed at the computer centers. With the advancement in computer technology and wireless communication, we all became producers of data. Every time visiting any webpage, buying a product online, posting on social media, even simply surfing the internet

Introduction, Basics and Types of Machine Learning generates data. In this age of Big Data, each of us is not only a generator but also a consumer of data. We want to have products and services specialized for us. We want our needs to be understood and interests to be predicted. The stored data is counted as experience and it helps in making further decisions. For example, in an online shopping platform, thousands of products are sold to hundreds of customers every day. The details of each transaction are stored, like customer Id, product details, date, amount, offer applied, etc. This results in a lot of data every day. Storing this information helps in analyzing and predicting the choice of customers, to maximize sales and profit. The customers can also get a review of the particular products. However, this prediction for the customer choice is not evident but there exists a certain pattern in the data. Such patterns may help us understand the process or we can use those patterns to make predictions: Assuming that the future, at least the near future, will not be much different from the past when the sample data was collected, the future predictions can also be expected to be right. Generally, computers use algorithms to solve any issue. An algorithm is defined as a set of instructions that should be carried out to transform the input to output. For example, a sorting algorithm puts a set of numbers as input and the output is their ordered list. However, for some tasks, there is no certain algorithm. For example, predicting customer behavior. The constantly evolving field of machine learning has led to a rise in demand and importance of the field too. Scientists' preference for machine learning is based on the fact that high-value predictions offer better decisions and smart actions in real-time without any human interference. Machine learning helps in analyzing large amounts of data and also simplifies the task data. The automation process has acquired much prominence and recognition by researchers. Machine learning involves automatic sets of generic methods. These methods have replaced the traditional statistical methods of data extraction and interpretation. Machine learning is a core part of artificial intelligence. It also helps in finding solutions to the problems in vision, speech recognition, and robotics. The past experiences or example data help in optimizing the computer programs. Models define

52%	MATCHING BLOCK 14/189	W

some parameters and learning depicts the execution of computer programs to optimize these parameters of the models using past experiences or training data.

These models can be either termed as predictive for making predictions in the future
Introduction, Basics and Types of Machine Learning or descriptive for gaining knowledge from data, or both. In machine learning, the theory of statistics is utilized to build mathematical models. 1.6

71%	MATCHING BLOCK 15/189	W

Applications of Machine Learning Applying machine learning to large databases is known as data mining. In data mining, a large amount of data is processed to build a simple model

of use value such as

61%	MATCHING BLOCK 16/189	W

high prediction accuracy. Following is a list of some of the applications of machine learning. ● Machine learning is used in the retail business to study consumer

behavior. ● Image recognition is one of the important applications of machine learning. It is used to identify objects, persons, places, digital images, etc. Automatic friend tagging suggestion in Facebook is one of the popular examples of machine learning in image recognition. ● Machine learning is used in speech recognition. For example, Alexa, Google search by voice, etc. ● In the field of

77%	MATCHING BLOCK 17/189	W

finance, banks analyze their past data to build models for use in credit applications, fraud detection, and stock marketing. ● In manufacturing industries, training models are used for optimization, control, and troubleshooting. ● Learning programs are

even

100%	MATCHING BLOCK 18/189	W

used for medical diagnosis. ● In telecommunications, call patterns are analyzed

to optimize the network and increase the quality of service. ●

In artificial intelligence, it is used to train a system to learn and adapt to changes so that the system designer does not

need to predict and come up with solutions for all situations that may occur. ● In the field of science, large amounts of physical, astronomical, and biological data can only be analyzed quickly enough by computers. Due to the constant evolution of the World Wide Web, finding relevant information manually becomes an impossible task. ● Machine learning is

applied in the design of computer-controlled vehicles to drive accurately when driving on various types of roads.

For example, Google Maps.
Introduction, Basics and Types of Machine Learning ● Machine learning methods have been used to develop programs that play games with the users. ● Machine learning is widely used by various e-commerce and entertainment companies such as Amazon Prime, Netflix, etc., for product recommendations to the user. ● Machine learning is applied for filtering spam emails and malware. 1.6.1 Examples of Machine Learning Applications Let's discuss some of the examples for machine learning applications to have a broader idea about the basics of this field. Learning Associations Consider an example of a sports shop. The distributor is interested in knowing if there is any certain pattern
in the purchase of the products. Like, if a customer buys a bat and ball together, then he is likely to also buy
batting gloves. After analyzing this customer behavior, it can be interpreted that there must be an association between the set of products {bat, ball} and the set {batting gloves}. Association rule in machine learning is a method for analyzing the relations between the variables in large databases, using the concept of mutual interest. The association rule for the above example is represented as: {bat, ball} ⇒ {batting gloves}
The measure of how likely a customer, who has bought
a bat and a ball,
to buy batting gloves also is given by the conditional probability P ({bat, ball}|{batting gloves}) Let this conditional probability is 0.8, then the rule may be stated more precisely as follows: "80% of customers who buy
a bat and a ball also buy batting gloves."
Consider an association rule of the form X ⇒ Y, that is, if people buy X then they are also likely to buy Y.
Introduction, Basics and Types of Machine Learning
Suppose there is a customer who buys X and does not buy Y . Then that customer is a potential Y customer. Once we find such customers, we can target them for cross- selling. A knowledge of such rules can be used for promotional pricing or product placements. In finding an association rule X ⇒ Y, we are interested in learning a conditional probability of
the form P (Y |X) where Y is the product the customer may buy and X is the product or the set of products the customer has already purchased.
If we may want to make a distinction among customers, we may
estimate P (Y |X, D) where D is a set of customer attributes, like gender, age, marital status, and so on, assuming that we have access to this information.
Classification
Classification, in machine learning, is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.
Considering an example, there is a training set of data as shown in table 1.2.
There are two attributes "Score1" and "Score2". The class label is called "Result". The class label has two possible values: "Pass" and "Fail". The data can be divided into two categories or classes: The set of data for which the class label is "Pass" and the set of data for which the class label is "Fail". Let us assume that we have no knowledge about the data other than what is given in the table.
Now, the problem can be posed as follows: If we have some new data, say "Score1 = 33" and "Score2 = 28", what value should be assigned to "Result" corresponding to the new data.
Table 1.2 Example data for a classification problem Score1 22 29 10 31 32 Score2 43 47 49 54 40 Result Pass Pass Fail Pass Pass
Introduction, Basics and Types of Machine Learning
To answer this question, using the given data alone we need to find the rule or the formula, or the method that has been used in assigning the values to the class label "Result". The problem of finding this rule or formula or the method is the classification problem. In general, even the general form of the rule or function, or method will not be known. So several different rules, etc. may have to be tested to obtain the correct rule or function, or method.
Another real-life
example of the classification problem can be of a credit card company. The
company receives hundreds of thousands of applications for new cards. The applications contain information regarding several attributes like annual salary, age, etc. The problem is to devise a rule to classify the applicants to those who are credit-worthy, who are not credit-worthy, or to those who require further analysis.

Some general features of the classification problems are: ●
A classification problem requires that examples be classified into one of two or more classes. ● A classification can have real-valued or discrete input variables. ●
A
discriminant of a classification problem is a rule or a function that is used to assign labels to new observations. ●
A problem with two classes is often called a two-class or binary classification problem. ● A problem with more than two classes is often called a multi-class classification problem. ● A problem where an example is assigned multiple classes is called a multi- label classification problem. There are several machine learning
algorithms for classification. The following are some of the well-known algorithms. ● Logistic regression ● Naive Bayes algorithm ● k-NN algorithm ● Decision tree algorithm ● Support vector machine algorithm ● Random forest algorithm
Introduction,
Basics and Types of Machine Learning Regression
In machine learning, a regression problem is a problem of predicting the value of a numeric variable based on the
observed
values of the variable. The value of the output variable may be a number, such as an integer or a floating-point value. These are often quantities, such as amounts and sizes. The input variables may be discrete or real-valued.
Let x denote the set of input variables and y the output variable. In machine learning, the general approach to regression is to assume a model, that is, some mathematical relation between x and y, involving some parameters say, θ, in the following form: y = f (x, θ) The function f(x, θ) is called the regression function. The machine learning algorithm optimizes the parameters in the set θ such that the approximation error is minimized; that is, the estimates of the values of the dependent variable, y are as close as possible to the correct values given in the training set.
For
example,
consider Table 1.3 showing the prices of cars.
If the input variables are “Age”, “Distance” and “Weight” and the output variable is “Price”, the model maybe y = f(x, θ) Price = a 0 + a 1 × (Age) + a 2 × (Distance) + a 3 × (Weight) where x = (Age, Distance, Weight) denotes the set of input variables and θ = (a 0 , a 1 , a 2 , a 3) denotes the set of parameters of the model.
Table 1.3 Example data for a regression problem: Prices of used cars Price (US$) Age (Years) Distance (Kms.) Weight (Pounds) 14950 25 38500 1170 14500 30 41000 1165 13750 27 41711 1165 12950 23 71138 1245
There are various types of regression techniques available to make predictions. These techniques mostly differ in three aspects, namely, the number and type of
Introduction, Basics and Types of Machine Learning
independent variables; the type of dependent variables; and the shape of the regression line. Some of these are listed below. ● Simple linear regression: There is only one continuous independent variable x and the assumed relation between the independent variable and the dependent variable y is y = a + bx. ● Multivariate linear regression: There are more than one independent variable, say x 1 , . . ., x n , and the assumed relation between the independent variables and the dependent variable is y = a 0 + a 1 x 1 + ⋯ + a n x n . ● Polynomial regression: There is only one continuous independent variable x and the assumed model is y = a 0 + a 1 x + ⋯ + a n x n . ● Logistic regression: The dependent variable is binary, that is, a variable
that
takes only the values 0 and 1. The assumed model involves certain probability distributions. 1.7 Issues in Machine Learning
There are various common practical issues in machine learning. Some of them are listed below: 1. Lack of Quality Data: The absence of good data is one of the main issues in Machine Learning. Data quality is fundamental for the algorithms to work as proposed. Incomplete data, unclean data, and noisy data are the typical rivals of an ideal machine learning. Different reasons for low data quality are- ● Data can be noisy which will result in inaccurate predictions. This often leads to less accuracy in classification and low-quality results. It is noted as one of the most common errors faced in terms of data. ● Incorrect or incomplete information can also lead to faulty programming through Machine Learning. Having less information will lead the program to analyze based on the minimal data present. Hence, decreasing the accuracy of the results.

Introduction, Basics and Types of Machine Learning ● For better future actions, the generalization of input and output of past data is crucial. But a common issue that occurs is, the output data can become difficult to generalize. 2. Understanding which processes need Automation: It is difficult to choose facts from fiction with respect to machine learning. Before deciding on the AI platform to work upon, we need to thoroughly evaluate the types of problems to be solved. The processes that are carried out manually every day with no variable output, are the easiest processes to automate. The complicated processes need to be inspected before automation. Machine learning can help in the automation of some processes but not for all the processes. 3. Inadequate Infrastructure: Machine learning requires large amounts of data stirring abilities. The frameworks based on inheritance are not able to deal with the responsibility. One should check if the infrastructure can deal with the issues in machine learning. If it can't, then one should look for upgrading it completely with good hardware and flexible storage. 4. Implementation: Organizations generally have analytics engines before upgrading to machine learning. Integrating newer Machine Learning methodologies into existing methodologies is a complicated task. Maintaining proper interpretation and documentation goes a long way to easing implementation. Partnering with an implementation partner can make the implementation of services like anomaly detection, predictive analysis, and ensemble modeling much easier. Some issues in machine learning regarding implementation are- ● Slow deployment – Although the models of Machine Learning are time efficient but their creating process is quite the opposite. As it is still a young innovation the implementation time is slow. ● Data Security – Saving confidential data on ML servers is a risk as the model will not be able to differentiate between sensitive and insensitive data. ● Lack of data is another key issue faced during the implementation of the model. Without adequate data, it is not possible to give out valuable intel. Introduction, Basics and Types of Machine Learning 5. Lack of Skilled Resources: Machine learning and Deep analytics in their current forms are still new technologies. Thus, there is a shortage of skilled employees available to manage and develop analytical content for Machine Learning. Data scientists often need a combination of domain experience as well as in-depth knowledge of science, technology, and mathematics. 1.8 Summary ● Being an application of Artificial Intelligence (AI), Machine learning aims at computers to learn like humans. With the help of machine learning, computers can learn automatically without human interference or assistance. ● Either for humans or a machine, the learning process includes four components, data storage, abstraction, generalization, and evaluation. ● The machine learning system is based on a model and this model is created by providing training. The seven phases of a machine learning system include Data Collection, Data Preparation, Data Wrangling, Data Analyzation, Training the model, Testing the model, and Deployment. ● Machine learning has a vast range of applications in the real world. It can be used in speech recognition, e-commerce and entertainment companies, filtering spam emails and malware, in the field of finance, banks, etc. ● Association rule in machine learning is a method for analyzing the relations between the variables in large databases, using the concept of mutual interest. ●
Classification, in machine learning, is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. ●
In machine learning, a regression problem is a problem of predicting the value of a numeric variable based on the observed values of the variable. 1.9
Key Terms ● Machine learning Program:

91% **MATCHING BLOCK 21/189** W

A computer program that learns from experience is called a machine learning program or simply a learning program. ●

Training: It is the process of fitting a model to a dataset. ● Dataset: The data obtained after collection is termed as a dataset. Introduction, Basics and Types of Machine Learning ● Data Mining: Applying machine learning to large databases is known as data mining. ●
Discriminant: A discriminant of a classification problem is a rule or a function that is used to assign labels to new observations. 1.10 Check Your Progress Short-Answer Type Q1)
A problem with more than two classes is often called a ________ classification problem.
Q2) Machine learning is applied for filtering spam emails and malware. (True/ False?) Q3) Logistic regression algorithm is used for ______ problem in machine learning. Q4) Applying machine learning to large databases is known as data mining. (True/ False?) Long-Answer Type Q1) What is Machine Learning? What is the importance of machine learning? Q2) List at least five issues in machine learning. Q3) Describe the phases of the machine learning lifecycle. Q4) What are the basic components of the learning process? Q5) List the applications of machine learning in detail. References: Machine Learning: a Probabilistic Perspective, Kevin Patrick Murphy, MIT Press, March 2014. Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004.

Introduction, Basics and Types of Machine Learning Unit 2 – Types of Machine Learning Structure 2.0 Introduction 2.1 Unit Objectives 2.2 Supervised Learning 2.3 Unsupervised Learning 2.4 Reinforcement learning 2.5 Semi-supervised learning 2.6 Relation to other fields 2.7 Summary 2.8 Key Terms 2.9 Check Your Progress 2.0 Introduction By now, we are aware of the basics, phases, and major issues in machine learning. Before gaining knowledge in depth, it is important to learn about the types of machine learning. We will discuss the categories of machine learning in brief, the details will be explained in the upcoming units. Machine learning is mainly divided into two categories. The first type of machine learning, supervised learning (or predictive) approach aims at mapping from inputs x to outputs y, given a labeled set of input-output pairs, Here, Ɗ is known as the training set, and N is the number of training examples. Each training input x i is a D-dimensional vector of numbers, representing, say, the height and weight of a person. These are called features, attributes, or covariates. In general, however, x i could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc. The second main type of machine learning is the unsupervised learning (or descriptive) approach. Here we are only given inputs, The goal is to find "interesting patterns" in the data. This is sometimes called knowledge discovery. This is a much less well-defined problem, since we are not told
Introduction, Basics and Types of Machine Learning what kinds of patterns to look for, and there is no obvious error metric to use (unlike supervised learning, where we can compare our prediction of y for a given x to the observed value). Apart from these two main categories, there is a third type of machine learning, known as reinforcement learning, which is less commonly used. This is useful for learning how to act or behave when given occasional reward or punishment signals. 2.1 Unit Objectives After completing this unit, the reader will be able to: ● Gain knowledge about types of machine learning. ● Illustrate supervised, unsupervised, reinforcement, and semi-supervised learning. ● Discuss the relation of machine learning to other fields. 2.2 Supervised Learning

97%	**MATCHING BLOCK 22/189**	W

Supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs. In supervised learning, each example in the training set is a pair consisting of an input object (typically a vector) and an output value.

Figure 2.1 represents the basic workflow in supervised learning.

97%	**MATCHING BLOCK 23/189**	W

A supervised learning algorithm analyzes the training data and produces a function, which can be used for mapping new examples. In the optimal case, the function will correctly determine the class labels for unseen instances. Both classification and regression problems are supervised learning problems. A wide range of supervised learning algorithms is available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems. A "supervised learning" is so-called because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers (that is, the correct outputs), the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

In simple terms, supervised learning is the type of machine learning that focuses on learning a classification or regression model, that is, learning from labeled training data (i.e., inputs that also contain the desired outputs or targets; basically, Introduction, Basics and Types of Machine Learning "examples" of what we want to predict).

92%	**MATCHING BLOCK 24/189**	W

A training set of examples with the correct responses (targets) is provided and, based on this training set, the algorithm generalizes to respond correctly to all possible inputs. This is also called learning from exemplars. Figure 2.1 Supervised Learning

Classification problems categorize all the variables that form the output. For mapping from inputs x to outputs y, where y ∈ {1, . . . , C}, with C being the number of classes. ● If C = 2, this is called binary classification (in which case we often assume y ∈ {0, 1}) ● If C &lt; 2, this is called multiclass classification. ● If the class labels are not mutually exclusive (e.g., somebody may be classified as tall and strong), it is called multi-label classification. Classification is probably the most widely used form of machine learning and has been used to solve many interesting and often difficult real-world problems. Let's discuss some of the examples here. E-mail spam filtering and Document classification The goal of document classification is to classify the given document, say, a web page or e-mail message, into one of the C classes. Consider e-mail spam filtering to be a special case of document classification, where the classes are spam (y = 1) or (y = 0). According to most of the classifiers, the input x has a fixed size. A transformation is defined as x ij = 1 if word j occurs in document i. On applying this transformation to

Introduction, Basics and Types of Machine Learning every document in our dataset, a binary document vs word co-occurrence matrix is obtained. For example, it should be noted that most spam messages have a high probability of containing words that are unwanted (like "cheap", "bye", etc.). Face Detection and Recognition Finding objects within an image is known as object detection or object localization. Face detection is a special case of object detection. One approach to solve this problem is to divide the image into small overlapping patches at different locations, scales, and orientations. Each patch is classified based on whether it contains a face-like texture or not. The system then returns the locations where the probability of face is adequately high. This is known as a sliding window detector. Such face-detection systems are used by modern digital cameras. Face detection can further help in the face recognition of any person. The identity of any person can be estimated using face detection. In the case of face recognition, the number of class labels C might be very large. 4 4 Szeliski, R. (2010), Computer Vision: Algorithms and Applications, Springer. Regression is similar to classification except the response variable in regression is continuous. Regression aims at predicting the value of a numeric variable based on the observed values of the variable. Some examples of real-world regression problems are predicting tomorrow's stock market price given current market conditions and other possible side information; Predicting the temperature at any location inside a building using weather data, time, door sensors, etc. 2.3 Unsupervised Learning

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions. Since the examples given to the learner are unlabeled, the accuracy of the structure that is output by the algorithm cannot be evaluated. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.

Introduction, Basics and Types of Machine Learning Unsupervised learning does not provide

correct responses, but instead the algorithm tries to identify similarities between the inputs so that inputs that have something in common are categorised together. The statistical approach to unsupervised learning is known as density estimation.

Figure 2.2 represents the basic workflow of unsupervised learning. There are two major differences between supervised and unsupervised learning. 1. Supervised learning is conditional density estimation, whereas unsupervised learning is unconditional density estimation. For supervised learning, the models are build of the form, p(x i | θ), while for unsupervised learning, the models are build of the form, p(y i | x i ,θ). 2. x i is a vector of features, so we need to create multivariate probability models. By contrast, in supervised learning, y i is usually just a single variable that we are trying to predict. This means that for most supervised learning problems, we can use univariate probability models (with input-dependent parameters), which significantly simplifies the problem. Figure 2.2

Unsupervised Learning Unsupervised learning is more applicable than supervised learning, since it does not

require a human expert (supervisor) to manually label the data. Labeled data is not only expensive, it also contains little information. This information is certainly not enough to reliably predict the parameters of complex models. Depending on the problem, the unsupervised learning model can organize the data in the following ways: • Clustering: The training data that is similar to each other are selected and grouped together.
Introduction, Basics and Types of Machine Learning • Anomaly Detection: Unsupervised learning can be used to flag outliers in a particular dataset. For example, banks can detect the fraud transactions on the basis of unusual purchasing behavior of a customer. • Association: Certain features of a data sample correlate with other features. By analyzing some key attributes of a data, the other attributes can be predicted using unsupervised learning. • Autoencoders: Autoencoders take input data, compress it into a code, then try to recreate the input data from that summarized code. 2.4 Reinforcement Learning Reinforcement learning is the third type of machine learning, which is useful for learning how to act or behave when given occasional reward or punishment signals. It is somewhat less commonly used.

100%	MATCHING BLOCK 28/189	W

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards. A learner (the program) is not told what actions to take as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situations and, through that, all subsequent rewards.

Figure 2.3 represents the basic workflow of reinforcement learning. Reinforcement is the process of learning from rewards while performing a series of actions. In reinforcement learning, we do not tell the learner or agent, for example, a (ro)bot, which action to take but merely assign a reward to each action and/or the overall outcome. Instead of having a "correct/false" label for each step, the learner must discover or learn a behavior that maximizes the reward for a series of actions. In that sense, it is not a supervised setting and somewhat related to unsupervised learning; however, reinforcement learning really is its own category of machine learning. Reinforcement learning will not be covered further in this class.
For

98%	MATCHING BLOCK 29/189	W

example, consider teaching a dog a new trick: we cannot tell it what to do, but we can reward/punish it if it does the right/ wrong thing. It has to find out what it did that made it get the reward/ punishment. We can use a similar method to train computers to do many tasks, such as playing chess, scheduling jobs, and controlling robot limbs. Reinforcement learning is different from supervised learning. Supervised learning is

Introduction, Basics and Types of Machine Learning learning from examples provided by a knowledgeable expert. Figure 2.3 Reinforcement Learning Typical applications of reinforcement learning involve playing games (chess, Go, Atari video games) and some form of robots, e.g., drones, warehouse robots, and more recently self- driving cars. 2.5 Semi-supervised learning Simply, semi-supervised learning can be defined as a mix of supervised and unsupervised
learning. We already know that in supervised learning tasks, some training examples contain outputs, but some do not. Then, a labeled training subset is used to label the unlabeled portion of the training set, which we then also utilize for model training. Semi-supervised learning contains the training dataset with both labeled and unlabeled data. This method is particularly useful when extracting relevant features from the data is difficult, and labeling examples is a time-intensive task for experts. Common situations for this kind of learning are medical images like CT scans or MRIs. A trained radiologist can go through and label a small subset of scans for tumors or diseases. It would be too time-intensive and costly to manually label all the scans — but the deep learning network can still benefit from the small proportion of labeled data and improve its accuracy compared to a fully unsupervised model.
Introduction, Basics and Types of Machine Learning Figure 2.4 Overview of the categories of Machine Learning (Source- Raschka and Mirjalili: Python Machine Learning, 2nd Ed.) 2.6 Relation to other fields Machine learning being an interdisciplinary field, shares common threads with the field of information theory, mathematical fields of statistics, game theory, and optimization. Naturally, machine learning is a subfield of computer science and a branch of AI (Artificial Intelligence). However, it should be noted that, in contrast with traditional AI, machine learning is not trying to build automated imitation of intelligent behavior, but rather to use the strengths and special abilities of computers to complement human intelligence, often performing tasks that fall way beyond human capabilities. For example, the ability to scan and process huge databases allows machine learning programs to detect patterns that are outside the scope of human perception. The component of experience, or training, in machine learning often refers to data that is randomly generated. The task of the learner is to process such randomly generated examples toward drawing conclusions that hold for the environment from which these examples are picked. Machine learning and Data Mining Data mining focuses on the discovery of patterns in datasets or "gaining knowledge

Introduction, Basics and Types of Machine Learning and insights" from data – often, this involves a heavy focus on computational techniques, working with databases, etc. Nowadays, the term data mining is more or less synonymous to "data science". We can then think of machine learning algorithms as tools within a data mining project. Data mining is not "just" but also emphasizes data processing, visualization, and tasks that are traditionally not categorized as "machine learning" (for example, association rule mining). Machine learning, AI, and deep learning Artificial intelligence (AI) is a subfield of computer science focussing on solving tasks that humans are good at (for example, natural language processing, image recognition, etc.). There are two subtypes of AI: Artificial general intelligence (AGI) and narrow AI. AGI refers to an intelligence that equals humans in several tasks, i.e., multi-purpose AI. In contrast, narrow AI is more narrowly focused on solving a particular task that humans are traditionally good at (e.g., playing a game, or driving a car, etc.). The field of machine learning has emerged as a subfield of AI. It was concerned with the development of algorithms so that computers can automatically learn (predictive) models from data. Consider an example in which we want to develop a program that can recognize handwritten digits from images. One approach would be to look at all of these images and come up with a set of (nested) if-this-then-that rules to determine which digit is displayed in a particular image. Another approach would be to use a machine learning algorithm, which can fit a predictive model based on a thousands of labeled image samples that we may have collected in a database. Deep learning is a subfield of machine learning, referring to a particular subset of models that are particularly good at certain tasks such as image recognition and natural language processing. Deep learning imitates the way humans gain certain types of knowledge. It is an important element of data science, which includes statistics and predictive modeling. At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction. To conclude, deep learning and machine learning definitely helps to develop AI, however, AI doesn't necessarily have to be developed using machine learning. Figure
Introduction, Basics and Types of Machine Learning 2.5 depicts the relationship between machine learning, deep learning, and artificial intelligence. Figure 2.5 Relationship between machine learning, deep learning, and artificial intelligence Machine learning and Statistics Machine learning serves a close relationship with statistics. Indeed there is a lot in common between the two disciplines, in terms of both the goals and techniques used. There are, however, a few significant differences of emphasis; if a doctor comes up with the hypothesis that there is a correlation between smoking and heart disease, it is the statistician's role to view samples of patients and check the validity of that hypothesis (this is the common statistical task of hypothesis testing). In contrast, machine learning aims to use the data gathered from samples of patients to come up with a description of the causes of heart disease. The hope is that automated techniques may be able to figure out meaningful patterns (or hypotheses) that may have been missed by the human observer. In contrast with traditional statistics, in machine learning in general, algorithmic considerations play a major role. Machine learning is about the execution of learning by computers; hence algorithmic Introduction, Basics and Types of Machine Learning issues are pivotal. We develop algorithms to perform the learning tasks and are concerned with their computational efficiency. Another difference is that while statistics is often interested in asymptotic behavior (like the convergence of sample- based statistical estimates as the sample sizes grow to infinity), the theory of machine learning focuses on finite sample bounds. Namely, given the size of available samples, machine learning theory aims to figure out the degree of accuracy that a learner can expect on the basis of such samples. 2.7 Summary ● Machine learning is mainly divided into two categories: supervised and unsupervised
learning. ●

93% **MATCHING BLOCK 30/189** **W**

Supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs. ● Supervised learning

focuses on learning a classification or regression model, that is, learning from labeled training data. ●

97% **MATCHING BLOCK 31/189** **W**

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. ● Unsupervised learning

is more applicable than supervised learning, since it does not require a human expert (supervisor) to manually label the data. ● Reinforcement learning is the third type of machine learning, which is useful for learning how to act or behave when given occasional reward or punishment signals. It is somewhat less commonly used. ● Semi-supervised learning contains the training dataset with both labeled and unlabeled data. This method is particularly useful when extracting relevant features from the data is difficult, and labeling examples is a time-intensive task for experts. ● Machine learning being an interdisciplinary field, shares common threads with the field of artificial intelligence, statistics, deep learning, data mining, etc. 2.8 Key Terms ● Training example: A row in the table representing the dataset. Synonymous to an observation, training record, training instance, training sample (in some contexts, sample refers to a collection of training examples).

Introduction, Basics and Types of Machine Learning ● Sliding Window: Sliding windows play an integral role in object classification, as they allow us to localize exactly "where" in an image an object resides. ● Deep learning: It is a subfield of machine learning, referring to a particular subset of models that are particularly good at certain tasks. ● Clustering: The training data that is similar to each other are selected and grouped together. ● Autoencoders: Autoencoders take input data, compress it into a code, then try to recreate the input data from that summarized code. 2.9 Check Your Progress Short-Answer Type Q1) Supervised learning is based on _________ data. Q2) Reinforcement learning is the most widely used category of machine learning. (True/ False?) Q3) Full form of AGI- a) Artificial general interpretation b) Artificial general intelligence c) Artificial gain interpretation d) Artificial gain intelligence Q4) The statistical approach to _________ learning is known as density estimation. Long-Answer Type Q1) Explain supervised learning with an example. Q2)
Distinguish between classification and regression. Q3) What are the differences between supervised and unsupervised learning? Q4) Write a short note on the relation between machine learning and artificial intelligence. Q5) List the two major categories of machine learning. Give examples. References: Machine Learning: a Probabilistic Perspective, Kevin Patrick Murphy, MIT Press, March 2014. Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004.
Introduction, Basics and Types of Machine Learning Unit 3 – Designing a learning system Structure 3.0 Introduction 3.1 Unit Objectives 3.2 Learning Models 3.2.1 Logical Models 3.2.2 Geometric Models 3.2.3 Probabilistic Models 3.3 Steps for Designing a Learning System 3.3.1

98% **MATCHING BLOCK 32/189** W

Type of training experience 3.3.2 Choosing the Target Function 3.3.3 Choosing a representation for the Target Function 3.3.4 Choosing an approximation algorithm for the Target Function 3.3.5 Final Design 3.4

Summary 3.5 Key Terms 3.6 Check Your Progress 3.0 Introduction As per the definition of machine learning, it enables a machine to automatically learn from the data provided. It also improves performance from an experience and predicts things without being explicitly programmed. Simply, when the training data is fed to a machine learning algorithm, this algorithm generates a mathematical model. This model helps the machine to make a prediction and take a decision without being explicitly programmed. It should be noted that while producing the training data, the more the machine will work with it, the more experience it will gain and more efficient result is produced.
Introduction, Basics and Types of Machine Learning Figure 3.1 Basic Machine Learning System For example, in an automatic or driverless Car, the training data is fed to the algorithm like how to drive a car on highway and street including the factors like speed limit, parking, stop at signal etc. On the basis of this training data, a logical mathematical model is created. The car will work according to this logical model. Also, the more data the data is fed the more efficient output is produced. The focus of this unit is to illustrate the basic steps for designing a learning system. Different types of learning models are also discussed. 3.1 Unit Objectives After completing this unit, the reader will be able to: ● Illustrate the basic steps for designing a learning system in detail. ● Gain knowledge about the different types of learning models. ● Discuss types of training experience and approximation for target functions. 3.2 Learning Models Before discussing the designing of a learning system, let's first discuss the different types of learning models. There are many types of learning models that can be used for learning. Some of them are: ● Linear classifier (numerical functions) where in finding the functions for classifying, we are going to focus on dividing these points with a straight line. This is called linear classification. ● Parametric (Probabilistic functions): A parametric model is one that can be parametrized by a finite number of parameters. It reduces the problem of
Introduction, Basics and Types of Machine Learning estimating a probability density function (pdf), discriminant, or regression function to estimating the values of a small number of parameters. Parametric estimation: all data instances affect the final global estimate. Examples of such models include Naïve Bayes and Hidden Markov models (HMM). ● Non-parametric (Instance-based functions): In this case, assumption is that similar inputs have similar outputs, and functions (pdf, discriminant, regression) change smoothly where similar data instances in the training data are found and their outputs are interpolated/ averaged. Examples include K- nearest neighbors, Kernel regression, Kernel density estimation, Local regression ● Non-metric (Symbolic functions): In this case, the learner must search the concept space to find the desired concept. The complexity of this concept space is a primary measure of the difficulty of a learning problem. Examples include Classification and regression tree (CART), ID3, etc. ● Aggregation: Bagging and Boosting are two examples of this technique. Bagging (bootstrap + aggregation) is the combining of many unstable predictors to produce an ensemble (stable) predictor. Each predictor in the ensemble is created by taking a bootstrap sample of the data. The bootstrap sample of N instances is obtained by drawing N examples at random, with replacement. Boosting is the combining of many weak predictors (e.g. tree stumps or 1 - R predictors) to produce an ensemble predictor. Each predictor is created by using a biased sample of the training data where Instances (training examples) with high error are weighted higher than those with lower error and hence difficult instances get more attention. Adaboost is a typical algorithm under this category.

68%	MATCHING BLOCK 33/189	W

Machine learning is concerned with using the right features to build the right models that achieve the right tasks. The collection of all possible outcomes, for a given problem, represents the sample space or instance space.

The
basic idea of Learning models has been divided into three categories: ●

100%	MATCHING BLOCK 34/189	W

Using a logical expression (Logical Models) ● Using the Geometry of the instance space (Geometric models) ● Using Probability to classify the instance space (Probabilistic models)

Introduction, Basics and Types of Machine Learning 3.2.1

98%	MATCHING BLOCK 35/189	W

Logical Models These models use a logical expression to divide the instance space into segments and hence construct grouping models. A logical expression is an expression that returns a Boolean value, i.e., a True or False outcome. Once the data is grouped using a logical expression, the data is divided into homogeneous groupings for the problem we are trying to solve. For example, for a classification problem, all the instances in the group belong to one class.

Logical models can be easily translated into rules that are understandable by humans. Such rules are easily organised in a tree structure, known as a feature tree. The idea of such a tree is that features are used to iteratively partition the instance space. The leaves of the tree therefore correspond to rectangular areas in the instance space (or hyper-rectangles). Depending on the task we are solving, we can then label the leaves with a class, a probability, a real value, and so on. Feature trees whose leaves are labeled with classes are commonly known as decision trees. Logical models often have different, equivalent formulations. For example, there are two instances x and y of the same class Healthy and the opposite class is Unhealthy. Two alternative formulations for this model can be; ● if $x = 1 \bigvee y = 1$ then Class = Healthy. else Class = Unhealthy. ● if $x = 0 \bigwedge y = 0$ then Class = Unhealthy. else Class = Healthy. The first of these alternative formulations combines the two rules in the original decision list by means of disjunction ('or'), denoted by $\bigvee$. The second one is a conjunctive condition ('and'), denoted by $\bigwedge$ for the opposite class (Unhealthy) and declares everything else as Healthy. 3.2.2 Geometric Models In logical models,

94%	MATCHING BLOCK 36/189	W

a logical expression is used to partition the instance space. Two instances are similar when they end up in the same logical segment. In geometric models, we consider models that define similarity by considering the geometry of the instance space. In Geometric models, features could be described as points in two

Introduction, Basics and Types of Machine Learning

100%	MATCHING BLOCK 37/189	W

dimensions (x- and y-axis) or a three-dimensional space (x, y, and z). Even when features are not intrinsically geometric, they could be modelled in a geometric manner (for example, temperature as a function of time can be modelled in two axes).

A geometric model is constructed directly in instance space, using geometric concepts such as lines, planes and distances. One main advantage of geometric classifiers is that they are easy to visualise, as long as we keep to two or three dimensions. It is important to keep in mind, though, that a Cartesian instance space has as many coordinates as there are features, which can be tens, hundreds, thousands, or even more. Such high-dimensional spaces are hard to imagine but are nevertheless very common in machine learning. Geometric concepts that potentially apply to high- dimensional spaces are usually prefixed with 'hyper-'. For instance, a decision boundary in an unspecified number of dimensions is called a hyperplane.

100%	MATCHING BLOCK 38/189	W

In geometric models, there are two ways we could impose similarity: ● We could use geometric concepts like lines or planes to segment (classify) the instance space. These are called Linear models. ● Alternatively, we can use the geometric notion of distance to represent similarity. In this case, if two points are close together, they have similar values for features and thus can be classed as similar. We call such models as Distance-based models. Linear Models Linear models are relatively simple. In this case, the function is represented as a linear combination of its inputs. Thus, if x 1 and x 2 are two scalars or vectors of the same dimension and a and b are arbitrary scalars, then ax 1 + bx 2 represents a linear combination of x 1 and x 2 . In the simplest case where f(x) represents a straight line, we have an equation of the form f(x) = mx + c where c represents the intercept and m represents the slope. Linear models are parametric, which means that they have a fixed form with a small number of numeric parameters that need to be learned from data. For example, in f (x) = mx + c, m and c are the parameters that we are trying to learn from the data. This technique is different from tree or rule models, where the structure of the model

is not
Introduction, Basics and Types of Machine Learning

93%	MATCHING BLOCK 39/189	W

fixed in advance. Linear models are stable, i.e., small variations in the training data have only a limited impact on the learning model.

95%	MATCHING BLOCK 40/189	W

As a result of having few parameters, Linear models have low variance and high bias. This implies that Linear models are less likely to overfit the training data than some other models. However, they are more likely to underfit. For example, if we want to learn the boundaries between countries based on labelled data, then linear models are not likely to give a good approximation. Distance-based Models Like Linear models, distance-based models are based on the geometry of data. As the name implies, distance-based models work on the concept of distance. In the context of Machine learning, the concept of distance is not based on merely the physical distance between two points. Instead, we could think of the distance between two points considering the mode of transport between two points.

For example,

100%	MATCHING BLOCK 41/189	W

in chess, the concept of distance depends on the piece used – for example, a Bishop can move diagonally. Thus, depending on the entity and the mode of travel, the concept of distance can be experienced differently. The distance metrics commonly used are Euclidean

and Manhattan. In a Cartesian coordinate system, distance can be measured by Euclidean distance, which is the square root of the sum of the squared distances along each coordinate: Euclidean To classify a new instance, we retrieve from memory the most similar training instance (i.e., the training instance with smallest Euclidean distance from the instance to be classified), and simply assign that training instance's class. This classifier is known as the nearest-neighbour classifier. Geometric models exploit these intuitions and are simple, powerful and allow many
Introduction, Basics and Types of Machine Learning variations with little effort. For instance, instead of using Euclidean distance, which can be sensitive to outliers, other distances can be used such as Manhattan distance, which sums the distances along each coordinate: Manhattan 3.2.3

100%	MATCHING BLOCK 42/189	W

Probabilistic Models The third family of machine learning algorithms is the probabilistic models.

97% **MATCHING BLOCK 43/189** W

The probabilistic models use the idea of probability to classify new entities. Probabilistic models see features and target variables as random variables. The process of modelling represents and manipulates the level of uncertainty with respect to these variables. There are two types of probabilistic models: Predictive and Generative. Predictive probability models use the idea of a conditional probability distribution P (Y | X) from which Y can be predicted from X. Generative models estimate the joint distribution P (Y, X). Once we know the joint distribution for the generative models, we can derive any conditional or marginal distribution involving the same variables. Thus, the generative model is capable of creating new data points and their labels, knowing the joint probability distribution. The joint distribution looks for a relationship between two variables. Once this relationship is inferred, it is possible to infer new data points. Naïve Bayes is an example of a probabilistic classifier. This can be accomplished using the Bayes rule: The Naïve Bayes algorithm is based on the idea of Conditional Probability. Conditional probability is based on finding the probability that something will happen, given that something else has already happened. The task of the algorithm then is to look at the evidence and to determine the likelihood of a specific class and assign a label

Introduction, Basics and Types of Machine Learning accordingly to each entity. Many of these models are based around the following idea. Let X denote the variables we know about, e.g., our instance's feature values; and let Y denote the target variables we're interested in, e.g., the instance's class. The key question in machine learning is how to model the relationship between X and Y. The statistician's approach is to assume that there is some underlying random process that generates the values for these variables, according to a well-defined but unknown probability distribution. We want to use the data to find out more about this distribution. Table 3.1 Some broad categories of models for all three learning models Logical

52% **MATCHING BLOCK 44/189** W

Models Geometric Models Probabilistic Models Decision Trees, Random Forest Trees, etc. K-nearest neighbors, Linear regression, Support Vector Machine (

SVM), Logistic regression, etc. Naïve Bayes, Gaussian process regression, conditional random field, etc. 3.3 Steps for Designing a Learning System As already stated, according to Tom Mitchell, "A computer program is said to be learning from training experience (E), with respect to some task (T). Thus, the performance measure (P) is the performance at task T.

100% **MATCHING BLOCK 45/189** W

The learning process starts with task T, performance measure P and training experience E and

the

91% **MATCHING BLOCK 46/189** W

objective is to find an unknown target function. The target function is an exact knowledge to be learned from the training experience and its unknown. For example, in a case of credit approval, the learning system will have customer application records as experience and task would be to classify whether the given customer application is eligible for a loan. So in this case, the training examples can be represented as (x 1 ,y 1) (x 2 ,y 2).. (x n ,y n), where X represents customer application details and y represents the status of credit approval.

100% **MATCHING BLOCK 47/189** W

So the target function to be learned in the credit approval learning system is a mapping function f:X→y. This function represents the exact knowledge defining the relationship between input variable X and output variable y.

Introduction, Basics and Types of Machine Learning Figure 3.2 Steps for designing learning system Let's discuss these steps in detail. 3.3.1 Type of Training Experience Choosing the training data or experience is the first task to be completed for designing a learning system. This experience will be fed to the machine learning algorithm. It should be noted that the data or experience that is fed to the algorithm, has a significant impact on the success or failure of the model. Thus, it is important to choose the training data or experience wisely. The success and failure of data can be affected by the following factors: ● The training experience will be able to provide direct or indirect feedback regarding choices. For example, while playing chess the training data will provide feedback to itself like instead of this move if this is chosen the chances of success increases. ● Second important attribute is the degree to which the learner will control the sequences of training examples. For example: when training data is fed to the machine then at that time accuracy is very less but when it gains experience while playing again and again with itself or opponent the machine algorithm will get feedback and control the chess game accordingly. ● Third important attribute is how it will represent the distribution of examples over which performance will be measured. In general, learning is most reliable when the training examples follow a distribution similar to that of future test
Introduction, Basics and Types of Machine Learning examples. In our chess learning scenario, the performance metric P is the percent of games the system wins in the world tournament. If its training experience E consists only of games played against itself, there is an obvious danger that this training experience might not be fully representative of the distribution of situations over which it will later be tested. For example, a Machine learning algorithm will get experience while going through a number of different cases and different examples. Thus, the algorithm will get more and more experience by passing through more and more examples and hence its performance will increase. To proceed with our design, let us decide that our system will train by playing games against itself. This has the advantage that no external trainer need be present, and it therefore allows the system to generate as much training data as time permits. We now have a fully specified learning task. Consider an example of the game- chess, for which the three elements will be defined as: ●

Task (T): To play chess. ● Performance measure (P): Total percent of the game won in the tournament. ● Training experience (E): A set of games played against itself.

In order to complete the design of the learning system, we must now choose 1. the exact type of knowledge to be learned 2. a representation for this target knowledge 3. a learning mechanism 3.3.2 Choosing the Target Function The next design choice is to determine exactly what type of knowledge will be learned and how this will be used by the performance program. Let us begin with a chess- playing program that can generate legal moves from any board state. The program

needs only to learn how to choose the best move from among

these legal moves. This learning task is representative of a large class of tasks for which the legal moves that define some large search space are known a priori, but for which the best search strategy is not known. Many optimization problems fall into this class, such as the
Introduction, Basics and Types of Machine Learning problems of scheduling and controlling manufacturing processes where the available manufacturing steps are well understood, but the best strategy for sequencing them is not. Choosing a target function means according to the data or experience fed to the algorithm, the system will choose the NextMove function, which will describe what type of legal moves should be taken. For example, while playing chess with the opponent, when the opponent will play then the machine learning algorithm will decide what be the number of possible legal moves taken in order to get success. The program

needs only to learn how to choose the best move from among

these legal moves. Assume a function NextMove such that: NextMove: $B \rightarrow M$ Here, B denotes the set of board states and M denotes the set of legal moves given a board state. NextMove is our target function. Although NextMove is an obvious choice for the target function in our example, this function will turn out to be very difficult to learn given the kind of indirect training experience available to our system. An alternative target function, and one that will turn out to be easier to learn in this setting, is an evaluation function that assigns a numerical score to any given board state. Let us call this target function V and again use the notation $V : B \rightarrow R$ to denote that V maps any legal board state from the set B to some real value (we use R to denote the set of real numbers). We intend for this target function V to assign

95% MATCHING BLOCK 51/189 W

higher scores to better board states. If the system can successfully learn such a target function V, then it can easily use it to select the best move from any current board position.

This can be accomplished by generating the successor board state produced by every legal move, then using V to choose the best successor state and therefore the best legal move.

99% MATCHING BLOCK 52/189 W

Let us therefore define the target value V (b) for an arbitrary board state b in B, as follows: 1. if b is a final board state that is won, then V (b) = 100 2. if b is a final board state that is lost, then V (b) = -100 3. if b is a final board state that is drawn, then V (b) = 0 4. if b is not a final state in the game, then V(b) = V(b'), where b' is the best final board state that can be achieved starting from b and playing optimally until the

Introduction, Basics and Types of Machine Learning end of the game (assuming the opponent plays optimally, as well).

92% MATCHING BLOCK 53/189 W

The case (4) is a recursive definition and to determine the value of V(b) for a particular board state, it performs the search ahead for the optimal line of play, all the way to the end of the game. So this definition is not efficiently computable by our chess playing program, we say that it is a non-operational definition. The goal of learning in this case is to discover an operational description of V; that is, a description that can be used by the chess playing program to evaluate states and select moves within realistic time bounds.

Thus, we have reduced the learning task in this case to the problem of discovering an operational description of the ideal target function V.

100% MATCHING BLOCK 54/189 W

It may be very difficult in general to learn such an operational form of V perfectly.

In fact, we often

100% MATCHING BLOCK 55/189 W

expect learning algorithms to acquire only some approximation to the target function,

and for this reason the process of learning the target function is often called function approximation. 3.3.3 Choosing a representation for the Target Function After knowing all the possible legal moves, the next step is to choose the optimized move using any suitable representation. These representations can be in the form of linear equations, hierarchical graph representation, Tabular form, etc. The NextMove function will move the Target move out of the possible moves. This will provide a higher success rate. Like, while playing chess, there are five possible moves. The machine will choose the most optimized move that will be successful. We need to

88% MATCHING BLOCK 56/189 W

choose a representation that the learning algorithm will use to describe the function

NextMove.

90% MATCHING BLOCK 57/189 W

The function NextMove will be calculated as a linear combination of the following board features: ● x1:

the number of black pieces on the board ● x2: the number of white pieces on the board ● x3: the number of black kings on the board ● x4: the number of white kings on the board ● x5: the

66%	MATCHING BLOCK 58/189	W
number of black pieces threatened by white (i.e., which can be captured on white's next turn) ● x6:		

the number of white pieces threatened by black V(b) = u 0 + u 1 x 1 + u 2 x 2 + u 3 x 3 + u 4 x 4 + u 5 x 5 + u 6 x 6 Introduction, Basics and Types of Machine Learning Here u 0 , u 1 , u 2 , u 3 , u 4 , u 5 , and u 6 are the coefficients or weights that will be chosen (learned) by

84%	MATCHING BLOCK 59/189	W
the learning algorithm. 3.3.4 Choosing an approximation algorithm for the Target Function Training data		

is not just enough to help in choosing an optimized move out of all the possible moves. The training function V had to consider a certain set of examples through which the training data approximates which steps are chosen. Each example will describe

81%	MATCHING BLOCK 60/189	W
a specific board state b and the training value V train (b) for b. In other words, each training example is an ordered pair		

of the form (b, V train (b)). For instance, the following training example describes a board state b in which black has won the game (note x 2 = 0 indicates that white has no remaining pieces) and for which the target function value V train (b) is therefore +100. ((x 1 = 3, x 2 = 0, x 3 = 1, x 4 = 0, x 5 = 0, x 6 = 0), +100) The machine that also provides feedback on it. For instance, when training data of playing chess is fed to an algorithm, it can then measure which step should be chosen in the next move and what is its success rate. To learn the target function NextMove,

67%	MATCHING BLOCK 61/189	W
we require a set of training examples, each describing a specific board state b and the training value (Correct Move) y for b.		

The training algorithm learns/ approximates the coefficients u 0 , u 1 , u 2 , u 3 , u 4 , u 5 , and u 6 with the help of these training examples by estimating and adjusting these weights. 3.3.5 Final Design The final design is created at last when the system goes from number of examples, failures and success, correct and incorrect decisions and what will be the next step etc.

94%	MATCHING BLOCK 62/189	W
The final design of our chess learning system can be naturally described by four distinct program modules that represent the central components in many learning systems. 1. The performance System:		

It is the module that must solve the given performance task, in this case playing chess, by using the learned target function(s). It takes an instance of a new problem (new game) as input and produces a trace of its solution (game history) as output. In our case, the strategy used by the Performance System to select its next move at each step is determined by the learned V evaluation function. Therefore, we expect its
Introduction, Basics and Types of Machine Learning performance to improve as this evaluation function becomes increasingly accurate. 2. The Critic: It takes as input the history or trace of the game and produces as output a set of training examples of the target function. As shown in figure 3.3, each training example in this case corresponds to some game state in the trace, along with an estimate V train , of the target function value for this example. 3. The Generalizer: It takes as input the training examples and produces an output hypothesis that is its estimate of the target function. It generalizes from the specific training examples, hypothesizing a general function that covers these examples and other cases beyond the training examples. In our example, the Generalizer corresponds to the LMS (Least Mean Square) algorithm, and the output hypothesis is the function f described by the learned weights u o to u 6 . 4. The Experiment Generator: It takes as input

88%	MATCHING BLOCK 63/189	W
the current hypothesis (currently learned function) and outputs a new problem (i.e., initial board state) for the Performance System to explore.		

Its role is to pick new practice problems that will maximize the learning rate of the overall system. In our example, the Experiment Generator follows a very simple strategy: It always proposes the same initial game board to begin a new game. More sophisticated strategies could involve creating board positions designed to explore particular regions of the state space. Figure 3.3 Final Design of the Chess Learning Program
Introduction, Basics and Types of Machine Learning The design choices we made for our checkers program produce specific instantiations for the performance system, critic; generalizer, and experiment generator. Many machine learning systems can-be usefully characterized in terms of these four generic modules. 3.4 Summary ● The collection of all possible outcomes, for a given problem, represents the sample space or instance space. ● The basic idea of Learning models has been divided into three categories:

using a logical expression (Logical Models), using the Geometry of the instance space (Geometric models), and using Probability to classify the instance space (Probabilistic models). ●

There are five steps for designing a learning system: Type of Training experience, choosing target function, Choosing representation of target function, Choosing function approximation, and Final Design. ● The training experience will be able to provide direct or indirect feedback regarding choices. 3.5 Key Terms ● Feature Tree: Logical models can be easily translated into rules that are understandable by humans. Such rules are easily organised in a tree structure, known as a feature tree. ● Decision Tree: Feature trees whose leaves are labeled with classes are commonly known as decision trees. ● Likelihood: The likelihood term, P(Y|X) is the probability of getting a result for a given value of the parameters. ● Function Approximation: We often

expect learning algorithms to acquire only some approximation to the target function,

and for this reason the process of learning the target function is often called function approximation. 3.6 Check Your Progress Short-Answer Type
Introduction, Basics and Types of Machine Learning Q1) The second step in designing a learning system is: a) Type of training experience b) Choosing function approximation c) Choosing Target Function d) Final Design Q2) The training experience will be able to provide ______ and _____ feedback regarding choices. Q3) ______ is a Target function for the chess playing algorithm. Q4) ________ takes as input the history or trace of the game and produces as output a set of training examples of the target function. Long-Answer Type Q1) Write a short note on Geometric Models. Q2) Differentiate between predictive and generative probabilistic models. Q3) List the steps for designing a learning system. Q4) What are the attributes that affect the success and failure of training data? Q5) What are the four program modules that represent the final design of a learning system? References: Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997.
Introduction, Basics and Types of Machine Learning Unit 4 – Machine Learning Framework Structure 4.0 Introduction 4.1 Unit Objectives 4.2 Machine learning frameworks and Libraries 4.3 Machine learning frameworks and libraries without special hardware support 4.3.1 Shogun 4.3.2 Scikit-Learn 4.3.3 Data analytic frameworks and libraries 4.4 Deep learning frameworks and libraries 4.4.1 TensorFlow 4.4.2 PyTorch 4.5 Summary 4.6 Key Terms 4.7 Check Your Progress 4.0 Introduction So far, we have discussed the basic concept of machine learning, types of machine learning, and designing a learning system. This unit illustrates machine learning framework and libraries. It also depicts the different types of frameworks that are being used in real-world applications. Machine learning Frameworks have become a standard paradigm these days. They not only democratize the development of machine learning algorithms but also speed up the process. And not just the open source community, but large organizations are also realizing the need to launch their own frameworks. A Machine Learning framework is an interface, library or tool which allows developers to more easily and quickly build machine learning models, without getting into the fundamentals of the underlying algorithms. It provides a clear, concise way for defining machine learning models using a collection of pre-built, optimized components. Some of the key features of good machine learning framework are: ➢ Optimized for performance ➢ Developer friendly. The framework utilizes traditional ways of building models. ➢ Is easy to understand and code on

Introduction, Basics and Types of Machine Learning ➢ Is not completely a black box ➢ Provide parallelization to distribute the computation process Simply, an efficient machine learning framework reduces the complexity of machine learning, making it accessible to more developers. 4.1 Unit Objectives After completing this unit, the reader will be able to: ● Gain knowledge about the machine learning frameworks and libraries. ● Discuss the deep learning frameworks and libraries. ● Illustrate the different types of machine learning frameworks. 4.2 Machine learning frameworks and Libraries The number of machine learning algorithms, as well as their different software implementations, is quite large. The common goal of many software tools is to facilitate the complicated data analysis process and to propose integrated environments on top of standard programming languages. These tools are designed for various purposes: as analytics platforms, predictive systems, recommender systems, processors (for images, sound or language). A number of them are oriented to fast processing and streaming of large-scale data, while others are specialized in implementing machine learning algorithms including neural networks and deep learning. The basic difference between a framework and a library can be stated as a library performs specific, well-defined operations, while a framework is a shell where the application defines the need of the operation. The shell still has code to link up the parts but the most important work is done by the application. The framework defines the concept but the application defines the fundamental functionality that end-users care about. Network protocols, compression, image manipulation, string utilities, regular expression evaluation, etc. are examples of libraries. Web application system, Plug-in manager, GUI system are some examples of frameworks. It is important to emphasize that there is no single tool suitable for every problem and often a combination of them is needed to succeed. Figure 4.1 represents a comprehensive grouped overview of machine learning frameworks and libraries. We have seen an explosion in developer tools and platforms related to machine
Introduction, Basics and Types of Machine Learning learning and artificial intelligence during the last few years. From cloud-based cognitive APIs to libraries to frameworks to pre-trained models, developers make many choices to infuse AI into their applications. AI engineers and researchers choose a framework to train machine learning models. These frameworks abstract the underlying hardware and software stack to expose a simple API in languages such as Python and R. For example, an ML developer can leverage the parallelism offered by GPUs to accelerate a training job without changing much of the code written for the CPU. These frameworks expose simpler APIs that translate to complex mathematical computations and numerical analysis often needed for training the machine learning models. Apart from training, the machine learning frameworks simplify inference. It is the process of utilizing a trained model for performing prediction or classification of live data. Machine learning frameworks are used in the domains related to computer vision, natural language processing, and time-series predictions. They are also used with structured data typically represented in a tabular form to perform linear regression and logistic regression to predict or classify the data. From figure 4.1, it can be depicted that there are many machine learning tools available based on their usage and application. In the upcoming sections, we will discuss some of the most widely used machine learning and deep learning frameworks.
Introduction, Basics and Types of Machine Learning Figure 4.1 An overview of machine learning frameworks and libraries (Source- Artificial Intelligence Review (2019) 52:77–124) 4.3 Machine learning frameworks and libraries without special hardware support With the rapid development of machine and deep learning in various fields, the big companies and research teams have developed independent and unique tools. There are certain machine learning frameworks and libraries that are without special hardware support. Such frameworks are very common and feasible to apply in different fields. 4.3.1 Shogun Shogun has been under active development since 1999. It is an open-source general purpose machine learning library. Shogun offers a wide range of efficient and unified machine learning methods built on an architecture written in C++ and licensed under
Introduction, Basics and Types of Machine Learning GNU GPLv3 license. 5 Shogun also covers a wide range of regression and classification methods as well as a number of linear methods, algorithms to train hidden Markov models, statistical testing, clustering, distance counting, and model evaluations and many more. Currently, Shogun is developed by a team of diverse volunteers and it is a sponsored project of NumFOCUS since 2017. The main perspective of Shogun is that the underlying algorithms are transparent, accessible and that anyone should be able to use them for free. The library Support Vector Machine (SVM) contains 15 implementations in combination with more than 35 kernel implementations, which can be furthermore combined/ constructed by sub-kernel weighting. 6 ● Breath-oriented machine learning /Data Mining toolbox with a lot of standard and cutting-edge machine learning algorithms. It has been successfully used in speech and handwriting recognition, medical diagnosis, bioinformatics, computer vision, object recognition, stock market analysis, network security, intrusion detection, and many more. Shogun can be used transparently in many languages and environments such as Python, Octave, R, Java/Scala, Lua, C#, and Ruby. It offers bindings to other sophisticated libraries including, LibSVM/LibLinear, SVMLight, LibOCAS, libqp, Vowpal Wabbit, Tapkee, SLEP, GPML and with future plans of interfacing TensorFlow and Stan. Merits of Shogun ● Open-source, cross-platform, API-oriented, the oldest and still maintained library with core implementation in C++. ● Bindings to many other machine learning libraries, programming interfaces in many languages. Demerits of Shogun ● The code is not easily maintainable or extendable as most of the codes have been written by researchers for their studies for a long time. 5 Sonnenburg S, Henschel S et. al., The shogun machine learning toolbox, Journal of Machine Learning Research (1799–1802), 2010. 6 http://www.shogun.ml/

Introduction, Basics and Types of Machine Learning ● Lack of documentation. ● Suitable mainly for academic use. Note:- The latest version of Shogun is 6.1.3 (December 2017). 4.3.2 Scikit-Learn Scikit-Learn is a widely known and popular open-source Python tool. It contains a comprehensive library of Data Mining /machine learning algorithms. 7 Since April 2016, Scikit-Learn has been provided in jointly-developed Anaconda for the Cloudera project on Hadoop clusters. The Scikit-Learn project was started as a Google Summer of Code project by David Cournapeau. Since 2015, it has been under active development sponsored by INRIA, Telecom ParisTech and occasionally Google through the Google Summer of Code. It extends the functionality of NumPy and SciPy packages with numerous data mining algorithms and provides functions to perform classification, regression, clustering, dimensionality reduction, model selection and preprocessing. It also uses the Matplotlib package for plotting charts. 8 ● General purpose, open-source, commercially usable, and popular Python machine learning tools. In addition to Scikit-Learn, Anaconda includes a number of popular packages for mathematics, science, and engineering for the Python ecosystem such as NumPy, SciPy and Pandas. Merits of Scikit-Learn ● Funded by INRIA, Telecom Paristech, Google and others. ● Well-updated and comprehensive set of algorithms and implementations. ● It is a part of many ecosystems; it is closely coupled with statistical and scientific Python packages. Demerits of Scikit-Learn ● API-oriented only. ● The library does not support GPUs. ● Only basic tools for Neural networks. 7 https://scikit-learn.org/stable/ 8 https://www.anaconda.com/
Introduction, Basics and Types of Machine Learning Note:- The latest released version of Scikit-Learn is 0.20.0 (September 2018), which came with declaration of dropping support for Python 3.4 in the upcoming version 0.21.0. 4.3.3 Data analytic frameworks and libraries The data analytic and visualization tools display analytics results in an interactive way. This can ease the understanding of difficult concepts and support decision makers. There are many data visualization packages at various levels of abstraction in R or Python; e.g., matplotlib, ggplot, seaborn, plotly, bokeh. Web-based notebooks/ applications have gained popularity in recent years. They are integrated with data analytic environments to create and share documents that contain data-driven live code, equations, visualizations and narrative text. The most well-known are: ➢ Jupyter notebook: Formerly, it was known as iPython notebook. It is an open- source application supporting; e.g., creation and sharing documents (notebooks), code, source equations, visualization and text descriptions for data transformation, numerical simulations, statistical modeling, data visualization and machine learning. ➢ Zeppelin: It is an interactive notebook designed for the processing, analysis and visualization of large data sets. It provides native support for Apache Spark distributed computing. Zeppelin allows them to extend their functionality through various interpreters; e.g., Spark, SparkSQL, Scala, Python, shell from Apache Spark analytics platform. Some open-source tools for data analytics, reporting and integration platforms are: Kibana, Grafana, Tableau, etc. The number of frameworks and libraries providing or using Machine learning /Neural Network /Deep learning techniques is high. Let's discuss some relevant subset of such frameworks. ● MatLab is a quite popular multi-paradigm numerical computing environment. It uses a proprietary programming language developed by MathWorks. MatLab has over 2 million users across industry and academia. The two most popular free alternatives to MatLab are GNU Octave (Octave 2018) and SciLab (SciLab
Introduction, Basics and Types of Machine Learning 2018). ● SAS (Statistical Analysis System) began as a project to analyse agricultural data at North Carolina State University in 1966. Currently, it is a proprietary software package written in C for advanced data analytics and business intelligence with more than 200 components. ● R is a free software environment for statistical computing and graphics including linear and non-linear modeling, classical statistical tests, time-series analysis, classification, and clustering. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. R is easy of use and extensible via packages. ● Python is a programming language created by Guido van Rossum and first released in 1991. Python is successfully used in thousands of real-world business applications around the world e.g., Google and YouTube. Primarily, Python was not adopted for machine learning because it was a general purpose programming language for research, development and production, at small and large scales. Python provides a dynamic type system and automatic memory management, with large and comprehensive libraries for scientific computation and data analysis. ● NumPy is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. NumPy stack has similar users to MatLab, GNU Octave, and SciLab. ● SciPy is an open-source Python library used for scientific computing and technical computing. SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, Pandas, and SymPy. ● Pandas is a Python package providing fast, flexible, and expressive data structures designed to make it easier to work with relational or labeled data. Its two primary data structures, Series (one-dimensional) and DataFrame (two- dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. 4.4 Deep learning frameworks and libraries Using GPU accelerators is already offered by many popular machine learning

Introduction, Basics and Types of Machine Learning frameworks and libraries. This speeds up the learning process with supported interfaces. 9 Figure 4.2 Popular deep learning frameworks and libraries layering in various abstraction implementation levels Some of them also allow the use of optimised libraries such as CUDA (cuDNN), and OpenCL to improve the performance even further. The main feature of many-core accelerators is that their massively parallel architecture allows them to speed up computations that involve matrix-based operations. The software development in the machine learning /deep learning direction community is highly dynamic and has various layers of abstraction as represented in figure 4.2. CUDA is a parallel-computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs. In GPU-accelerated applications, the sequential part of the workload runs on the CPU, which is optimized for single-threaded performance, while the compute intensive portion of the application runs on thousands of GPU cores in parallel. When using CUDA, developers program in popular languages such as C, C++, Fortran, Python and MATLAB and express parallelism through extensions in the form of a few basic keywords. The CUDA Toolkit from NVIDIA provides everything you need to develop GPU-accelerated applications. The CUDA Toolkit includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime. 10 9 Kalogeiton V, Lathuiliere S, Luc P, Lucas T, Shmelkov K, Deep learning frameworks: Tensorflow, Theano, Keras, Torch and Caffe, 2016. 10 https://developer.nvidia.com

Introduction, Basics and Types of Machine Learning The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. Deep learning researchers and framework developers worldwide rely on cuDNN for high-performance GPU acceleration. It allows them to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning. cuDNN accelerates widely used deep learning frameworks, including Caffe2, Chainer, Keras, MATLAB, MxNet, PaddlePaddle, PyTorch, and TensorFlow. For access to NVIDIA optimized deep learning framework containers that have cuDNN integrated into frameworks, visit NVIDIA GPU CLOUD to learn more and get started. Some of the frameworks of deep learning are discussed below. 4.4.1 TensorFlow TensorFlow is an open-source software library for numerical computation using data flow graphs. TensorFlow was created and is maintained by the Google Brain team within Google's Machine Intelligence research organization for machine learning and deep learning. It is currently released under the Apache 2.0 open-source license. TensorFlow is designed for large-scale distributed training and inference. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The distributed TensorFlow architecture contains distributed master and worker services with kernel implementations. These include 200 standard operations, including mathematical, array manipulation, control flow, and state management operations, written in C++. TensorFlow was designed to be used both in research, development, and production systems. It can run on a single CPU system, GPUs, mobile devices, and large-scale distributed systems of hundreds of nodes. In addition, TensorFlow Lite is a lightweight solution for mobile and embedded devices. It enables on-device machine learning inference with low latency and a small binary size but has coverage for a limited set of operators. It also supports hardware acceleration with the Android Neural Networks API. TensorFlow programming interfaces include APIs for Python and C++ and

Introduction, Basics and Types of Machine Learning developments for Java, GO, R, and Haskell are on the way. TensorFlow is also supported in Google and Amazon cloud environments. Merits of TensorFlow ● By far the most popular deep learning tool, open-source, fast-evolving, supported by Google. ● Numerical library for dataflow programming that provides the basis for deep learning research and development. ● Efficiently works with mathematical expressions including multi-dimensional arrays. ● Supports GPU/ CPU computing, is efficient in multi-GPU settings, mobile computing, high scalability of computation across machines and huge data sets. Demerits of TensorFlow ● Still lower-level API, difficult to use directly for creating deep learning models. 4.4.2 PyTorch PyTorch is a Python library for GPU-accelerated deep learning. The library is a Python interface of the same optimized C libraries that Torch uses. It has been developed by Facebook's Artificial Intelligence research group since 2016. Torch is a scientific computing framework with wide support for machine learning algorithms based on the Lua programming language. Torch is supported and used by Facebook, Google, DeepMind, Twitter, and many other organizations and it is freely available under a BSD license. It uses an object-oriented paradigm and is implemented in C++. It is mainly used for large-scale learning (speech, image, and video applications), supervised learning, unsupervised learning, reinforcement learning, Neural Networks, optimization, graphical models, and image processing. PyTorch is written in Python, C, and CUDA. The library integrates acceleration libraries such as Intel MKL and NVIDIA (cuDNN, NCCL). At the core, it uses CPU and GPU Tensor and Neural Network backends written as independent libraries on a C99 API. PyTorch uses a technique called reverse-mode auto-differentiation, which allows changing the way a network behaves with small effort (i.e. dynamic computational

Introduction, Basics and Types of Machine Learning graph or DCG). The library is used by both the scientific and industrial communities. Merits of PyTorch • Dynamic computational graph (reverse-mode auto-differentiation). • Supports automatic differentiation for NumPy and SciPy. • Elegant and flexible Python programming for development (Caffe2PyTorch 2018). • Supports the Open Neural Network Exchange (ONNX) format, which allows to easily transform models between CNTK, Caffe2, PyTorch, MXNet, and other DL tools. Demerits of PyTorch • Does not have interfaces for monitoring and visualization like TensorFlow. • Comparatively, PyTorch is a new deep learning framework and currently has less community support. Recently, newly distributed frameworks have emerged to address the scalability of algorithms for Big Data analysis using the MapReduce programming model, being Apache Hadoop and Apache Spark the two most popular implementations. The main advantages of these distributed systems is their elasticity, reliability, and transparent scalability in a user-friendly way. They are intended to provide users with easy and automatic fault-tolerant workload distribution without the inconveniences of taking into account the specific details of the underlying hardware architecture of a cluster. These popular distributed computing frameworks are not mutually exclusive technologies with GPUs, although they aim at different scaling purposes. These technologies can complement each other and target complementary computing scopes. 4.5 Summary • A Machine Learning framework is an interface, library or tool which allows developers to more easily and quickly build machine learning models, without getting into the fundamentals of the underlying algorithms. • The basic difference between a framework and a library can be stated as a library performs specific, well-defined operations, while a framework is a shell where the application defines the need of the operation.
Introduction, Basics and Types of Machine Learning • Machine learning frameworks are used in the domains related to computer vision, natural language processing, and time-series predictions. • Shogun and Scikit-Learn are popular machine learning frameworks without special hardware support. • TensorFlow and Pytorch are popular deep learning frameworks. • New distributed frameworks utilize MapReduce programming model for machine learning/ deep learning frameworks. 4.6 Key Terms • Inference: It is the process of utilizing a trained model for performing prediction or classification of live data. • Lua: Lua is a powerful, efficient, lightweight, embeddable scripting language. It supports procedural programming, object-oriented programming, functional programming, data-driven programming, and data description. • CUDA: It is a parallel-computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). • Keras: Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research. 4.7 Check Your Progress Short-Answer Type Q1) Shogun and Scikit-Learn are types of _______ frameworks. Q2) A framework is a shell where the application defines the need of the operation. (True/ False?) Q3) PyTorch does not have interfaces for monitoring and visualization. (True/ False?) Q4) _____ is a parallel-computing platform and programming model for general computing on graphical processing units (GPUs). Long-Answer Type Q1) Write a short note on PyTorch. Is it better than TensorFlow? Q2) What is a machine learning framework? List the features of a good machine learning framework.
Introduction, Basics and Types of Machine Learning Q3) Discuss the basic features, merits, and limitations of scikit-Learn. Q4) Differentiate between machine learning and deep learning frameworks. Q5) Define deep learning frameworks. Explain any one framework in detail. References: Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey, Giang Nguyen et. al., CrossMark, Artificial Intelligence Review, 52:77-124, 2019.
Supervised Learning MODULE: II SUPERVISED LEARNING
Supervised Learning
Supervised Learning Unit 5 – Decision Trees Structure 5.0 Introduction 5.1 Unit Objectives 5.2 Decision Tree: Introduction 5.3 Decision Trees as Classifier 5.4 Induction of Decision Trees 5.5 Binary Split of a Numeric Attribute 5.6 Pruning 5.7 Converting the Decision Tree into Rules 5.8 Summary 5.9 Key Terms 5.10 Check Your Progress 5.0 Introduction As already discussed,

supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs.

Classification is
probably the most widely used form of machine learning and has been used to solve many interesting and often difficult real-world problems. It is the process of predicting the class of the given data. Classes can be targets/ labels or categories. The task of approximating a mapping function (f) from input variables (x) to discrete output variables (y) is known as Classification Predictive modeling. A classifier utilizes some training data to understand how given input variables relate to the class.
Classification belongs to the category of supervised learning where the targets are also provided with the input data.
There are two types of learners in classification: lazy learners and eager learners. Lazy learners store the training data and wait until the testing data appears. When this data appears, classification is done on the basis of most related data in the stored training data. For example, K-nearest neighbor. Lazy learners have less training time but more predicting time as compared to eager learners. In the case of Eager Learners, a classification model is constructed on the basis of the given training data before receiving data for classification. It must commit to the single hypothesis, covering the entire instance space. Due to the time taken in model

Supervised Learning construction, eager learners take a long time to train and less time in predicting. For example, Decision tree, Naive Bayes, Artificial Neural Networks. There are a lot of classification algorithms available now but it is not possible to conclude which one is superior to the other. It depends on the application and nature of the available data set. For example, if the classes are linearly separable, the linear classifiers like Logistic regression, Fisher's linear discriminant can outperform sophisticated models and vice versa. Some of the most popular and efficient classifiers are discussed in the upcoming units. This unit illustrates the basics of decision trees. The reader will learn how to benefit from tree-pruning, and how to convert the induced tree to a set of rules. 5.1 Unit Objectives After completing this unit, the reader will be able to: ● Gain knowledge about the decision trees in machine learning. ● Discuss the use of decision trees as classifier. ● Illustrate the conversion of a decision tree into rules. 5.2 Decision Tree: Introduction The classification or regression models are built in the form of a tree structure, known as Decision tree. An if-then rule set is utilized by the decision tree. This rule set is mutually exclusive and exhaustive for classification. The training data is provided one at a time and consequently, the
rules are learned sequentially. Each time a rule is learned, the tuples covered by the rules are removed. This process continues on the
training set of data, until a termination condition is met.
Decision tree learning is one of the most widely used and practical methods for inductive inference.
It is a method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions. The decision tree learning methods search a completely expressive hypothesis space and thus avoid the difficulties of restricted hypothesis spaces. Their inductive bias is a preference for small trees over large trees. Decision Trees are a type of supervised machine learning,

93%	MATCHING BLOCK 67/189	W

where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final Supervised Learning outcomes and the decision nodes are where the data is split.

The
trees classify instances by sorting them down
the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute
of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.
An instance is classified by starting at the root node of the tree, testing the
attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is
then repeated for the subtree rooted at the new node. Let's consider an example of a typical learned decision tree, in which it classifies if Saturday mornings are suitable for playing badminton. Suppose the instance, (Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong) On sorting the above instance, the leftmost branch of this decision tree can be classified as a negative instance (i.e., this branch predicts that PlayBadminton = No). Figure 5.1 represents the required decision tree for the concept of PlayBadminton. Generally, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions. For example, the decision tree shown in Figure 5.1 corresponds to the expression (Outlook = Sunny $\wedge$ Humidity = Normal) $\vee$ (Outlook = Overcast) $\vee$ (Outlook = Rain $\wedge$ Wind = Weak) Figure 5.1 A decision tree for the concept PlayBadminton.
Supervised Learning
There are two types of decision trees: 1. Classification Trees:
Tree models where the target variable can take a discrete set of values are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. 2. Regression Trees: Decision trees where the target variable can take continuous values (
real numbers) like
the price of a house, or a patient's length of stay in a hospital,
are called regression trees.

Although a variety of decision tree learning methods have been developed with somewhat differing capabilities and requirements, decision tree learning is generally best suited to problems with the following characteristics: ● Instances are represented by attribute-value pairs: Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot). The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold). However, extensions to the basic algorithm allow handling real-valued attributes as well (e.g., representing Temperature numerically). ● The target function has discrete output values: The decision tree in Figure 5.1 assigns a boolean classification (e.g., yes or no) to each example. Decision tree methods easily extend to learning functions with more than two possible output values. A more substantial extension allows learning target functions with real- valued outputs, though the application of decision trees in this setting is less common. ● Disjunctive descriptions may be required: As noted above, decision trees naturally represent disjunctive expressions. ● The training data may contain errors: Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples. ● The training data may contain missing attribute values: Decision tree methods can be used even when some training examples have unknown values (e.g., if the Humidity of the day is known for only some of the training examples).
Supervised Learning Many practical problems have been found to fit these characteristics. Decision tree learning has therefore been applied to problems such as learning to classify medical patients by
their disease, equipment malfunctions by their cause, and loan applicants by their likelihood of defaulting on payments. Such problems, in which the task is to classify examples into one of a discrete set of possible categories, are often
referred to as classification problems. How does a Decision Tree Work? To summarize, a decision tree is a supervised learning algorithm that works for both discrete and continuous variables. It splits the dataset into subsets on the basis of the most significant attribute in the dataset. How the decision tree identifies this attribute and how this splitting is done is decided by the algorithms. The most significant predictor is designated as the root node, splitting is done to form sub-nodes called decision nodes, and the nodes which do not split further are terminal or leaf nodes. In the decision tree, the dataset is divided into homogeneous and non-overlapping regions. It follows a top-down approach as the top region presents all the observations at a single place which splits into two or more branches that further split. This approach is also called a greedy approach as it only considers the current node between the worked on without focusing on the future nodes. The decision tree algorithms will continue running until a stop criteria such as the minimum number of observations etc. is reached. Once a decision tree is built, many nodes may represent outliers or noisy data. Tree pruning method is applied to remove unwanted data. This, in turn, improves the accuracy of the classification model. To find the accuracy of the model, a test set consisting of test tuples and class labels is used. The percentages of the test set tuples are correctly classified by the model to identify the accuracy of the model. If the model is found to be accurate then it is used to classify the data tuples for which the class labels are not known. Some of the decision tree algorithms include Hunt's Algorithm, ID3, CD4.5, and CART.
Supervised Learning 5.3 Decision Trees as Classifier Consider a training set shown in Table 5.1 consisting of eight examples described by three attributes and labeled as positive or negative instances of a given class. To simplify the explanation of the basic concepts, we will assume that all attributes are discrete. Later, when the underlying principles become clear, we will slightly generalize the approach to make it usable also in domains with continuous attributes or with mixed sets of continuous and discrete attributes. Figure 5.2 shows a few example decision trees corresponding to the data in Table 5.1. The internal nodes represent attribute-value tests, the edges indicate how to proceed in the case of diverse test results, and the leafs 1 contain class labels. Table 5.1 Eight training data set examples having three attributes and classified as positive and negative instances of a given class Example Crust Size Shape Filling Size Class E1 big circle small positive E2 small circle small positive E3 big square small negative E4 big triangle small negative E5 big square big positive E6 small square small negative E7 small square big positive E8 big circle big positive
Supervised Learning Figure 5.2 Example decision trees for all training examples listed in Table 6.1, tri, sq, and c stand for triangle, square, and circle, respectively Let us illustrate the process using the tree from Figure 5.2 (b). The root asks about the shape, each of whose three values is represented by one edge. In examples E1; E2, and E8, we find the value shape = circle which corresponds to the left edge. The example is sent down along this edge, ending in a leaf that is labeled positive. This indeed is the class common to all these three examples. In E4, shape = triangle, and the corresponding edge ends in a leaf labeled negative, again, the correct class. Somewhat more complicated is the situation with examples E3; E5; E6, and E7 where shape = square. For this particular value, the edge ends not in a leaf, but only at a test-containing node, this one inquiring about the value of filling-size. In the case of E5 and E7, the value is big, which leads to a leaf labeled with positive. In the other two examples, E3 and E6, the value is small, and this sends them to a leaf labeled with negative. It should be noted that the decision tree from Figure 5.2 (b) identifies the correct class for all training examples. The decision tree classifier offers one striking advantage of Interpretability. Considering table 5.1, if anybody asks why example E1 is deemed positive, the answer is, "because its shape is a circle." Other classifiers do not offer explanations of this kind. This means that one can interpret a decision tree as a set of rules. A domain expert inspecting these rules may then decide whether they are intuitively appealing, and whether they agree with his or her "human understanding" of the problem at hand. The expert may even be willing to suggest improvements to the tree; for instance, by pointing out spurious tests that have found their way into the data structure only on account of some random regularity in the data.

Supervised Learning 5.4 Induction of Decision Trees Decision tree induction is the method of learning the decision trees from the training set. The training set consists of attributes and class labels. Applications of decision tree induction include astronomy, financial analysis, medical diagnosis, manufacturing, and production. A decision tree is a flowchart tree-like structure that is made from training set tuples. The dataset is broken down into smaller subsets and is present in the form of nodes of a tree. The tree structure has a root node, internal nodes or decision nodes, leaf node, and branches. The root node is the topmost node. It represents the best attribute selected for classification. Internal nodes of the decision nodes represent a test of an attribute of the dataset leaf node or terminal node which represents the classification or decision label. The branches show the outcome of the test performed. Some decision trees only have binary nodes, that means exactly two branches of a node, while some decision trees are non-binary. Let us try our hand at creating a decision tree manually. Considering table 5.1, suppose we decide that the root node should test the value of shape. In the training set, three different outcomes are found: circle, triangle, and square. For each, the classifier will need a separate edge leading from the root. The first, defined by shape = circle, will be followed by examples T C = {E1, E2, E8}; the second, defined by shape = triangle, will be followed by T T = {E4}; and the last, defined by shape = square, will be followed by T S = {E3, E5, E6, E7}. Each of the three edges will begin at the root and end in another node, either an attribute test or a leaf containing a class label. Seeing that all examples in T C are positive, we will let this edge point at a leaf labeled with positive. Similarly, the edge followed by the examples from T T will point at a leaf labeled with negative. Certain difficulties will arise only in the case of the last edge because T S is a mixture of both classes. To separate them, we need another test, say, filling-size, to be placed at the end of the edge. This attribute can acquire two values, small and big, dividing T S into two subsets. Of these, T S-S = {E3, E6} is characterized by filling-size = small; the other, T S-B = {E5, E7}, is characterized by filling-size = big. All examples in T S-S are positive, and all examples in T S-B are negative. This allows us to let both edges end in leaves, the former labeled with positive, the latter with negative. At
Supervised Learning this moment, the tree-building process can stop because each training example presented to the classifier thus created will reach a leaf. It should be noticed that each node of the tree can be associated with a set of examples that pass through it or end in it. Starting with the root, each test divides the training set into disjoint subsets, and these into further subsets, and so on until each subset is “pure” in the sense that all its examples belong to the same class. This is why the approach is sometimes referred to as the divide-and-conquer technique. We can also determine the size of the decision tree. The smallest of the data structures in figure 5.2 consists of two attribute tests; the largest, of five. Differences of this kind may have a strong impact on the classifier’s behavior. The number of nodes in the tree is not the only criterion of size. Just as important is the number of tests that have to be carried out when classifying an average example. For instance, in a domain where shape is almost always circle or triangle (and only very rarely square), the average number of tests prescribed by the tree from figure 5.2 (b) will only slightly exceed, 1. Because both shape = circle and shape = triangle immediately point at leafs with class labels. But if the prevailing shape is square, the average number of tests approaches 2. Quite often, then, a bigger tree may result in fewer tests than a smaller one. Induction of Small Decision Trees When illustrating the behavior of the divide-and-conquer technique on the manual tree-building procedure, we picked the attributes at random. When doing so, we observed that some choices led to smaller trees than others. Apparently, the attributes differ in how much information they convey. For instance, shape is capable of immediately labeling some examples as positive (if the value is circle) or negative (if the value is triangle); but crust-size cannot do so unless assisted by some other attribute. Assuming that there is a way to measure the amount of information provided by each attribute, we have to formalize the technique for induction of decision trees by a pseudocode as represented in Table 5.2.
Supervised Learning Table 5.2 Induction of Decision trees Let T be the training set. grow (T): 1. Find the attribute, at, that contributes the maximum information about the class labels. 2. Divide T into subsets, T i , each characterized by a different value of at. 3. For each T i : If all examples in T i belong to the same class, then create a leaf labeled with this class; otherwise, apply the same procedure recursively to each training subset: grow (T i). 5.5 Binary Split of a Numeric Attribute Every possible test which splits the training dataset into several subsets will eventually lead to the construction of a complete decision tree, provided that at least two of the generated subsets are not empty. Several splitting rules have been proposed in the literature. CART 1 uses Gini index to measure the class diversity in the nodes of a decision tree. ID3 2 attempts to maximize the information gain achieved through the use of a given attribute to branch the tree. C4.5 3 To create a relatively small decision tree, the divide-and-conquer technique relies on one critical component: the ability to decide how much information about the class labels is conveyed by the individual attributes. This section introduces a mechanism to calculate this quantity. normalizes this information gain criterion in order to reduce the tree branching factor and adjusts C4.5 criterion to improve its performance with continuous attributes. The above splitting criteria provide a mechanism for ranking alternative divisions of the training set when building decision trees. There must be some way of generating possible divisions of the training set. In other words, the alternative tests which lead to different decision trees must be enumerated in order to be ranked. 1 L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, California, USA, 1984, ISBN 0-534-98054-6. 2 J.R. Quinlan, Induction on decision trees, Mach. Learn. 1 (1986) 81–106. 3 J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993, ISBN 1-55860-238-0.

Supervised Learning Most decision tree algorithms define a template of possible tests so that it is possible to examine all the tests which match with the template. Those tests usually involve a single attribute because it makes the trees easier to understand and sidesteps the combinatorial explosion that results if multiple attributes can appear in a single test. C4.5-like algorithms, which build multi-way decision trees for discrete attributes, check the value of such categorical attributes and build a branch for each value of the selected attribute. More complex tests are also allowed, by grouping values of the attribute in order to reduce the tree branching factor. In the extreme case, all attribute values are clustered into two groups in order to build binary trees (as in CART). However, we feel that this strategy makes the trees more complex to understand for human experts. When an attribute A is continuous, i.e. it has numerical values, a binary test is usually performed. This test compares the value of A against a threshold t: A $\leq$ t. Once you can determine that the best possible threshold is between v i and v i+1 , you must choose an accurate value for the threshold to be used in the decision tree. Most algorithms choose the midpoint of the interval [v i ,v i+1] as the actual threshold, that is This approach ensures that any threshold value used in the decision tree actually appears in the data, although it could be misleading if the value sample in the training dataset is not representative. Using binary splits on numerical attributes implies that the attributes involved should be able to appear several times in the paths from the root of the tree to its leaves. Although these repetitions can be simplified when converting the decision tree into a set of rules, they make the constructed tree more leafy, unnecessarily deeper, and harder to understand for human experts. 5.6 Pruning Decision-tree learners can create over-complex trees that do not generalise well from the training data. This is known as overfitting. Mechanisms such as pruning are necessary to avoid this problem. One of the important fundamental problems associated with decision trees is the issue of "overfitting" of the data which in turn

Supervised Learning results in poor generalization. This essentially means that the decision tree is so detailed that it can fit the training samples but fails to generalize when a hitherto unknown sample is presented that misleads to low predictive accuracy of new data. A solution to this problem is to prune the tree. Figure 5.3 Difference between well-fit and overfit decision tree

Pruning is a data compression technique in machine learning and search algorithms
that reduces the size of decision trees by removing sections of the tree that
are non- critical and redundant
to classify instances.

Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

One of the questions that arises in a decision tree algorithm is the optimal size of the final tree.

A tree that is too large risks overfitting the training data and poorly generalizing to new samples.

A small tree might not capture important structural information about the sample space. However, it is hard to tell when a tree algorithm should stop because it is impossible to tell if the addition of a single extra node will dramatically decrease error. This problem is known as the horizon effect. A common strategy is to grow the tree until each node contains a small number of instances then use pruning to remove nodes that do not provide additional information. Pruning should reduce the size of a learning tree without reducing predictive accuracy as measured by a cross-validation set. There are many techniques for tree pruning that differ in the measurement that is used to optimize performance. There are several approaches to avoiding overfitting in decision tree learning. These can be grouped into two classes: ● Pre-Pruning approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data.

Supervised Learning ● Post-Pruning approaches that
allow the tree to overfit the data, and then post- prune the
tree. Although the first of these approaches might seem more direct, the second approach of post-pruning overfit trees has been found to be more successful in practice. This is due to the difficulty in the first approach of estimating precisely when to stop growing the tree. Figure 5.4 Decision Tree before and after pruning Reduced Error Pruning One of the simplest forms of pruning is reduced error pruning. Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected then the change is kept. While somewhat naive, reduced error pruning has the advantage of simplicity and speed. Using a separate set of data to guide pruning is an effective approach provided a large amount of data is available. The major drawback of this approach is that when data is limited, withholding part of it for the validation set reduces even further the number of examples available for training. The following section presents an alternative approach to pruning that has been found useful in many practical situations where data is limited. Many additional techniques have been proposed as well, involving partitioning the available data several different times in multiple ways, then averaging the results. Rule Post- Pruning In practice, one quite successful method for finding high accuracy hypotheses is a technique we shall call rule post-pruning. A variant of this pruning method is used by C4.5, which is an outgrowth of the original ID3 algorithm. Rule post-pruning involves the following steps:

Supervised Learning ● Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur. ● Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node. ● Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy. ● Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances. 5.7 Converting the Decision Tree into Rules One of the advantages of decision trees in comparison with the other classifiers is their interpretability. Any sequence of tests along the path from the root to a leaf represents an if-then rule, and this rule explains why the classifier has labeled a given example with this or that class. Rules Generated by a Decision Tree It is easy to convert a decision tree to a set of rules. It is enough to notice that a leaf is reached through a series of edges whose specific choice is determined by the results of the attribute tests encountered along the way. Each leaf is thus associated with a concrete conjunction of test results. For an example, let us write down the complete set of rules for the positive class as obtained from the decision tree in figure 5.2 (a). if crust-size=big AND filling-size=big then positive if crust-size=big AND filling-size=small AND shape=circle then positive if crust-size=small AND shape=circle then positive if crust-size=small AND (shape=square OR triangle) AND filling-size=big then positive else negative Note the default class, negative, in the last line. An example is labeled with the default class if all rules fail, which means that the value of the if -part of each rule is false. We notice that, in this two-class domain, we need to consider only the rules resulting in
Supervised Learning the pos class, the other class being the default option. We could have done it the other way round, considering only the rules for the negative class, and making pos the default class. This would actually be more economical because there are only two leafs labeled with the negative, and therefore only two corresponding rules. The reader is encouraged to write down these two rules by way of a simple exercise. At any rate, the lesson is clear: in a domain with K classes, only the rules for K-1 classes are needed, the last class serving as the default. 5.8 Summary ● Classification is probably the most widely used form of machine learning and has been used to solve many interesting and often difficult real-world problems. ● There are two types of learners in classification: lazy learners and eager learners. ● The classification or regression models are built in the form of a tree structure, known as Decision tree. ●

The tree can be explained by two entities, namely decision nodes and leaves.

Decision tree induction is the method of learning the decision trees from the training set. ● Decision-tree learners can create over-complex trees that do not generalise well from the training data. This is known as overfitting. ●
Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting. 5.9 Key Terms ● Lazy learners: They store the training data and wait until the testing data appears. ● Eager Learners: In the case of eager learners, a classification model is constructed on the basis of the given training data before receiving data for classification. ● Gini index: The Gini index is a measure of the distribution of income across a population. A higher Gini index indicates greater inequality, with high-income individuals receiving much larger percentages of the total income of the population.
Supervised Learning ● Interpretability: It is the degree to which a model can be understood in human terms. 5.10 Check Your Progress Short-Answer Type Q1) Decision tree is a display of an algorithm. (True/ False?) Q2) Which of the following are decision tree nodes? i) Decision Nodes ii) End Nodes iii) Chance Nodes iv) All of the above Q3)
Tree models where the target variable can take a discrete set of values are called _______.
Q4) Decision tree follows a _________ approach. Q5) The decision tree classifier offers one striking advantage of Interpretability. (True/ False?) Long-Answer Type Q1)
Explain the concept of a decision tree with an example. Q2) What are the different types of decision trees?
Q3) Differentiate between classification and regression trees? Q4) What is overfitting in decision tree learning? How to overcome it? Q5) Explain the Reduced Error Pruning. References: Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997. An Introduction to Machine Learning, 2 nd edition, Miroslav Kubat, Springer, 2017. Building multi-way decision trees with numerical attributes, Fernando Berzal, Juan- Carlos Cubero, Nicolas Marin, Daniel Sanchez, 2003.
Supervised Learning Unit 6 – Classification and Regression Trees Structure 6.0 Introduction 6.1 Unit Objectives 6.2 Classification Trees 6.3 Regression Trees 6.4 Advantages and Limitations 6.5 CART Algorithm 6.6 Regression Analysis 6.7 Types of Regression 6.7.1 Linear Regression 6.7.2 Logistic Regression 6.8 Summary 6.9 Key Terms 6.10 Check Your Progress 6.0 Introduction We have already discussed a basic introduction to classification and regression. To memorize, classification is the process of finding or discovering a model or function which helps in separating the data into multiple categorical classes i.e. discrete values. In classification, data is categorized under different labels according to some parameters given in input and then the labels are predicted for the data. Regression is the process of finding a model or function for distinguishing the data into continuous real values instead of using classes or discrete values. It can also identify the distribution movement depending on the historical data. Because a regression predictive model predicts a quantity, therefore, the skill of the model must be reported as an error in those predictions. Decision trees also follow the concept of classification and regression. In this unit, we will discuss classification and regression trees in detail. The concept of regression and types of regression are also illustrated. 6.1 Unit Objectives After completing this unit, the reader will be able to: ● Gain knowledge about classification and regression trees.
Supervised Learning ● Discuss the advantages and limitations of classification and regression trees. ● Illustrate the Regression analysis and types of

96%	MATCHING BLOCK 69/189	W

regression. 6.2 Classification Trees A classification tree is an algorithm where the target variable is fixed or categorical. The algorithm is then used to identify the "class" within which a target variable would most likely fall. An example of a classification-type problem would be determining who will or will not subscribe to a digital platform; or who will or will not graduate from high school. These are examples of simple binary classifications where the categorical dependent variable can assume only one of two, mutually exclusive values. In other cases, you might have to predict among a number of different variables. For instance, you may have to predict which type of smartphone a consumer may decide to purchase. In such cases, there are multiple values for the categorical dependent variable. Figure 6.1 represents how a classic classification tree looks like.

Figure 6.1 An example of Classification Tree Classification Tree Analysis (CTA) is a type of machine learning algorithm used for classifying remotely sensed and ancillary data in support of land cover mapping and analysis. A classification tree is a structural mapping of binary decisions that lead to a decision about the class (interpretation) of an object (such as a pixel). CTA is an analytical procedure that takes examples of known classes (i.e., training data) and constructs a decision tree based on measured attributes such as reflectance. In TerrSet the CTA module is based on the C4.5 algorithm. In essence, the algorithm iteratively selects the attribute (such as reflectance band) and value that can split a set of samples into two groups, minimizing the variability within each subgroup
Supervised Learning while maximizing the contrast between the groups. 6.3

100%	MATCHING BLOCK 70/189	W

Regression Trees A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value. As an example of a regression type problem, you may want to predict the selling prices of a residential house, which is a continuous dependent variable. This will depend on both continuous factors like square footage as well as categorical factors like the style of home, area in which the property is located and so on.

Figure 6.2 shows an example of a regression tree. Figure 6.2 An example of Regression Tree All regression techniques contain a single output (response) variable and one or more input (predictor) variables. The output variable is numerical. The general regression tree building methodology allows input variables to be a mixture of continuous and categorical variables. A decision tree is generated when each decision node in the tree contains a test on some input variable's value. The terminal nodes of the tree contain the predicted output variable values. A Regression tree may be considered as a variant of decision trees, designed to approximate real-valued functions, instead of being used for classification methods. Binary Recursive Partitioning is one of the processes of building the regress tree. It is an iterative process that splits the data into partitions or branches, and then continues splitting each partition into smaller groups as the method moves up each
Supervised Learning branch. Initially, all records in the training set are grouped into the same partition. The algorithm then begins allocating the data into the first two partitions or branches, using every possible binary split on every field. The algorithm selects the split that minimizes the sum of the squared deviations from the mean in the two separate partitions. This splitting rule is then applied to each of the new branches. This process continues until each node reaches a user-specified minimum node size and becomes a terminal node. (If the sum of squared deviations from the mean in a node is zero, then that node is considered a terminal node even if it has not reached the minimum size.) The obtained tree is pruned to minimize the sum of: 1. The output variable variance in the validation data, taken one terminal node at a time. 2. The product of the cost complexity factor and the number of terminal nodes. If the cost complexity factor is specified as zero, then pruning is simply finding the tree that performs best on validation data in terms of total terminal node variance. Larger values of the cost complexity factor result in smaller trees. Pruning is performed on a last-in first-out basis, meaning the last grown node is the first to be subject to elimination. 6.4 Advantages and Limitations

99%	MATCHING BLOCK 71/189	W

The purpose of the analysis conducted by any classification or regression tree is to create a set of if-else conditions that allow for the accurate prediction or classification of a case. ● The Results are Simplistic The interpretation of results summarized in classification or regression trees is usually fairly simple. The simplicity of results helps in the following ways. It allows for the rapid classification of new observations. That's because it is much simpler to evaluate just one or two logical conditions than to compute scores using complex nonlinear equations for each group. It can often result in a simpler model which explains why the observations are either classified or predicted in a certain way. For instance, business problems are much easier to explain with if-then statements than with complex nonlinear equations. ● Classification and Regression Trees are Nonparametric

and

Nonlinear The results from classification and regression trees can be summarized in

Supervised Learning

simplistic if-then conditions. This negates the need for the following implicit assumptions. ➔ The predictor variables and the dependent variable are linear. ➔ The predictor variables and the dependent variable follow some specific nonlinear link function. ➔ The predictor variables and the dependent variable are monotonic. Since there is no need for such implicit assumptions, classification and regression tree methods are well suited to data mining. This is because there is very little knowledge or assumptions that can be made beforehand about how the different variables are related. As a result, classification and regression trees can actually reveal relationships between these variables that would not have been possible using other techniques. ● Classification and Regression Trees Implicitly Perform Feature Selection Feature selection or variable screening is an important part of analytics. When we use decision trees, the top few nodes on which the tree is split are the most important variables within the set. As a result, feature selection gets performed automatically and we don't need to do it again. Limitations of Classification and Regression Trees Classification and regression tree tutorials, as well as classification and regression tree ppts, exist in abundance. This is a testament to the popularity of these decision trees and how frequently they are used. However, these decision trees are not without their disadvantages. There are many classification and regression trees examples where the use of a decision tree has not led to the optimal result. Here are some of the limitations of classification and regression trees. ● Overfitting Overfitting occurs when the tree takes into account a lot of noise that exists in the data and comes up with an inaccurate result. ● High variance In this case, a small variance in the data can lead to a very high variance in the prediction, thereby affecting the stability of the outcome. ● Low bias A decision tree that is very complex usually has a low bias. This makes it very

Supervised Learning

difficult for the model to incorporate any new data. 6.5 CART Algorithm Classification And Regression Trees (CART)

algorithm is a classification algorithm for building a decision tree based on Gini's impurity index as a splitting criterion. CART is a binary tree built by splitting a node into two child nodes repeatedly. The algorithm works repeatedly in three steps: 1. Find each feature's best split. For each feature with K different values there exist K-1 possible splits. Find the split, which maximizes the splitting criterion. The resulting set of splits contains best splits (one for each feature). 2. Find the node's best split. Among the best splits from Step i find the one, which maximizes the splitting criterion. 3. Split the node using the best node split from Step ii and repeat from Step i until stopping criterion is satisfied. As splitting criterion we used Gini's impurity index, which is defined for node t as: (t) = ∑ , (|) (|) (|) where C(i|j) is cost of misclassifying a class j case as a class i case (in our case C(i|j) = 1, if i ≠ j and C(i|j) = 0 if i = j), p(i|t) (p(j|t) respectively) is probability of case in class i (j) given that falls into node t. The Gini impurity criterion is type of decrease of impurity, which is defined as: Δ i(s, t) = i(t) – p L i(t L) – p R i(t R) where Δi(s, t) is decrease of impurity at node t with split s, p L (p R) are probabilities of sending case to the left (right) child node t L (t R) and i(t L) (i(t R)) is Gini impurity measure for left (right) child node. In order to enhance generalization of the decision tree we used pruning with a combination of cross-validation error rate estimation. The algorithm for pruning works as follows: 1. Split randomly training data into 10 folds. 2. Select pruning level of tree (level 0 equals to full decision tree). 3. Use 9 folds for creation of 9 new pruned trees and estimate error on the last 10th fold. 4. Repeat from Step ii until all pruning levels are used. 5. Find the smallest error and use the pruning level assigned to it. 6. Until pruning level is reached, remove all terminal nodes in the lowest tree level
Supervised Learning and assign decision class to parent node. Decision value is equal to class with higher number of cases covered by node.

97%	MATCHING BLOCK 75/189	W

The Classification and regression tree (CART) methodology is one of the oldest and most fundamental algorithms. It is used to predict outcomes based on certain predictor variables. They are excellent for data mining tasks because they require very little data pre-processing. Decision tree models are easy to understand and implement which gives them a strong advantage when compared to other analytical models. 6.6 Regression Analysis Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as temperature, age, salary, price, etc. We can understand the concept of regression analysis using the below example: Suppose there is a marketing company A, who does various advertisements every year and gets sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:

Advertisement

98%	MATCHING BLOCK 76/189	W

Sales $90 $1000 $120 $1300 $150 $1800 $100 $1200 $130 $1380 $200 ?? Now, the company wants to do the advertisement of $200 in the year 2021 and wants to know the prediction about the sales for this year. So to solve such types of prediction problems in machine learning, we need regression analysis. Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between

Supervised Learning

93%	MATCHING BLOCK 77/189	W

variables. In Regression, we plot a graph between the variables which best fits the given data points, using this plot, the machine learning model can make predictions about the data. In simple words, regression shows a line or curve that passes through all the

data points on a

87%	MATCHING BLOCK 78/189	W

target-predictor graph in such a way that the vertical distance between the data points and the regression line is minimum. The distance between data points and line tells whether a model has captured a strong relationship or not. Some examples of regression can be as prediction of rain using temperature and other factors; determining Market trends; prediction of road accidents due to rash driving. Some basic terminologies related to the Regression Analysis: ● Dependent Variable: The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called target variable. ● Independent Variable: The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variables, also called as a predictor. ● Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided. ● Multicollinearity: If the independent variables are highly correlated with each other than other variables, then such

a

100%	MATCHING BLOCK 79/189	W

condition is called Multicollinearity. It should not be present in the dataset, because it creates

a

97% MATCHING BLOCK 80/189 W

problem while ranking the most affecting variable. ● Underfitting and Overfitting: If our algorithm works well with the training dataset but not well with

the test dataset, then such a

100% MATCHING BLOCK 81/189 W

problem is called Overfitting. And if our algorithm does not perform well even with

a training dataset, then such a

87% MATCHING BLOCK 82/189 W

problem is called underfitting. As mentioned above, Regression analysis helps in the prediction of a continuous variable. There are various scenarios in the real world where we need some future predictions such as weather conditions, sales

predictions, marketing trends, etc. For such a

95% MATCHING BLOCK 83/189 W

case we need some technology which can make predictions more accurately. So we need Regression analysis which is a statistical method and used in machine learning and data science. Below are some other reasons for using Regression

Supervised Learning

92% MATCHING BLOCK 84/189 W

analysis: ● Regression estimates the relationship between the target and the independent variable. ● It is used to find the trends in data. ● It helps to predict real/continuous values. ● By performing the regression, we can confidently determine the most important factor, the least important factor, and how each factor is affecting the other factors. 6.7 Types of Regression There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Some important types of regression are listed below: ● Linear Regression ● Logistic Regression ● Polynomial Regression ● Support Vector Regression ● Decision Tree Regression ● Random Forest Regression ● Ridge Regression ● Lasso Regression

Supervised Learning
Figure 6.3 Types of

96% MATCHING BLOCK 85/189 W

Regression 6.7.1 Linear Regression ● Linear regression is a statistical regression method which is used for predictive analysis. ● It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables. ● It is used for solving the regression problem in machine learning. ● Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression. ● If there is only one input variable (x), then such linear regression is called simple linear regression. And if there is more than one input variable, then such linear regression is called multiple linear regression. ● The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of the year of experience. Below is the mathematical equation for Linear regression: Y= aX+b Here, Y = dependent variables (target variables) X= Independent variables (predictor variables) a and b are the linear coefficients Some popular applications of linear regression are: Supervised Learning ● Analyzing trends and sales estimates ● Salary forecasting ● Real estate prediction ● Arriving at ETAs in traffic

100%	MATCHING BLOCK 86/189	W

Linear Regression Line A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

a)

97%	MATCHING BLOCK 87/189	W

Positive Linear Relationship: If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship. b) Negative Linear Relationship: If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship. When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error. The different values for weights or the coefficient of lines (a0, a1) gives a different line of regression, so we need to calculate the best values for a0 and a1 to find the best fit line, so to calculate this we use cost function. Cost function- ● The different values for weights or coefficient of lines (a0, a1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. ● Cost function optimizes the regression coefficients or weights. It measures how

Supervised Learning

98%	MATCHING BLOCK 88/189	W

a linear regression model is performing. ● We can use the cost function to find the accuracy of the mapping function, which maps the input variable to the output variable. This mapping function is also known as Hypothesis function. 6.7.2

96%	MATCHING BLOCK 89/189	W

Logistic Regression ● Logistic regression is another supervised learning algorithm which is used to solve the classification problems. In classification problems, we have dependent variables in a binary or discrete format such as 0 or 1. ● Logistic regression algorithm works with categorical

variables

95%	MATCHING BLOCK 90/189	W

such as 0 or 1, Yes or No, True or False, Spam or not spam, etc. ● It is a predictive analysis algorithm which works on the concept of probability. ● Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used. ● Logistic regression uses sigmoid function or logistic function which is a complex cost function. This sigmoid function is used to model the data in logistic regression. The function can be represented as: f(x) = Output between the 0 and 1 value x = input to the function e = base of natural logarithm When we provide the input values (data) to the function, it gives the S-curve as follows: ● It uses the concept of threshold levels, values above the threshold level are rounded up to 1, and values below the threshold level are rounded up to 0. Supervised Learning There are three types of logistic regression: ● Binary (0/1, pass/fail) ● Multi (cats, dogs, lions) ● Ordinal (low, medium, high) 6.8

Summary ● Classification is the process of finding or discovering a model or function which helps in separating the data into multiple categorical classes i.e. discrete values. ● Regression is the process of finding a model or function for distinguishing the data into continuous real values instead of using classes or discrete values. ●

100%	MATCHING BLOCK 91/189	W

A classification tree is an algorithm where the target variable is fixed or categorical. ●

100%	MATCHING BLOCK 92/189	W

A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value. ●

100% MATCHING BLOCK 93/189 W

The purpose of the analysis conducted by any classification or regression tree is to create a set of if-else conditions that allow for the accurate prediction or classification of a case. ●

CART is a binary tree built by splitting a node into two child nodes repeatedly. 6.9 Key Terms ●

100% MATCHING BLOCK 94/189 W

Overfitting: Overfitting occurs when the tree takes into account a lot of noise that exists in the data and comes up with an inaccurate result. ●

Binary Recursive Partitioning: It is one of the processes of building the regress tree. ● Classification Tree Analysis (CTA): It is a type of machine learning algorithm used for classifying remotely sensed and ancillary data in support of land cover mapping and analysis. ●

100% MATCHING BLOCK 95/189 W

Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. ●

Mapping function: It maps the input variable to the output variable.
Supervised Learning 6.10 Check Your Progress Short-Answer Type Q1) _______ identifies the distribution movement depending on the historical data. Q2) Predicting the selling prices of a residential house is an example of _______ trees. Q3) In regression trees, pruning is performed on a last-in first-out basis. (True/ False?) Long-Answer Type Q1) List the advantages and limitations of classification and regression trees. Q2) Explain the CART algorithm with an example. Q3) Give at least four reasons for using regression analysis. References Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997. An Introduction to Machine Learning, 2 nd edition, Miroslav Kubat, Springer, 2017.
Supervised Learning Unit 7 – Bayesian and Nearest- Neighbor Classifiers Structure 7.0 Introduction 7.1 Unit Objectives 7.2 Bayesian Learning: Introduction 7.3 Bayes' Theorem 7.4 Naive Bayes' Algorithm 7.5 Instance-based Learning: Introduction 7.6 The K-Nearest-Neighbor Algorithm 7.7 Weighted K-Nearest-Neighbor 7.8 Performance Evaluation 7.8.1 Basic Performance Criteria 7.8.2 Precision and Recall 7.9 Summary 7.10 Key Terms 7.11 Check Your Progress 7.0 Introduction In the previous units, we have already demonstrated the logical models (decision trees). In this unit, we will discuss the probabilistic models of machine learning.

100% MATCHING BLOCK 96/189 W

The probabilistic models use the idea of probability to classify new entities. Probabilistic models see features and target variables as random variables. The process of modelling represents and manipulates the level of uncertainty with respect to these variables.

These models include algorithms that follow the probability approach. Such algorithms include Bayes' theorem and based classifiers, Nearest-neighbor and based classifiers. 7.1 Unit Objectives After completing this unit, the reader will be able to: ● Describe Bayes' theorem with examples. ● Discuss Naïve Bayes' Algorithm. ● Illustrate K-Nearest- Neighbor Algorithm.

Supervised Learning ● Gain knowledge about the performance evaluation in machine learning. 7.2 Bayesian Learning: Introduction Bayesian reasoning provides a probabilistic approach to inference. It is based on the assumption that the quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data. It is important to machine learning because it provides a quantitative approach to weighing the evidence supporting alternative hypotheses. Bayesian reasoning provides the basis for learning algorithms that directly manipulate probabilities, as well as a framework for analyzing the operation of other algorithms that do not explicitly manipulate probabilities. Bayesian learning methods are relevant to our study of machine learning for two different reasons. 1. Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems. 2. Bayesian methods provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities. We also use a Bayesian analysis to justify a key design choice in neural network learning algorithms: choosing to minimize the sum of squared errors when searching the space of possible neural networks. We also derive an alternative error function, cross entropy, that is more appropriate than the sum of squared errors when learning target functions that predict probabilities. We use a Bayesian perspective to analyze the inductive bias of decision tree learning algorithms that favor short decision trees and examine the closely related Minimum Description Length principle. A basic familiarity with Bayesian methods is important to understanding and characterizing the operation of many algorithms in machine learning. Features of Bayesian learning methods include: ● Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example. ● Prior knowledge can be combined with observed data to determine the final
Supervised Learning probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting- a) A prior probability for each candidate hypothesis b) A probability distribution over observed data for each possible hypothesis. ● Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g. hypotheses such as "this regular student has a 93% chance of scoring good marks"). ● New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities. ● Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured. Practical difficulties in Bayesian Methods 1. Bayesian methods typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions. 2. The significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses). In certain specialized situations, this computational cost can be significantly reduced.
The Bayesian classifier is an algorithm for classifying multiclass datasets. This is based on the Bayes' theorem in probability theory. Before proceeding, let's memorize
the concept of conditional probability. Conditional probability The probability of the occurrence of an event A given that an event B has already occurred is called
the conditional probability of A given B and is denoted by $P(A|B)$. We have
Independent
events 1. Two events
A and B are said to be independent if
Supervised Learning
$P(A \cap B) = P(A)P(B)$ 2. Three events A, B, C are said to be pairwise independent if $P(B \cap C) = P(B)P(C)$ $P(C \cap A) = P(C)$
$P(A)$
P (
$A \cap B) = P(A)P(B)$ 3. Three events A, B, C are said to be mutually independent if $P(B \cap C) = P(B)P(C)$ ....(7.1) $P(C \cap A) =$
$P(C)$
$P(A)$ ....(7.2)
$P(A \cap B) = P(A)P(B)$ ....(7.3) $P(A \cap B \cap C) = P(A)$
$P(B)$
P (
C) ....(7.4) 4.
In general, a family of k events $A_1, A_2, \ldots, A_k$ is said to be mutually independent if for any subfamily consisting of $A_{i1}, \ldots A_{im}$ we have $P(A_{i1} \cap \ldots \cap A_{im}) = P(A_{i1}) \ldots P($
$A_{im})$.
Consider events and respective probabilities as shown in figure 7.1. It can be seen that, in this case, the conditions Eqs.(7.1)−(7.3) are satisfied, but Eq.(7.4) is not satisfied. But if the probabilities are as in figure 7.2, then Eq.(7.4) is satisfied but all the conditions in Eqs.(7.1)−(7.2) are not satisfied. Figure 6.1: Events A, B, C which are not mutually independent Figure 6.2: Events A, B, C which are not mutually independent.
Supervised Learning 7.3 Bayes' Theorem Let A and B
be
any two events in a random experiment. If $P(A) \neq 0$, then 1.

The importance of the result is that it helps us to "invert" conditional probabilities, that is, to express the conditional probability P (A|B) in terms of the conditional probability P (B|A). 2. The following terminology is used in this context: ● A is called the proposition and B is called the evidence. ● P (A) is called the prior probability of proposition and P (B) is called the prior probability of evidence. ●
P (
A|B) is called the posterior probability of A given B. ● P (B|A) is called the likelihood of B given A.
Generalization: Let the sample space be divided into disjoint events B 1 , B 2 , . . . , B n and A be any event. Then we have
Example-1
Consider a set of patients coming for treatment in a certain clinic. Let A denote the event that a "Patient has liver disease" and B the event that a "Patient is an alcoholic." It is known from experience that 10% of the patients entering the clinic have liver disease and 5% of the patients are alcoholics. Also, among those patients diagnosed with liver disease, 7% are alcoholics. Given that a patient is alcoholic, what is the probability that he will have liver disease? Using the notations of probability, we have
Supervised Learning Example-2
Three factories A, B, C of an electric bulb manufacturing company produce respectively 35%. 35% and 30% of the total output. Approximately 1.5%, 1% and 2% of the bulbs produced by these factories are known to be defective. If a randomly selected bulb manufactured by the company was found to be defective, what is the probability that the bulb was manufactured in factory A? Let A, B, C denote the events that a randomly selected bulb was manufactured in factory A, B, C respectively. Let D denote the event that a bulb is defective. We have the following data: P (A) = 0.35, P (B) = 0.35, P (C) = 0.30 P (D|A) = 0.015, P (D|B) = 0.010, P (D|C) = 0.020 We are required to find P (A|D). By the generalisation of the Bayes' theorem we have: 7.4
Naive Bayes' Algorithm One highly practical Bayesian learning method is the naive Bayes learner, often called the
naive Bayes classifier. The Naive Bayes algorithm is based on the following assumptions: ● All the features are independent and are unrelated to each other.
Presence or absence of a feature does not
infl uence
the presence or absence of any other feature. ●
The data has class-conditional independence, which means that events are independent so long as they are conditioned on the same class value. These assumptions are, in general, true in many real world problems. It is because of these assumptions, the algorithm is called a naive algorithm. Suppose we have a training data set consisting of N examples having n features. Let the features
be named as (F 1 , . . . , F n).
A feature vector is of the form (f 1 , f 2 , . . . , f n). Associated with each
example, there is a certain class label. Let the set of class labels be {c 1 , c 2 , . . . ,
c p }.
Suppose we are given a test instance having the feature vector
Supervised Learning
X = (x 1 , x 2 , . . . , x n).
We are required to determine the most appropriate class label that should be assigned to the test instance. For this purpose we compute the following conditional probabilities P (c 1 |X), P (c 2 |X), . . . , P (c p |X) .... (7.5) and choose the maximum among them. Let the maximum probability be P (c i |X). Then, we choose c i as the most appropriate class label for the training instance having X as the feature vector. The direct computation of the probabilities given in Eq.(7.5) are difficult for a number of reasons. The Bayes' theorem can
be
applied to obtain a simpler method. This is explained below. Using Bayes' theorem, we have: ....(7.6)
Since, by assumption, the data has class-conditional independence, we note that the events x 1 |c k , x 2 |c k , ⋯,
x
n |c k are independent (because they are all conditioned on the same class label c k). Hence we have P (X|c k) = P ((x 1 , x 2 , . . . , x n)|c k) = P (x 1 |c k) P (x 2 |c k)⋯
P (x n |c k) Substituting these values in Eq,(7.6) we get
Since the denominator P (X) is independent of the class labels, we have
P (c k |X) ∝ P (x 1 |c k) P (x 2 |c k)⋯ P (x n |c k) P (c
k) So it is enough to find the maximum among the following values: P (x 1 |c k) P (x 2 |
c k)⋯ P (x n |c k) P (c k), k = 1, . . . , p Note:
The various probabilities in the above expression are computed as follows:
Supervised Learning Table 7.1 Naive Bayes' Algorithm
Let there be a training data set having n features F 1 , . . . , F n . Let f 1 denote an arbitrary value of F 1 , f 2 of F 2 , and so on. Let the set of class labels be {c 1 , c 2 , . . . ,
c p }.
Let there be given a test instance having the feature vector X = (x 1 , x 2 , . . . , x n). We are required to determine the most appropriate class label that should be assigned to the test instance. Step 1: Compute the probabilities P (
c k)

for k = 1, . . . , p. Step 2: Form a table showing the conditional probabilities
P (f 1 |c k), P (f 2 |c k), . . . , P (f n |c k)
for all values of f 1 , f 2 , . . . , f n and for k = 1, . . . , p. Step 3: Compute the products q k = P (x 1 |c k) P (x 2 |c k)⋯ P (x n |c k) P (c k) for k = 1, . . . , p. Step 4: Find j such
that q j = max {q 1 , q 2 , . . . , q p }. Step 5: Assign the class label c j
to the test instance X. In Table 7.1, Steps 1 and 2 constitute the learning phase of the algorithm. The remaining steps constitute the testing phase. For testing purposes, only the table of probabilities is required; the original data set is not required. Example: Consider a training data set consisting of the fauna of the world. Each unit has three features named “Swim”, “Fly” and “Crawl”. Let the possible values of these features be as follows: Swim Fast, Slow, No
Fly Long, Short, Rarely, No Crawl
Yes, No For simplicity, each unit is classified as “Animal”, “Bird” or “Fish”. Let the training data set be as in Table 7.2. Use naive Bayes algorithm to classify a particular species if its features are (Slow, Rarely, No)?
Supervised Learning Table 7.2 Sample data set for Naive Bayes Algorithm S.
No. Swim Fly Crawl Class 1
Fast No No Fish 2 Fast No Yes Animal 3 Slow No No Animal 4 Fast No No Animal 5 No Short No Bird 6 No Short No Bird 7 No Rarely No Animal 8 Slow No Yes Animal 9 Slow No No Fish 10 Slow No Yes Fish 11 No Long No
Bird 12 Fast No No Bird
In this example, the features are F 1 = “Swim”, F 2 = “Fly”, F 3 = “Crawl”. The class labels are c 1 = “Animal”, c 2 = “ Bird”, c 3 = “Fish”. The test instance is (Slow, Rarely, No) and so we have: x 1 = “Slow”, x 2 = “Rarely”, x 3 = “No”. We construct the frequency table shown in Table 7.3 which summarises the data. (It may be noted that the construction of the frequency table is not part of the algorithm.)
Supervised Learning
Table 7.3 Frequency table for the data in Table 7.2 Step 1: We compute
the following probabilities. Step 2: We construct the following table of conditional probabilities:
Table 7.4 Table of the conditional probabilities P (f i |c k) Note: The conditional probabilities are calculated as follows:
Step 3: We now calculate the following numbers: q1 =
P (x 1 |c 1)
P (x 2 |c 1) P (x 3 |c 1)
P(c 1) = (2/5) × (1/5) × (3/5) × (5/12) = 0.02 q2 = P (x 1 |c 2) P (x 2 |c 2) P (x 3 |c 2) P (c 2) = (0/4) × (0/4) × (3/4) × (4/12) = 0
Supervised Learning q3 = P (x 1 |c 3)
P (x 2 |c 3) P (x 3 |c 3) P (c 3) = (2/3) × (0/3) × (3/3) × (3/12) = 0 Step 4:
Now max {q1, q2, q3} = 0.05. Step 5: The maximum is q 1 and it corresponds to the class label c 1 = “ Animal”. So we assign the class label Animal to the test instance (Slow, Rarely, No). The naive Bayes algorithm can be applied to a data set only if the features are categorical. This is so because the various probabilities are computed using the various frequencies and the frequencies can be counted only if each feature has a limited set of values. If a feature is numeric, it has to be discretized before applying the algorithm. The discretization is affected by putting the numeric values into categories known as bins. Because of this discretization is also known as binning. This is ideal when there are large amounts of data. There are several different ways to discretize a numeric feature. ● If there are natural categories or cut points in the distribution of values, use these cut points to create the bins. For example, let the data consist of records of times when certain activities were carried out. The categories, or bins, may be created as in figure 7.3.
Figure 7.3: Example for
Discretization of numeric data ● If there are no obvious cut points, we may discretize the feature using
Supervised Learning
quantiles. We may divide the data into three bins with tertiles, four bins with quartiles, or five bins with quintiles, etc. Maximum likelihood estimation (ML estimation) To develop a Bayesian classifier, we need the probabilities P (x|c k) for the class labels c 1 , . . . , c k . These probabilities are estimated from the given data. There is
a
need to know whether the sample is truly random so that the computed probabilities are good approximations to true probabilities. If they are good approximations of true probabilities, then there would be an underlying probability distribution. Suppose we have reasons to believe that the underlying distribution has a particular form, say binomial, Poisson or normal. These forms are defined by probability functions or probability density functions. There are parameters which define these functions, and these parameters are to be estimated to test whether a given data follows some particular distribution. Maximum likelihood estimation is
a
particular method to estimate the parameters of a probability distribution. Maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model, given observations. MLE attempts to find the parameter values that maximize the likelihood function, given the observations. The resulting estimate is called a maximum likelihood estimate, which is also abbreviated as MLE. Suppose we have a random sample X = {x 1 , . . . , x n } taken from a probability distribution having
the probability mass function or probability density function p(x|θ) where x denotes a value of the

random variable and θ denotes the set of parameters that appear in the function. The likelihood of sample X is a function of the parameter θ and is defined as l(θ) = p (x 1 |θ) p (x 2 |θ) . . . p (x n |θ) In maximum likelihood estimation, we find the value of θ that makes the value of the likelihood function maximum. For computation convenience, we define the log likelihood function as the logarithm of the likelihood function: L(θ) = log l(θ) = log p(x 1 |θ) + log p(x 2 |θ) + ⋯ + log p(
x n |θ)
A value of θ that maximizes L(θ) will also maximise l(θ) and vice-versa. Hence, in maximum likelihood estimation, we find θ that maximizes the log likelihood function. Sometimes the maximum likelihood estimate of θ is denoted by ? .
Supervised Learning 7.5 Instance-based Learning: Introduction In contrast to learning methods that construct a general, explicit description of the target function when training examples are provided, instance-based learning methods simply store the training examples. Generalizing beyond these examples is postponed until a new instance must be classified. Each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a target function value for the new instance. Instance- based learning includes nearest neighbor methods that assume instances can be represented as points in a Euclidean space. It also includes case-based reasoning methods that use more complex, symbolic representations for instances. Instance-based methods are sometimes referred to as "lazy" learning methods because they delay processing until a new instance must be classified. A key advantage of this kind of delayed, or lazy, learning is that instead of estimating the target function once for the entire instance space, these methods can estimate it locally and differently for each new instance to be classified. Instance-based learning methods such as nearest neighbor and locally weighted regression are conceptually straightforward approaches to approximating real-valued or discrete-valued target functions. Learning in these algorithms consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance. One key difference between these approaches and the methods discussed in other chapters is that instance-based approaches can construct a different approximation to the target function for each distinct query instance that must be classified. In fact, many techniques construct only a local approximation to the target function that applies in the neighborhood of the new query instance, and never construct an approximation designed to perform well over the entire instance space. This has significant advantages when the target function is very complex, but can still be described by a collection of less complex local approximations. Instance-based methods can also use more complex, symbolic representations for instances. In case-based learning, instances are represented in this fashion and the process for identifying "neighboring" instances is elaborated accordingly. Case-based reasoning has been applied to tasks such as storing and reusing past experience at a help desk, reasoning about legal cases by referring to previous cases, and solving
Supervised Learning complex scheduling problems by reusing relevant portions of previously solved problems. Disadvantage of instance-based approaches ● The cost of classifying new instances can be high. This is due to the fact that nearly all computation takes place at classification time rather than when the training examples are first encountered. Therefore, techniques for efficiently indexing training examples are a significant practical issue in reducing the computation required at query time. ● The instance-based approaches typically consider all attributes of the instances when attempting to retrieve similar training examples from memory. If the target concept depends on only a few of the many available attributes, then the instances that are truly most "similar" may well be a large distance apart. 7.6 The K-Nearest-Neighbor Algorithm The most basic instance-based method is the k-Nearest Neighbor algorithm.
This algorithm assumes all instances correspond to points in the n-dimensional space ℜ n . The nearest neighbors of an instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance x
be described by the feature vector where
a r (x) denotes the value of the r th attribute of instance x. Then the distance between two instances x i and x j is defined to be d (x
i ,
x
j) , where In nearest-neighbor learning the target function may be either discrete-valued or real-valued. Let us first consider learning discrete-valued target functions of the form f :ℜ n →V, where V is the finite set {v 1 , . . . v S }.
The k-Nearest Neighbor algorithm for approximating a discrete-valued target function is given in Table 7.5. As shown there, the value f (x q) returned by this algorithm as its estimate of f (x q)
is just the most common value of f among the k training examples nearest to x
q . If we choose k = 1, then the 1-Nearest Neighbor algorithm assigns to ? (x q)
the value f (x i) where x i is the training instance nearest to x
q .
For larger values of k, the algorithm assigns the most common value among the k nearest training examples.
Supervised Learning Table 7.5 The k-Nearest Neighbor algorithm for
approximating a discrete-valued function f :ℜ n →V
Training algorithm: ● For each training example (x, f (x)) , add the example to the list training_examples Classification algorithm: ● Given
a query
instance x q to be classified, ➔ Let x 1 . . . x k denote the k instances from training_examples that are nearest to x q ➔ Return where δ(a, b) = 1 if a = b and where δ (a, b) = 0 otherwise.
The

k-Nearest Neighbor algorithm is easily adapted to approximating continuous- valued target functions. To accomplish this, we have the algorithm calculate the mean value of
the
k nearest training examples rather than calculate their most common value. More precisely, to approximate a real-valued target function f :ℜ n →ℜ we replace the final line of the above algorithm by
the line ●
K-Nearest Neighbour (K-NN) is one of the simplest Machine Learning algorithms based on Supervised Learning
technique. ● The
K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories. ●
The K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm. ● The K-NN
algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. ● K-NN is a non-parametric algorithm,
which means it does not make any
Supervised Learning assumption on underlying data. ● It is also called a
lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the
dataset. ● The K-NN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data. ● Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know whether it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.
In k-NN classification, the output is a class membership. An object
is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (
k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest
neighbors. Need for K-NN algorithm Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x 1 , so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider figure 7.4.
Supervised Learning Figure 7.4 Example for K-NN algorithm The K-NN
working can be explained on the basis of the below algorithm: ● Step-1: Select the number K of the neighbors. ● Step-2: Calculate the Euclidean distance of K number of neighbors. ● Step-3: Take the K nearest neighbors as per the calculated Euclidean distance. ● Step-4: Among these k neighbors, count the number of the data points in each category. ● Step-5: Assign the new data points to that category for which the number of the neighbor is maximum. ● Step-6: Our model is ready. Euclidean Distance
In a plane,
the geometric distance between two points, x = (x 1 ,x 2) and y = (y 1 , y 2), is obtained with the help of the
pythagorean theorem . This formula is easy to generalize to a domain with n continuous attributes where the Euclidean distance between x = (x 1 ,......., x n) and y = (y 1 ,......., y n) is defined as follows: Advantages of KNN Algorithm: ● It is simple to implement. ● It is robust to the noisy training data ● It can be more effective if the training data is large.
Supervised Learning Disadvantages of KNN Algorithm: ● Always needs to determine the value of K which may be complex some time. ● The computation cost is high because of calculating the distance between the data points for all the training samples. 7.7 Weighted K-Nearest-Neighbor Weighted KNN is a modified version of k-nearest neighbors. One of the many issues that affect the performance of the kNN algorithm is the choice of the hyperparameter k. If k is too small, the algorithm would be more sensitive to outliers. If k is too large, then the neighborhood may include too many points from other classes. Another issue is the approach to combining the class labels. The simplest method is to take the majority vote, but this can be a problem if the nearest neighbors vary widely in their distance and the closest neighbors more reliably indicate the class of the object. In figure 7.5, the task is to determine the class of object x, represented by a black dot. Since three of the nearest neighbors are negative, and only two are positive, the 5-NN classifier will label x as negative. And yet, something seems to be wrong, here: the three negative neighbors are quite distant from x; as such, they may not deserve to have the same interpretation as the two positive examples in the object's immediate vicinity. Figure 7.5 The testimony of the two "positive" neighbors should outweigh that of the three more distant "negative" neighbors Situations of this kind motivate the introduction of weighted voting, in which the weight of each of the nearest neighbors is made proportional to its distance from x: the closer the neighbor, the greater its impact.

Supervised Learning Let us denote the weights as w 1 ,......., w k . The weighted k-NN classifier sums up the weights of those neighbors that recommend the positive class (let the result be denoted by Σ +) and then sums up the weights of those neighbors that support the negative class (Σ -). The final verdict is based on which is higher: if Σ + &lt; Σ - , then x is deemed positive, otherwise it is deemed negative. Generalization to the case with n &lt; 2 classes is straightforward. For the sake of illustration, suppose the positive label is found in neighbors with weights 0.6 and 0.7, respectively, and the negative label is found in neighbors with weights 0.1; 0.2; and 0.3. The weighted k-NN classifier will choose the positive class because the combined weight of the positive neighbors, Σ + = 0.6 + 0.7 = 1:3, is greater than that of the negative neighbors, Σ - = 0.1 + 0.2 + 0.3 = 0.6. Just as in figure 7.5, the more frequent negative label is outvoted by the positive neighbors because the latter are closer to x. Let us introduce a simple formula to calculate the weights. Suppose the k neighbors are ordered according to their distances, d 1 , ..., d k , from x so that d 1 is the smallest distance and d k is the greatest distance. The weight of the i-th closest neighbor is calculated as follows: Table 7.6 Weighted k-Nearest Neighbor Algorithm

Supervised Learning ● Let L = {(x i , y i), i = 1, . . . ,n} be a training set of observations x i with given class y i and let x be a new observation (query point), whose class label y has to be predicted. ● Compute d(x i , x) for i = 1, . . . , n, the distance between the query point and every other point in the training set. ● Select D′ ⊆ D, the set of k nearest training data points to the query points ● Predict the class of the query point, using distance-weighted voting. The v represents the class labels. Use the following formula: 7.8 Performance Evaluation The performance evaluation in machine learning is a fairly straightforward matter. All it takes is to apply the induced classifier to a set of examples whose classes are known, and then count the number of errors the classifier has made. In reality, things are not as simple. Error rate rarely paints the whole picture, and there are situations in which it can even be misleading. This is why the conscientious engineer wants to be acquainted with other criteria to assess the classifiers' performance. This knowledge will enable her to choose the one that is best in capturing the behavioral aspects of interest. So much for the evaluation of classifiers. Somewhat different is the question of how to compare the suitability of alternative machine-learning techniques for induction in a given domain. Dividing the set of pre-classified examples randomly into two subsets (one for induction, the other for testing) may not be the best thing to do, especially if the training set is small; random division may then result in subsets that do not represent the given domain properly. To obtain more reliable results, repeated random runs are necessary. 7.8.1 Basic Performance Criteria When testing a classifier on an example whose real class is known, we can encounter only the following four different outcomes: 1. the example is positive and the classifier correctly recognizes it as such (true

Supervised Learning positive); 2. the example is negative and the classifier correctly recognizes it as such (true negative); 3. the example is positive, but the classifier labels it as negative (false negative); 4. the example is negative, but the classifier labels it as positive (false positive). When applying the classifier to an entire set of examples (whose real classes are known), each of these four outcomes will occur a different number of times—and these numbers are then employed in the performance criteria defined below. The symbols representing the four outcomes are summarized in Table 7.7. Specifically, ● N TP is the number of true positives, ● N TN is the number of true negatives, ● N FP is the number of false positives ● N FN is the number of false negatives. In the entire example set, T, only these four categories are possible; therefore, the size of the set, |T|, equals the sum, |T| = N FP + N FN + N TP + N TN . Table 7.7 The basic quantities used in the definitions of performance criteria Note that the number of correct classifications is the number of true positives plus the number of true negatives, N TP + N TN ; and the number of errors is the number of false positives plus the number of false negatives, N FP + N FN . Error Rate and Classification Accuracy A classifier's error rate, E, is the frequency of errors made by the classifier over a given set of examples. It is calculated by dividing the number of errors, N FP + N FN , by the total number of examples, N FP + N FN + N TP + N TN .

Supervised Learning Classification Accuracy (Acc): It is the frequency of correct classifications made by the classifier over a given set of examples. Classification accuracy is calculated by dividing the number of correct classifications, N TP + N TN , by the total number of examples, N FP + N FN + N TP + N TN . Note that Acc = 1 - E. 7.8.2 Precision and Recall In some applications, negative examples outnumber positive ones by a wide margin. When this happens, error rate offers a misleading picture of classification performance. To see why, just consider the case where only 2% of all examples are positive, and all the remaining 98% are negative. A "classifier" that returns the negative class for any example in the set will be correct 98% of the time—which may look like a remarkable feat. And yet, the reader will agree, a classifier that never recognizes a positive example is useless. In some domains, error rate and classification accuracy will hardly tell us anything reasonable about the classifier's practical utility. Rather than averaging the performance over both (or all) classes, we need criteria that focus on a class which, while important, is represented by only a few examples. Precision Precision is the probability that the classifier is right when labeling an example as positive. By this we mean the percentage of true positives, N TP , among all examples that the classifier has labeled as positive: N TP + N FP . The value is thus obtained by the following formula: Recall By this we mean the probability that a positive example will be correctly recognized as such (by the classifier). The value is therefore obtained by dividing the number of true positives, N TP , by the number of positives in the given set: N TP + N FN . Here is the formula:

Supervised Learning Note that the last two formulas differ only in the denominator. Whereas precision is the frequency of true positives among all examples deemed positive by the classifier, recall is the frequency of the same true positives among all positive examples in the set. In some domains, precision is more important than recall. For instance, when you purchase something from an e-commerce web site, their recommender system often reacts with a message to the effect that, “Customers who have bought X buy also Y.” The obvious intention is to cajole you into buying Y as well. Recommender systems are sometimes created with the help of machine learning techniques applied to the company’s historical data. When evaluating their performance, the engineer wants to achieve high precision. The customers better be happy about the recommended merchandise, or else they will ignore the recommendations in the future. The value of recall is here unimportant. The list offered on the web site has to be of limited size, and so it does not matter much that the system identifies only a small percentage of all items that the customers may like. In other domains, by contrast, recall is more important. This is often the case in medical diagnosis. A patient suffering from X, and properly diagnosed as such, represents a true positive. A patient suffering from X but not diagnosed as such represents a false negative, something the doctor wants to avoid—which means that N FN should be small. In many classifiers, tweaking certain parameters can modify the values of N FP and N FN , thus affecting (at least to some extent) the classifier’s behavior, for instance, by improving recall at the cost of worsened precision or vice versa. This possibility can be useful in domains where the user has an idea as to which of these two quantities is more important. 7.9 Summary ● Bayesian reasoning provides a probabilistic approach to inference. ● Bayesian methods provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.
Supervised Learning ● In
the Naive Bayes algorithm, all the features are independent and are unrelated to each other.
Presence or absence of a feature does not influence the presence or absence of any other feature. ●
The
naive Bayes algorithm can be applied to a data set only if the features are categorical. ●
Maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model, given observations. ●
Instance-based methods are sometimes referred to as "lazy" learning methods because they delay processing until a new instance must be classified. ●
In nearest-neighbor learning the target function may be either discrete-valued or real-valued. ●
K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. ● Weighted KNN is a modified version of k-nearest neighbors. 7.10 Key Terms ● Precision: Precision is the probability that the classifier is right when labeling an example as positive. ● Recall: Recall is the probability that a positive example will be correctly recognized as such (by the classifier). ● Lazy learner algorithm: K-nearest neighbor
algorithm is also known as lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the
dataset. ● Binning:
The discretization is affected by putting the numeric values into categories known as bins. Because of this discretization is also known as binning. ●
Class-conditional independence: It
means that events are independent so long as they are conditioned on the same class value. 7.11
Check Your Progress Short-Answer Type
Supervised Learning Q1) _______ is a non-parametric and Instance-based learning algorithm. Q2)
In nearest-neighbor learning the target function may be either discrete-valued or real-valued. (
True/ False?) Q3) Disadvantage(s) of instance-based approaches: a) The cost of classifying new instances can be high. b) They consider all attributes of the instances when attempting to retrieve similar training examples from memory. c) Both (a) and (b). d) None of the above Q4) _________
is a method of estimating the parameters of a statistical model, given observations.
Q5) Bayesian reasoning follows a non- probabilistic approach to inference. (True/ False?) Long-Answer Type Q1) What are the basic features of Bayesian learning methods? Q2) State and discuss Bayes’ theorem with a suitable example. Q3) Differentiate between Bayes’ and Naive Bayes’ algorithm with an example. Q4) What is performance evaluation of Machine learning? Give the basic performance criteria. Q5) Write an algorithm for the K-nearest neighbor approach of learning. References: Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997. An Introduction to Machine Learning, 2 nd edition, Miroslav Kubat, Springer, 2017.
Supervised Learning
Supervised Learning Unit 8 – Neural Networks Structure 8.0 Introduction 8.1 Unit Objectives 8.2 Artificial Neurons 8.3 Activation Function 8.4 Perceptron 8.5 Artificial Neural Networks 8.6 Multilayer Perceptrons 8.7 The Backpropagation Algorithm 8.8 Summary 8.9 Key Terms 8.10 Check Your Progress 8.0 Introduction Neural network learning methods provide a robust approach to approximating real- valued, discrete-valued, and vector-valued target functions. For certain types of problems, such as learning to interpret complex real-world sensor data, artificial neural networks are among the most effective learning methods currently known.
An Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs. Just as a brain uses a network of interconnected cells called neurons to create a massive parallel processor, ANN uses a network of artificial neurons or nodes to solve learning problems.

To accomplish a relationship between biological neurons and artificial neurons, let us first
examine how a biological neuron functions. Figure 8.1 gives a schematic representation of the functioning of a biological neuron.
Supervised Learning Figure 8.1 Anatomy
of a neuron As per figure 8.1, the incoming signals are received by the cell's dendrites through a biochemical process. The process allows the impulse to be weighted according to its relative importance or frequency. As the cell body begins accumulating the incoming signals, a threshold is reached at which the cell fires and the output signal is transmitted via an electrochemical process down the axon. At the axon's terminals, the electric signal is again processed as a chemical signal to be passed to the neighboring neurons across a tiny gap known as a synapse. 4
Figure 8.2 Flow of signals in a biological neuron
Biological learning systems are built of very complex webs of interconnected neurons. The hu- man brain has an interconnected network of approximately 10 11 neurons, each connected, on an average, to 10 4 other neurons.
Neuron activity is typically excited or inhibited through connections to other neurons. The fastest neuron 4 Neuron (2018, February 15). In Wikipedia, The Free Encyclopedia. Retrieved 01:44, February 23, 2018
Supervised Learning switching times are known to be on the order of 10 -3 seconds, quite slow compared to computer switching speeds of 10 -10 seconds. Yet humans are able to make surprisingly complex decisions, surprisingly quickly.
Even though the neuron switching speeds are much slower than
computer switching speeds, humans are able to make complex decisions relatively quickly. Because of this, it is believed that the information processing capabilities of biological neural systems is a consequence of the ability of such systems to carry out a huge number of parallel processes distributed over many neurons. The developments in ANN systems are motivated by the desire to implement this kind of highly parallel computation using distributed representations.
The field of neural networks is too rich to be covered in the space we have at our disposal. We will therefore provide only the basic information about multilayer perceptrons. The unit describes how each of them classifies examples, and then describes some elementary mechanisms to induce them from training data. 8.1 Unit Objectives After completing this unit, the reader will be able to: ● Describe the basic concept of Artificial Neurons. ● Discuss Activation function and Perceptron. ● Gain knowledge about the Artificial Neural Networks. ● Illustrate Multilayer Perceptrons as Classifiers. 8.2
Artificial Neurons An artificial neuron is a mathematical function conceived as a model of biological neurons. Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs
and sums them to produce an output. Each input is separately weighted, and the sum is passed through a function known as an activation function or transfer function.
Supervised Learning Figure 8.3 Schematic representation of an Artificial Neuron The schematic representation of an Artificial neuron is shown in figure 8.3. The meaning of certain notations in the figure are given below: ➢
x 1 , x 2 , . . . x n : input signals ➢
w 1 , w 2 , . . . w n : weights associated with input signals ➢ x 0 : input signal taking the constant value 1 ➢
w 0 : weight associated with x 0 (called bias) ➢ ∑ : indicates summation of input signals ➢ f : function which produces the output ➢
y :
output signal
The function f can be expressed in the following form: .... (8.1) The small circles in the schematic representation of the artificial neuron shown in Figure 8.3 are called the nodes of the neuron. The circles on the left side which receive the values of x 1 , x 2 , . . . x n are called the input nodes and the circle on the right side which outputs the value of y is called
the output node. 8.3
Activation Function In an artificial neural network, the function which takes the incoming signals as input and produces the output signal is known as the activation function.
It should be noted that
Eq. (8.1) represents the activation function for the ANN model shown in figure 8.3
Supervised Learning
The activation function is the mechanism by which the artificial neuron processes incoming information and passes it throughout the network. Just as the artificial neuron is modeled after the biological version, so is the activation function modeled after nature's design. Let x 1 , x 2 , . . . , x n be the input signals, w 1 , w 2 , . . . , w n be the associated weights and −w 0 the threshold.
Let x = w 0 + w 1 x 1 + ⋯ + w n
x n .
Some of the widely used simple activation functions are discussed below. ● Unit-Step Functions: Unit-step function is defined as follows: ●
Threshold activation function: The threshold
activation function is defined by: ●
Sigmoid activation function (logistic function): One of the most commonly used activation functions is the sigmoid activation function. It is defined as follows: ●
Linear activation function: The linear activation function is defined by F(x) = mx + c ; This defines a straight line in the xy-plane
Supervised Learning ● Piecewise (or saturated) linear activation function: ●
Gaussian activation function ● Hyperbolic tangential activation function: 8.4 Perceptron

91%	MATCHING BLOCK 97/189	W

One type of ANN system is based on a unit called a perceptron, illustrated in Figure 8.4. A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise. More precisely, given inputs x 1 through x n , the output o(x 1 , . . . , x n) computed by the perceptron is where each w i is a real-valued constant, or weight, that determines the contribution Supervised Learning of input x i to the perceptron output. Notice the quantity (-w 0) is a threshold that the weighted combination of inputs w 1 x 1 + . . . + w n x n must surpass in order for the perceptron to output a 1. To simplify notation, we imagine an additional constant input x 0 = 1, allowing us to write the above inequality as , or in vector form as .For brevity, we will sometimes write the perceptron function as

where,

100%	MATCHING BLOCK 98/189	W

Learning a perceptron involves choosing values for the weights w 0 , . . . , w n . Therefore, the space H of candidate hypotheses considered in perceptron learning is the set of all possible real-valued weight vectors.

Figure 8.4 Schematic representation of a perceptron A single perceptron can be used to represent many boolean functions. For example, if we assume boolean values of 1 (true) and -1 (false), then one way to use a two-input perceptron to implement the AND function is to set the weights w 0 = -8, and w 1 = w 2 = 0.5. This perceptron can be made to represent the OR function instead by altering the threshold to wo = -0.3. In fact, AND and OR can be viewed as special cases of m-of-n functions: that is, functions where at least m of the n inputs to the perceptron must be true. The OR function corresponds to rn = 1 and the AND function to m = n. Any m-of-n function is easily represented using a perceptron by setting all input weights to the same value (e.g., 0.5) and then setting the threshold w 0 accordingly.
Supervised Learning Perceptrons can represent all of the primitive boolean functions AND, OR, NAND (~AND), and NOR (~OR). Unfortunately, however, some boolean functions cannot be represented by a single perceptron, such as the XOR function whose value is 1 if and only if x 1 ≠ x 2 . The ability of perceptrons to represent AND, OR, NAND, and NOR is important because every boolean function can be represented by some network of interconnected units based on these primitives. In fact, every boolean function can be represented by some network of perceptrons only two levels deep, in which

100%	MATCHING BLOCK 99/189	W

the inputs are fed to multiple units, and the outputs of these units are then input to a second, final stage. One way is to represent the boolean function in disjunctive normal form (i.e., as the disjunction (OR) of a set of conjunctions (ANDs) of the inputs and their negations). Note that the input to an AND perceptron can be negated simply by changing the sign of the corresponding input weight.

Figure 8.5 Representation of x 1 AND x 2 by a perceptron Perceptron Learning Algorithm This
algorithm can be applied only if the training examples are linearly separable.
In the algorithm, we use the following notations: n : Number of input variables y = f (z) : Output from the perceptron for an input vector z D = {(x 1 , d 1), . . . , (x s , d s)} : Training set of s samples
x j = (x j0 , x j1 , . . . , x jn) : The n-dimensional input vector
d j :
Desired output value of the perceptron for the input x j x ji : Value of the i-th feature of the j-th training input vector
x j0 : 1 w i : Weight of the i-th input variable
Supervised Learning w i (t) : Weight i at the t-th iteration Table 8.1 Perceptron Learning
Algorithm Step 1: Initialize the weights and the threshold. Weights may be initialized to 0 or to a small random value. Step 2: For each example j in the training set D, perform the following steps over the input x j and desired output d j : a) Calculate the actual output: y j (t) = f [w 0 (t)
x j0 + w 1 (t)x j1 + w 2 (t)x j2 + ⋯ + w n (t)x jn]
b) Update the weights: w i (t + 1) = w i (t) + (d j – y j (t))x ji
for all features 0 ≤ i ≤ n.
Step 3: Step 2 is repeated until the iteration error is less than a user- specified
error threshold , or a predetermined number of iterations have been completed, where s is again the size of the sample set.

97%	MATCHING BLOCK 100/189	W

The Perceptron Training Rule Although we are interested in learning networks of many interconnected units, let us begin by understanding how to learn the weights for a single perceptron. Here the precise learning problem is to determine a weight vector that causes the perceptron to produce the correct 1 output for each of the given training examples. Several algorithms are known to solve this learning problem. Here we consider the perceptron rule. These two algorithms are guaranteed to converge to somewhat different acceptable hypotheses, under somewhat different conditions. They are important to ANNs because they provide the basis for learning networks of many units. One way to learn an acceptable weight vector is to begin with random weights, then

Supervised Learning

98%	MATCHING BLOCK 101/189	W

iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example. This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly. Weights are modified at each step according to the perceptron training rule, which revises the weight w i associated with input x i according to the rule Here t is the target output for the current training example, o is the output generated by the perceptron, and η is a positive constant called the learning rate. The role of the learning rate is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases. Why should this update rule converge toward successful weight values? To get an intuitive feel, consider some specific cases. Suppose the training example is correctly classified already by the perceptron. In this case, (t - o) is zero, making Δw i zero, so that no weights are updated. Suppose the

w

99%	MATCHING BLOCK 102/189	W

perceptron outputs a -1, when the target output is +1. To make the perceptron output a+1 instead of -1 in this case, the weights must be altered to increase the value of .For example, if x i &lt;0, then increasing w i will bring the perceptron closer to correctly classifying this example. Notice the training rule will increase w, in this case, because (t - o),

η, and x i are all positive. For example, if x i = 0.8, η= 0.1,

95%	MATCHING BLOCK 103/189	W

t = 1, and o = - 1, then the weight update will be On the other hand, if t = -1 and o = 1, then weights associated with positive x i will be decreased rather than increased. In fact, the above learning procedure can be proven to converge within a finite number of applications of the perceptron training rule to a weight vector that correctly classifies all training examples, provided the training examples are linearly separable and provided a sufficiently small η is used. If the data are not linearly separable, convergence is not assured.

Supervised Learning 8.5
Artificial Neural Networks An artificial neural network (ANN) is a
computing system inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units called artificial neurons.
Each connection between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it. Each connection between artificial neurons has a weight attached to it that gets adjusted as learning proceeds. Artificial neurons may have a threshold such that only if the aggregate signal crosses that threshold the signal is sent. Artificial neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs,
as shown in figure 8.6.
Signals travel from the input layer to the output layer, possibly after traversing the layers multiple times.
Figure 8.6 Layers in

93%	MATCHING BLOCK 104/189	W

Artificial Neural Network ANN can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neuron outputs and neuron inputs can be viewed as the directed edges with weights. The ANN receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs. Afterward, each of the inputs is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. Supervised Learning If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

Characteristics of an ANN An ANN can be defined and implemented in several different ways. The way the following characteristics are defined determines a particular variant of an ANN. ● The activation function: This function defines how a neuron's combined input signals are transformed into a single output signal to be broadcasted further in the network. ● The network topology (or architecture): This describes the number of neurons in the model as well as the number of layers and manner in which they are connected. ● The training algorithm: This algorithm specifies how connection weights are set in order to inhibit or excite neurons in proportion to the input signal.
Layers of ANN ●

100%	MATCHING BLOCK 105/189	W

Input Layer: As the name suggests, it accepts inputs in several different formats provided by the programmer. ● Hidden Layer: The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns. ● Output Layer: The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function. It determines

the

100%	MATCHING BLOCK 106/189	W

weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer.

Supervised Learning

94%	MATCHING BLOCK 107/189	W

Types of ANN There are various types of ANN depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification. ● Feedback ANN: In this type of ANN, the output returns into the network to accomplish the best-evolved results internally.

98%	MATCHING BLOCK 108/189	W

The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs. ● Feed-Forward ANN: A feed-forward network is a basic neural network consisting of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

98%	MATCHING BLOCK 109/189	W

Advantages of ANN ● Parallel processing capability: Artificial neural networks have a numerical value that can perform more than one task simultaneously. ● Storing data on the entire network: Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working. ● Capability to work with incomplete knowledge: After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data. ● Having a memory distribution: For ANN to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output. ● Having fault tolerance: Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Supervised Learning

98%	MATCHING BLOCK 110/189	W

Disadvantages of ANN ● Assurance of proper network structure: There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error. ● Unrecognized behavior of the network: It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network. ● Hardware dependence: Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent. ● Difficulty of showing the issue to the network: ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities. 8.6

Multilayer Perceptrons

98%	MATCHING BLOCK 111/189	W

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function f(·):R m → R o by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X = x 1 ,x 2 ,...,x m and a target y, it can learn a nonlinear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 8.7 shows a one hidden layer MLP with scalar output.

Supervised Learning Figure 8.7 An example of

100%	MATCHING BLOCK 112/189	W

one hidden layer MLP with scalar output The leftmost layer, known as the input layer, consists of a set of neurons {x i |x 1 , x 2 ,...,x m } representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation w 1 x 1 + w 2 x 2 +...+ w m x m , followed by a non-linear activation function g(·):R→R -like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

87%	MATCHING BLOCK 113/189	W

Advantages of Multilayer Perceptron ● Capability to learn non-linear models. ● Capability to learn models in real-time. Disadvantages of Multilayer Perceptron ● MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy. ● MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations. ● MLP is sensitive to feature scaling.

Supervised Learning 8.7 The Backpropagation Algorithm The backpropagation algorithm was discovered in 1985-86. The Backpropagation algorithm learns the weights for a multilayer network, given a network with a fixed set of units and interconnections. It employs gradient descent to attempt to minimize the squared error between the network output values and the target values for these outputs. Because we are considering networks with multiple output units rather than single units as before, we begin by redefining E to sum the errors over all of the network output units where output is the set of output units in the network, and t kd and o kd are the target and output values associated with the kth output unit and training example d. The learning problem faced by backpropagation is to search a large hypothesis space defined by all possible weight values for all the units in the network. An outline of the algorithm is given below: 1.
Initially the weights are assigned at random. 2. Then the algorithm iterates through many cycles of two processes until a stopping criterion is reached. Each cycle is known as an epoch. Each epoch includes: a) A forward phase in which the neurons are activated in sequence from the input layer to the output layer, applying each neuron's weights and activation function along the way. Upon reaching the final layer, an output signal is produced. b) A backward phase in which the network's output signal resulting from the forward phase is compared to the true target value in the training data. The difference between the network's output signal and the true value results in an error that is propagated backwards in the network to modify the connection weights between neurons and reduce future errors. 3. The technique used to determine how much a weight should be changed is known as
the gradient descent method. At every stage of the computation,
Supervised Learning
the error is a function of the weights. If we plot the error against the wights, we get a higher dimensional analog of something like a curve or surface. At any point on this surface, the gradient suggests how steeply the error will be reduced or increased for a change in the weight. The algorithm will attempt to change the weights that result in the greatest reduction in error (see Figure 8.8).
Figure 8.8 A simplified model of the error surface showing the direction of gradient
The backpropagation algorithm trains a given feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted. The adjustments are based on the mean square error of the output response to the sample input and it is known as the delta learning rule. The set of these sample patterns are repeatedly presented to the network until the error value is minimized. There are various notations used in the algorithm. M : Number of layers (excluding the input layer which is assigned the layer number 0)
N j : Number of neurons (nodes) in j-th layer X p = (X p1 , X p2 , . . . , X pN0) : p-th training sample
Supervised Learning T p = (T p1 , T p2 , . . . , T pNM) : Known output corresponding to the p-th training sample O p = (O p1 , O p2 , . . . , O pNM) : Actual
output by the network corresponding to the p-th training sample Y ji : Output from the i-th neuron in layer j
W jik :
Connection weight from k-th neuron in layer (j – 1) to i-th neuron in layer j δ ji : Error value associated with the i-th neuron in layer j
Table 8.2 The Backpropagation Algorithm
Supervised Learning
Step 1: Initialize connection weights into small random values. Step 2: Present the pth sample input vector of pattern
X p = (X p1 , X p2 , ..., X pN0) and the corresponding output target T p = (T p1 , T p2 , ..., T pNM)
to the network. Step 3: Pass the input values to the first layer, layer 1. For every input node i in layer 0, perform: Y 0
i = X pi .
Step 4: For every neuron i in every layer j = 1, 2, ..., M , find the output from the neuron:
Step 5: Obtain output values. For every output node i in layer M, perform:
O pi = Y Mi . Step 6: Calculate error value δ ji
for every neuron i in every layer in backward order j = M, M–1, . . . , 2, 1, from output to input layer, followed by weight adjustments. For the output layer, the error value is:
δ Mi = Y Mi (1 – Y Mi)(T pi – Y Mi),
and for hidden layers:
The weight adjustment can be done for every connection from neuron k in layer (j – 1) to every neuron j in every layer
i: where η represents
the
weight adjustment factor (called the learning rate) normalized between 0 and 1. Step 7: The actions in steps 2 through 6 will be repeated for every training sample pattern p, and repeated for these sets until the sum of the squares of output errors is minimized.
Supervised Learning
Figure 8.9 Notations of backpropagation algorithm: The i-th node in layer j 8.8
Summary ● Neural network learning methods provide a robust approach to approximating real-valued, discrete-valued, and vector-valued target functions. ●

An Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs. ●
An
artificial neuron is a mathematical function conceived as a model of biological neurons. ●
In an artificial neural network, the function which takes the incoming signals as input and produces the output signal is known as the activation function. ●

A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise. 8.9

Key Terms ● Neurons: It is the basic working unit of the brain, a specialized cell designed to transmit information to other nerve cells, muscle, or gland cells.
ANN uses a network of artificial neurons or nodes to solve learning problems. ●
Dendrites:
The incoming signals are received by the cell's dendrites through a biochemical
process. ●
Synapses: Synapses are microscopic gaps that separate the terminal buttons of
Supervised Learning one neuron from receptors (usually, located on the dendrites) of another neuron. ●
The training algorithm: This algorithm specifies how connection weights are set in order to inhibit or excite neurons in proportion to the input signal. 8.10
Check Your Progress Short-Answer Type Q1) Artificial neurons are also termed as _______. Q2)
An Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal. (
True/ False?) Q3) In a neuron, at ________
terminals, the electric signal is processed as a chemical signal to be passed to the neighboring neurons.
Q4) Which of the following can be used as an activation function? a) Unit-step function b) Linear function c) Sigmoid function d) All of the above Long-Answer Type Q1) Write an algorithm for perceptron learning. Q2) What is an activation function? Give examples. Q3) Define perceptron training rule. How is it utilized to solve learning problems? Q4) Write a short note on Artificial Neural Networks. References: Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997. An Introduction to Machine Learning, 2 nd edition, Miroslav Kubat, Springer, 2017.
Supervised Learning Unit 9 – Support Vector Machines Structure 9.0 Introduction 9.1 Unit Objectives 9.2 SVM Properties 9.3 Hard Margin SVM 9.4 Soft Margin SVM 9.5 Kernel SVM 9.6 Multiclass SVM 9.7 Application of SVM 9.8 Summary 9.9 Key Terms 9.10 Check Your Progress 9.0 Introduction

Support Vector Machine or SVM is one of the most popular supervised learning algorithms, which is used for classification as well as regression problems. However, primarily, it is used for classification problems in machine learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n- dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence

the

algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

Supervised Learning Figure 9.1 Categories in Hyperplane Support vector machine (SVM) technique is also defined as a sparse kernel decision machine that avoids computing posterior probabilities when building its learning model. SVM offers a principled approach to machine learning problems because of its mathematical foundation in statistical learning theory. SVM constructs its solution in terms of a subset of the training input. SVM has been extensively used for classification, regression, novelty detection tasks, and feature reduction. Generally, SVM evolved from a robust theory of implementation, whereas artificial neural networks (ANN) moved heuristically from application to theory. SVM distinguishes itself from ANN in that it does not suffer from the classical multi-local minima—the issue of dimensionality and overfitting. Overfitting, which happens when the machine learning model strives to achieve a zero error on all training data, is more likely to occur with machine learning approaches whose training metrics depend on variants of the sum of squares error. By minimizing the structural risk rather than the empirical risk, as in the case of ANN, SVM avoids overfitting. SVM does not control model complexity, as ANN does, by limiting the feature set; instead, it automatically determines the model complexity by selecting the number of support vectors. In this unit, we present a brief introduction to SVM, and we discuss hard margin and soft margin SVM. In the beginning we try to define SVM properties and the need for SVM. The mathematical formulation of SVM is presented, and theory for the implementation of SVM is briefly discussed. Finally some conclusions on SVM and application areas are included.
Supervised Learning 9.1 Unit Objectives After completing this unit, the reader will be able to: ● Describe the basic concept of Support Vector Machine (SVM). ● Discuss the properties of SVM. ● Gain knowledge about Hard margin, Soft margin and kernel SVM. ● Illustrate the applications of SVM. 9.2 SVM Properties SVM was introduced by Boser, Guyon, and Vapnik (1992) during the Fifth Annual Association for Computing Machinery Workshop on Computational Learning Theory. SVM is deeply rooted to the principles of statistics, optimization, and machine learning. Known for their robustness, good generalization ability, and unique global optimum solutions, SVMs are probably the most popular machine learning approach for supervised learning, yet their principle is very simple. In his comparison of SVM with 16 classifiers, on 21 datasets, Meyer, Leisch, and Hornik (2003) showed that SVM is one of the most powerful classifiers in machine learning. SVMs have found their way into countless applications, such as weather prediction, power estimation stock prediction, defect classification, speaker recognition, handwriting identification, image and audio processing, video analysis, and medical diagnosis. The following properties make SVM an attractive machine learning framework: ● SVM is a sparse technique Like nonparametric methods, SVM requires that all the training data be available, that is, stored in memory during the training phase, when the parameters of the SVM model are learned. However, once the model parameters are identified, SVM depends only on a subset of these training instances, called support vectors, for future prediction. Support vectors define the margins of the hyperplanes. The complexity of the classification task with SVM depends on the number of support vectors rather than the dimensionality of the input space. ● SVM is a kernel technique SVM uses the kernel trick to map the data into a higher-dimensional space before solving the machine learning task as a convex optimization problem in Supervised Learning which optima are found analytically rather than heuristically, as with other machine learning techniques. The instances that have different labels share the input space in a manner that prevents a linear hyperplane from correctly separating the different classes involved in this classification task. SVM maps the data, using predefined kernel functions, into a new but higher-dimensional space, where a linear separator would be able to discriminate between the different classes. ● SVM is a maximum margin separator Beyond minimizing the error or a cost function, based on the training datasets (similar to other discriminant machine learning techniques), SVM imposes an additional constraint on the optimization problem: the hyperplane needs to be situated such that it is at a maximum distance from the different classes. Such a term forces the optimization step to find the hyperplane that would eventually generalize better because it is situated at an equal and maximum distance from the classes. This is essential, because training is done on a sample of the population, whereas prediction is to be performed on yet-to-be-seen instances that may have a distribution that is slightly different from that of the subset trained on. Figure 9.2 Maximal separating hyperplane, margin and support vectors
Consider a linearly separable data set having two class labels "–1" and "+1". Consider a separating hyperplane H for the data set. ➔ Consider the perpendicular distances from the training instances to the separating hyperplane H and consider the smallest such perpendicular
Supervised Learning
distance. The double of this smallest distance is called the margin of the separating hyperplane H (
as shown in figure 9.2). ➔
The hyperplane for which the margin is the largest is called the maximal margin hyperplane (also called maximum margin hyperplane) or the optimal separating hyperplane. ➔ The maximal margin hyperplane is also called the support vector machine for the data set. ➔ The data points that lie closest to the maximal margin hyperplane are called the support
vectors. ●

SVM uses structural risk minimization (SRM) and satisfies the duality and convexity requirements. Structural Risk Minimization (SRM) is an inductive principle that selects a model for learning from a finite training dataset. As an indicator of capacity control, SRM proposes a trade-off between the VC dimensions, that is, the hypothesis of space complexity and the empirical error (a measure of how well the model fits the training data). SRM's formulation is a convex optimization with n variables in the cost function to be maximized and m constraints, solvable in polynomial time. SRM uses a set of models sequenced in an increasing order of complexity. Figure 9.2 shows how the overall model error varies with the complexity index of a machine learning model. For non- complex models, the error is high because a simple model cannot capture all the complexity of the data which results in an underfitting situation. As the complexity index increases, the error reaches its minimum for the optimal model indexed h* before it starts increasing again. For high model indices, the structure starts adapting its learning model to the training data which results in an overfitting that reduces the training error value and increases the model VC however, at the expense of a deterioration in the test error. SVM elegantly groups multiple features that were already being proposed in research in the 1960s to form what is referred to as the maximal margin classifier. SVM borrows concepts from large-margin hyperplanes (Duda 1973; Cover 1995; Vapnik and Lerner 1963; Vapnik and Chervonenkis 1964); kernels as inner products in the feature space (Aizermann, Braverman, and Rozonoer 1964); kernel usage (Wahba 1990; Poggio
Supervised Learning 1990) and sparseness (Cover 1995). Mangasarian (1965) also proposed an optimization approach similar to the one adopted by SVM. The concept of slack, used to address noise in data and nonseparability, was originally introduced by Smith (1968) and was further enhanced by Bennett and Mangasarian (1992). Incorporated into SVM formulation by Cortes (1995), soft-margin SVM represents a modification of the hard-margin SVM through its adoption of the concept of slack to account for noisy data at the separating boundaries. Figure 9.2 Relationship between error trends and model index SVM relies on the complexity of the hypothesis space and empirical error (a measure of how well the model fits the training data). Vapnik-Chervonenkis (VC) theory proves that a VC bound on the risk exists. VC is a measure of the complexity of the hypothesis space. The VC dimension of a hypothesis $\mathcal{H}$ relates to the maximum number of points that can be shattered by $\mathcal{H}$. $\mathcal{H}$ shatters N points, if $\mathcal{H}$ correctly separates all the positive instances from the negative ones. In other words, the VC capacity is equal to the number of training points N that the model can separate into 2 N different labels. This capacity is related to the amount of training data available. The VC dimension h affects the generalization error, as it is bounded by ||w|| (vector norm) where w is the weight vector of the separating hyperplane and the radius of the smallest sphere R that contains all the training points, according to: . The overall error of a machine learning model consists of ε = ε emp + ε g , where ε emp is
Supervised Learning the training error, and ε g is the generalization error. The empirical risk of a model f is The lower bound for risk is where 1 – η is the probability of this bound being true for any function in the class of functions with VC dimension h, independent of the data distribution. It should be noted that there are 2 N different learning problems that can be defined, as N points can be labeled in 2 N manners as positive or negative. For instance, for three points, there are 24 different labels and 8 different classification boundaries that can be learned. Thus, the VC dimension in R 2 is 3. 9.3 Hard Margin SVM The SVM technique is a classifier that finds a hyperplane or a function g(x) = w T x + b that correctly separates two classes with a maximum margin. Figure 9.3 shows a separating hyperplane corresponding to a hard-margin SVM (also called a linear SVM). Figure 9.3 (a) Hard-maximum-margin separating hyperplane (b) Illustration of Linear SVM. (Source- Andrew W. Moore slides, 2003) ● Mathematically, given a set of points x i that belong to two linearly separable classes ω 1 , ω 2 , the distance of any instance from the hyperplane is equal to . SVM aims to find w, b, such that the value of g(x) equals 1 for the nearest data points belonging to class ω 1 and –1 for the nearest ones of ω 2 .
Supervised Learning Figure 9.4 Representation of Hyperplanes This can be viewed as having a margin of whereas, w T x + b = 1 for x∈ω 1 , and w T x + b = – 1 for x∈ ω 2 . This leads to an optimization problem that minimizes the objective function subject to the constraint When an optimization problem—whether minimization or maximization—has constraints in the variables being optimized, the cost or error function is augmented by adding to it the constraints, multiplied by the Lagrange multipliers. ● Based on the above discussion, we now formulate the SVM problem as the following optimization problem. Given a two-class linearly separable dataset of N points of the form
where the y i 's are either +1 or 1, find a vector and a number b which
Supervised Learning ● The solution of the SVM problem gives us a classifier
for classifying unclassified data instances. This is known as the SVM classifier for a given dataset.
Table 9.1 represents the algorithm to find SVM
Classifier. Let and b = b ∗ be a solution of the SVM problem. Let $\vec{x}$ be an unclassified data instance. ➔ Assign the class label +1 to $\vec{x}$ if $\vec{w} * \cdot \vec{x} - b *$ &lt; 0. ➔ Assign the class label -1 to $\vec{x}$
if $\vec{w} * \cdot \vec{x} - b *$ &gt; 0.
Table 9.1
Algorithm to find SVM classifier Given a two-class linearly separable dataset of N points of the form
where the y i 's are either +1 or 1. 9.4 Soft Margin SVM
The algorithm for finding the SVM classifier will give a solution only if the given two- class dataset is linearly separable. But, in real life problems, two-class datasets are only rarely linearly separable. In such a case we introduce additional variables, ξ i , called slack variables which store deviations from the margin. There are two types of
Supervised Learning
deviation: ● An instance may lie on the wrong side of the hyperplane and be misclassified. ● It may be on the right side but may lie in the margin, namely, not sufficiently away from the hyperplane. (

as shown in figure 9.5) It can be illustrated that: ➔ If ξ i = 0, then x⃗ i
is correctly classified and there is no problem with x⃗ i . ➔ If 0 &gt;
ξ i &gt; 1 then x⃗ i is
correctly classified but it is in the margin. ➔ If ξ i &lt; 1, then x⃗ i is misclassified.
The sum
is defined as the soft error and this is added as a penalty to the function to be minimized. We also introduce a
factor C to the soft error.
As C is increased, a tighter margin is obtained, and more emphasis is placed on minimizing the number of misclassifications. As C is decreased, more violations are allowed, because maximizing the margin between the two classes becomes the SVM aim.
With these modifications,
we now reformulate the SVM problem as follows:
Given a two-class linearly separable dataset of N points of the form
where the y i 's are either +1 or 1, find vectors ω⃗ and ξ⃗ and a number b which
Supervised Learning Figure 9.5 Soft margin hyperplanes
The hyperplanes given by the equations with the values of ω⃗
and b obtained as solutions of the reformulated problem, are called the soft margin hyperplanes for the SVM problem. 9.5 Kernel SVM When a problem is not linearly separable in input space, soft-margin SVM cannot find a robust separating hyperplane that minimizes the number of misclassified data points and that generalizes well. For that,
a kernel can be used to transform the data to a higher-dimensional space,
referred to as kernel space, where data will be linearly separable. In the kernel space a linear hyperplane can thus be obtained to separate the different classes involved in the classification task instead of solving a high-order separating hypersurface in the input space. This is an attractive method, because the overhead on going to kernel space is insignificant compared with learning a nonlinear surface. A kernel should be a Hermitian and positive semidefinite matrix and needs to satisfy Mercer's theorem, which translates into evaluating the kernel or Gram matrix on all pairs of data points as positive and semidefinite, forming
Supervised Learning where φ(x) belongs to the Hilbert space. In other words, ∬ K (x, u) g (x) g (u) dxdu ≥ 0 ∀ g (x) , where ∫ g 2 (x) dx &gt; + •. Some popular kernel functions include ● Linear kernel: K (x, u) = x T . u ● Polynomial function: K (x , u) = (ax T u + c) q , q &lt; 0 ● Hyperbolic tangent (sigmoid): K (x , u) = tanh (βx T u +) ● Gaussian radial basis function (RBF): ● Laplacian radial basis function: ● Randomized blocks analysis of variance (ANOVA RB) kernel: ● Linear spline kernel in 1D: Kernel selection is heavily dependent on the data specifics. For instance, the linear kernel is useful in large sparse data vectors. However, it ranks behind the polynomial kernel, which avoids zeroing the Hessian. The polynomial kernel is widely used in image processing, whereas the ANOVA RB kernel is usually reserved for regression tasks. The Gaussian and Laplace RBFs are general-purpose kernels that are mostly applied in the absence of prior knowledge. A kernel matrix that ends up being diagonal indicates that the feature space is redundant and that another kernel should be tried after feature reduction. It should be noted that when kernels are used to transform the feature vectors from input space to kernel space for linearly non separable datasets, the kernel matrix computation requires massive memory and computational resources, for big data. Figure 9.6 displays the two-dimensional exclusive-OR (XOR) data, a linearly non separable distribution in input space (upper-left) as well as in the feature space. In the latter, 16 points (for different sets) are created for the four inputs when the kernel is
Supervised Learning applied. The choice of the Gaussian RBF kernel-smoothing parameter σ 2 affects the distribution of the data in the kernel space. Because the choice of parameter value is essential for transforming the data from a linearly non separable space to a linearly separable one, grid searches are performed to find the most suitable values. Figure 9.6 Two-dimensional XOR data, from input space to kernel space The formulation of the kernel SVM is given as: subject to where φ(x i) is such that K (x i , x j) = φ (x i) . φ (x j).
Supervised Learning Table 9.2 Algorithm for Kernel SVM 9.6
Multiclass SVM In machine learning, the multiclass classification is the problem of classifying instances into one of three or more classes.
It should also be noted that
classifying instances into one of the two classes is called binary classification. Support vector machines can be constructed only when the dataset has only two class-labels and is linearly separable. We have already discussed a method to extend the concept of SVM's to the case where the dataset is not linearly separable. In this section we consider how the SVM's can be used to obtain classifiers when there are more than two class labels. Two methods are generally used to handle such cases known by the names "One- against-all" and "one-against-one". ● One-against-all Method The One-Against-All (OAA) SVMs were first introduced by Vladimir Vapnik in 1995. Let there be p class labels, say, c 1 , c 2 , . . . , c p . We construct the following p two-
Supervised Learning
class datasets and
obtain the corresponding SVM classifiers. First, we assign the class labels +1
to
all instances having class label c 1 and the class label –1 to all the remaining instances in the data set. Let be the SVM classifier function for the resulting two-class dataset. Next, we assign the class labels +1 to all
instances
having class label c 2 and the class label –1 to all the remaining instances in the data set. Let

be the SVM classifier function for the resulting two-class dataset. We continue like this and generate SVM classifier functions .
Two criteria have been developed to assign a class label to a test instance $\vec{z}$. 1.
A data point $\vec{z}$ would be classified under a certain class if and only if that
class's SVM accepted
it and all other classes' SVMs rejected it. Thus $\vec{z}$ will be assigned c i if for all j ≠ i. 2. $\vec{z}$ is the assigned the class label c i if has the highest value among regardless of sign. Figure 9.7 illustrates the one-against-all method with three classes.
Figure 9.7
One-
against-all Method ● One-against-one Method In the one-against-one (OAO) (also called one-vs-one (OVO)) strategy, a SVM classifier
is constructed for each pair of classes. If there are p different class labels, a total of p(p – 1)/2 classifiers are constructed. An unknown instance is classified with the class getting the most votes. Ties are broken arbitrarily.
For example, let there be three classes, A, B and C. In the OVO method we construct 3(3 – 1)/2 = 3
SVM binary classifiers. Now, if $\vec{z}$ is to be classified, we apply each of the three classifiers
to $\vec{z}$.
Let the three classifiers assign the classes A, B, B respectively to $\vec{z}$ Since a label to $\vec{z}$ is assigned by the majority voting, in this example, we assign the class label of B
to
$\vec{z}$.
Supervised Learning
One-vs-one (OVO) strategy is not a particular feature of SVM. Indeed, OVO can be applied to any binary classifier to solve multi-class classification
problems. Figure 9.8 One-against-one Method 9.7 Application of SVM As we have seen, SVMs depend on supervised learning algorithms. The aim of using SVM is to correctly classify unseen data. SVMs have a number of applications in several fields. SVMs are more flexible for new data. This makes them easier to use in the applications where we need more flexibility in the training and testing data. Due to the large margin that it likes to generate, we can fit in more data and classify it perfectly. Some common applications of SVM discussed below- ● Image-based analysis and face detection SVMs are used in many tasks when it comes to dealing with images. SVMs are particularly used in one definite application of image processing: facial features extraction and recognition. While working with facial features, there is a need for algorithms that can properly classify different features based on very fine-tuned feature extractions. Facial expressions are one of the most versatile features while working with image processing. Like your face can be way different than that of others. But still, the algorithm should be able to classify and acknowledge the facial expressions given by both of you, even though the model is trained on just your face or the other person. ● Geospatial data analysis One of the fields which has the most uncleaned data is the geoscience field. The geospatial data is one of the noisiest data. SVMs still work pretty well, especially when the dataset size is not so high. The most problematic set of geospatial analysis problems are inversion problems. For example, the geo-
Supervised Learning sounding problem. In this, the electromagnetic data acquired is passed through the SVMs algorithm and it can correctly classify those. ● Medical Applications Human-based proteins are very delicate structures and are prone to too much noise as well as errors while using the SVM algorithms for recognition. Another field there is the remote homology which uses SVMs to the fullest. This is where the analysis is dependent on how the protein sequences are modeled. The use of SVMs does spread over to the detection of various diseases, based on either image data or text data (value-based). However, it is important to mention that they are widely used in many fine crafted classification applications, necessary for the medical sector. ● Text-based Applications We can use SVMs to classify the handwriting of two different people. SVMs train better when it comes to applications such as detection of the curves and straights used in typical handwriting. SVMs can also be used in pure computer- based texts. For example, a typical text-based classification task is the email spam classifier. In that, we need to classify an email that is spam from the email which is not a spam. It is one of the most used applications in the email delivery systems provided by platforms like Gmail. SVMs can correctly classify the spams from the pool of emails. Some of the SVMs trained on structured data achieve as high as 97% accuracy for this application. Audio-based analysis is also a field in which SVMs offer a solution. We can use many audio-based preprocessing functions and then use the SVMs for classification or just simple speech recognition. Both of these applications are quite useful and widely used. ● Security-based Applications SVMs are used for basic encryption as well as complex analysis of different materials to see and even break the encryptions and other security measures. SVMs can also be used to detect the encryption schemas uploaded to the images, to hide them. This field of application has a wide scope of research. 9.8 Summary ●

Support Vector Machine or SVM is one of the most popular supervised learning

Supervised Learning algorithms, which is used for classification as well as regression problems. ●

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors. ●

The data points that lie closest to the maximal margin hyperplane are called the support
vectors. ●
Structural Risk Minimization (SRM) is an inductive principle that selects a model for learning from a finite training dataset. ●

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier. ● Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier. ●

A kernel can be used to transform the data to a higher-dimensional space,
referred to as kernel space, where data will be linearly separable. ●
In machine learning, the multiclass classification is the problem of classifying instances into one of three or more classes. 9.9 Key Terms ● Hyperplane: Hyperplanes are decision boundaries that help classify the data points. ● Support Vectors: SVM depends only on a subset of the training instances, called support vectors. ● Slack Variables: The additional variables that store deviations from the margin in Soft margin SVM. ● Binary classification:
Classifying instances into one of the two classes is called binary classification. 9.10
Check Your Progress Short-Answer Type Q1) _________ are decision boundaries that help classify the data points. Q2) SVMs are used for basic encryption as well as complex analysis of different
Supervised Learning materials to see and even break the encryptions and other security measures. (True/ False?) Q3) In the _______ strategy, a multiclass SVM classifier is constructed for each pair of classes. Q4) A kernel should be- a) Hermitian b) positive semidefinite matrix c) Both a) & b) d) None of the above Long-Answer Type Q1) What is meant by maximum margin hyperplane? Q2)
State the mathematical formulation of the SVM problem. Give an outline of the method for solving the problem.
Q3)
Define the support vectors of a two-class dataset. Q4) What is a kernel function? Give an example.
References: Machine Learning: a Probabilistic Perspective, Kevin Patrick Murphy, MIT Press, March 2014. Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz and Shai Ben-David, Cambridge University Press, 2014. Tutorial on Support Vector Machine (SVM), Vikramaditya Jakkula, School of EECS, Washington State University, Pullman. Support Vector Machines for Classification, Mariette Awad, Rahul Khanna, ResearchGate, 2015. (DOI: 10.1007/978-1-4302-5990-9_3)
Unsupervised Learning MODULE: III UNSUPERVISED LEARNING
Unsupervised Learning
Unsupervised Learning Unit 10 – Introduction to Clustering Structure 10.0 Introduction 10.1 Unit Objectives 10.2 Features of Clustering 10.3 K-means Clustering 10.4 Similarity Measures 10.4.1 Numeric Attributes 10.4.2 Binary Attributes 10.5 Applications of Clustering 10.6 Summary 10.7 Key Terms 10.8 Check Your Progress 10.0 Introduction We have already learnt that unsupervised

learning is used to draw inferences from data sets consisting of input data without labeled responses. In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of the

94%	MATCHING BLOCK 123/189	W

dataset, group that data according to similarities, and represent that dataset in a compressed format. The unsupervised learning algorithm

can be further categorized into two types of problems: ●
Clustering: It can be defined as the method of organizing data instances into groups based on their similarity. In other words a cluster is a collection or group of data instances that are similar to each other and dissimilar to data instances belonging to other clusters. ●

95%	MATCHING BLOCK 124/189	W

Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occur together in the dataset. Association rule

Unsupervised Learning

100%	MATCHING BLOCK 125/189	W

makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Clustering is the most important unsupervised learning approach associated with machine learning. It can be viewed as a method for data exploration which essentially means looking for patterns or structures in the data space that may be of interest in a collection of unlabeled data. Essentially no classes are associated with data instances a priori as in the case of supervised learning. A set of data instances or samples can be grouped differently based on different criteria or features, in other words clustering is subjective. For example, we consider a set of seven people. They have been grouped into three clusters based on whether they are school employees, they belong to a family or based on their gender. Therefore choosing the attributes or features based on which clustering is to be carried out is an important aspect of clustering just as it was for classification. When we are given a set of instances or examples represented as a set of points, we need to define the notion of distance between these points. We then group the points into some number of clusters, such that members of a cluster are close or similar to each other while members of different clusters are dissimilar or farther apart than members belonging to the same cluster. Figure 10.1 shows a data set that has three natural clusters where the data points group together based on the distance. An outlier is a data point that is isolated from all other data points. Figure 10.1 Clusters and Outlier In this unit, we will illustrate clustering in detail. One of the most widely used K- means clustering is also discussed. Similarity measures on the basis of numeric and binary attributes are described. A wide range of applications of clustering are also
Unsupervised Learning listed in the unit. 10.1 Unit Objectives After completing this unit, the reader will be able to: ● Describe the basic concept of clustering in unsupervised learning. ● Discuss the features and applications of clustering. ● Gain knowledge about similarity measures in clustering. 10.2 Features of Clustering In general clustering deals with high dimensional data. Examples of such data include text documents and images. Dealing with such high dimensional data is an important aspect of clustering. Another important aspect that influences the effectiveness of clustering is the choice of distance function or the similarity or dissimilarity measure used. The basic clustering algorithms can be divided into two types namely hierarchical clustering and partitional clustering. The choice of which type and which algorithm is another important aspect of clustering. We also need to decide on the parameters to evaluate clustering quality. In general the algorithms strive to maximize inter-cluster distance and minimize intra-cluster distance. In other words the quality of clustering depends on the algorithm used, the distance function selected, and the application for which it is to be used. The inter-cluster distance shows the distance between the data point with the cluster center, while the intra-cluster distance shows the distance between the data point of one cluster with the other data point in the other cluster. High Dimensional Data Often as we discussed clustering needs to deal with high dimensional data where given a cloud of data points we want to understand its structure. Clustering needs to capture the structure using some dimensionality techniques and then performing clustering. Partitional Clustering In partitional clustering the data points are divided into a finite number of partitions which are disjoint subsets of the set of data points, that is, each data point is assigned to exactly one subset. These types of clustering algorithms can be viewed as problems

Unsupervised Learning of iteratively relocating data points between clusters until an optimal partition of the data points has been obtained. One of the most important and popular clustering algorithms – the k-means algorithm is an example of partitional clustering. In the basic algorithm the data points are partitioned into K clusters and the partitioning criterion is optimized using methods such as minimizing the squared error. In the case of the basic iterative algorithm of K-means or K-medoids, both of which belong to the partitional clustering category the convergence is local and the globally optimal solution is not always guaranteed. The number of data points in any data set is finite and the number of distinct partitions is also finite. It is possible to tackle the problem of local minima by using exhaustive search methods. 10.3 K-means Clustering The K-means algorithm is a heuristic method where each cluster is represented by the center of the cluster and the algorithm converges when the centroids of the clusters do not change. The K-means algorithm is the simplest partitioning method for clustering and widely used in data mining applications. As the case with any partitional algorithm, basically the K-means algorithm is an iterative algorithm which divides the given data set into K disjoint groups. This partitional method uses prototypes for representing the cluster. For a given K, we need to

find a partition of K clusters that optimizes the chosen partitioning criterion

or cost function. K-means Algorithm Figure 10.2 represents the flow diagram for the K-means algorithm.

Unsupervised Learning Figure 10.2 Flow Diagram of K-means Algorithm Steps of K-Means ● Initialize k values of centroids ● The following two steps are repeated until the data points do not change partitions and there is no change in the centroid a) Partition the data points according to the current centroids. The similarity between each data point and each centroid is determined and the data points are moved to the partition to which it is most similar. b) The new centroids of the data points in each partition are then calculated. The procedure follows a simple and easy way

to classify a given data set through a certain number of clusters (assume k clusters)

fixed apriori.

The main idea is to define k centers, one for each cluster. These centers

are placed as much as possible far away from each other.

The next step is to take each point belonging to a given dataset and associate it to the nearest center.

When no point is pending,

the first step is completed and an early group age

is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points

and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes

are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by: where, ||xi - vj|| is the Euclidean distance between xi and vj. c i is the number of data points in i th cluster. c is the number of cluster centers.

Unsupervised Learning Table 10.1 Algorithm for K-means Clustering Let X = {x 1 , x 2 , x 3 ,........, x n } be the set of n data points and V = {v 1 , v 2 ,......., v c } be the set of centroids. 1. First randomly select 'k' cluster centroids. 2. Then calculate the distance between each data point and the cluster centroids. 3. Find the cluster whose centroid is nearest to the data point. Then assign the data point to this cluster. 4. Recalculate the new cluster centroid using:

where, c i represents the number of data points in i th cluster. 5.

Recalculate the distance between each data point and the newly obtained cluster centroids. 6. If no data point has been reassigned then we stop, otherwise we repeat from step 3. Example: In the example shown in figure 10.3, we see that initially the cluster centroids are chosen at random as we are talking about an unsupervised learning technique where we are not provided with the labelled samples. Even after the first round using these randomly chosen centroids the complete set of data points are partitioned all at once. As you can see after round 2 the centroids have moved since the data points have been used to calculate the new centroids. The process is repeated and the centroids are relocated until convergence occurs when no data point changes partitions.

Unsupervised Learning Figure 10.3 Example of the K-means Algorithm Issues associated with the K-means Algorithm ● Choosing the right value of K: The first issue we need to address is to choose the correct value of K, that is, how many partitions should we have. One way of selecting K is to first try different values of K and studying the change in the average distance to centroid as K increases. ● Issues in selecting initial points: One of the issues associated with K-means clustering is the initial choice of the K-centroids. Though we have explained that these initial centroids can be chosen at random this may not lead to fast convergence. If there are K clusters then the chance of selecting one centroid from each cluster is small and the chance becomes even lower when K is large. If we assume that clusters are of same size, n, that is each cluster has n data points then the number of ways of selecting K centroids is large and the probability that we choose the K across clusters is small. ● Stopping/ Convergence Criterion: The stopping criteria can be defined in any one of the following ways. We can stop the iteration when there are no or a minimum number of re-assignments of data points to different clusters. We can also define the stopping criteria to be when there is no (or minimum) change of centroids, or in terms of error as minimum decrease in the sum of squared error (SSE). For each point, the error is the distance to the nearest cluster:

Unsupervised Learning where, C j is the jth cluster, m j is the centroid of cluster C j (the mean vector of all the data points in C j), and dist(x, m j) is the distance between data point x and centroid m j . Given two clusterings (clustering is the set of clusters formed), we can choose the one with the smallest error. One straightforward way to reduce SSE is to increase K, the number of clusters. A good clustering with smaller K can have a lower SSE than a poor clustering with higher K. Strengths of K-means K-means is the most popular clustering algorithm. The major strengths of K- means is that it is easy to understand and implement. It generally has a time complexity of the O(tkn), where
n is the number of data points, K is the number of
clusters, and t is the number of iterations. Since both K and t are small, K-means is considered a linear time algorithm. It terminates at a local optimum if SSE is used. However the global optimum is hard to find due to complexity, which is its weakness. Weaknesses of K-means ● The algorithm is only applicable for data where the concept of mean can be defined. For categorical data, where K-means is called as K-mode, the centroid is represented by most frequent values. The user needs to specify the value of K. ● A different initialization (selection of centroids) may sometimes produce a different clustering. The algorithm itself requires the labeling and interpretation of the clusters to be carried out in subsequent phases. ● The algorithm is sensitive to outliers, where outliers are data points that are very far away from other data points. These outliers could be errors in the data recording or some special data points with very different values. Including these outliers in the calculation of the centroid may affect the whole clustering process. One method to deal with outliers is to remove some data points in the clustering process that are much further away from the centroids than other data points. We may prefer to monitor these possible outliers over a few iterations and then decide to remove them. Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small. Then we assign the rest
Unsupervised Learning of the data points to the clusters by distance or similarity comparison, or classification. 10.4 Similarity Measures As we have already discussed, clustering is the grouping together of “similar” data. Choosing an appropriate (dis)similarity measure is a critical step in clustering. Similarity measure is often described as the inverse of the distance function that is less the distance more is the similarity. There are many distance functions for the different types of data such as numeric data, nominal data etc. Distance measures can also be defined specifically for different applications. 10.4.1 Distance functions for Numeric Attributes In general in the case of numeric attributes distance is denoted as dist(x i , x j), where x i and x j are data points. Please note that these data points can be vectors. The most commonly used distance functions in this context are Euclidean distance and Manhattan (city block) distance. These two distance functions are special cases of Minkowski distance. ● Minkowski Distance The distance between two data points x i and x j is defined as the h th root of the sum of the difference between them in each dimension taken to the power of h, where h is any positive integer. ● Euclidean Distance The Euclidean distance between two data points x i and x j is the square root of the sum of the squares of the difference between them in each dimension. This is one of the most commonly used distance functions for clustering data points with numerical attributes. In the case of Euclidean distance h=2. ● Manhattan Distance This distance is sometimes used because of the reduced computational cost as in this case there is no need for calculation of power and power root functions.
Unsupervised Learning It should be noted that this is not a trivial issue since we need to calculate the distance between each and every data point and the number of data points is usually large. Moreover each data point in turn may be represented by a high dimensional vector which further affects the computational cost. In the case of Manhattan distance h=1. ● Weighted Euclidean distance In the case of weighted Euclidean distance the difference between the data points in each dimension is weighted. Each dimension in the vector representing the points corresponds to different attributes or features so this essentially means that we can weight each feature according to its importance in defining the cluster. Here the sum of the weights w 1 , w 2 ,... w r = 1. ● Chebychev distance This distance is equal to the maximum difference between the values of any one of the attributes and the distance measure is given as: 10.4.2 Distance functions for BinaryAttributes Binary attributes have two values or states but no ordering relationships, e.g., attribute gender has two values male and female. In the case of binary attributes normally a confusion matrix is used where the i th and j th data points are represented as vectors x i and x j . We use a confusion matrix to introduce the distance functions/ measures. Figure 10.4 Confusion Matrix
Unsupervised Learning In the above figure 10.4, ● a corresponds to the number of attributes with the value of 0 for both data points x i and x j. ● b corresponds to 0 for x i and 1 for x j. ● c corresponds to 1 for x i and 0 for x j ● d corresponds to the number of attributes with the value of 1 for both data points x i and x j. The confusion matrix can be used when the binary attribute is symmetric, that is, if both states (0 and 1) have equal importance, and carry the same weights. Then the distance function is the proportion of mismatches of their values: dist(x i , x j) = (b+c)/(a+b+c+d) However sometimes the binary attributes are asymmetric, that is, one of the states is more important than the other. We assume that state 1 represents the important state in which case the Jaccard measure using the confusion matrix can be defined as: JDist(x i , x j) = (b+c)/(a+b+c) For text documents normally we use cosine similarity which is a similarity measure not a distance measure. Cosine similarity is a measure of similarity between two vectors obtained by measuring the cosine of the angle between them (as shown in figure 10.5). The similarity between any two given documents d j and d k , represented as vectors is given as: In this case, w i is a weight probably based on the frequency of words in the documents. Figure 10.5 The Cosine Angle between vectors

Unsupervised Learning The result of the Cosine function is equal to 1 when the angle is 0, and it is less than 1 when the angle is of any other value. As the angle between the vectors decreases, the cosine value approaches 1, that is the two vectors are closer, and the similarity between the documents increases. 10.5 Applications of Clustering In the context of machine learning, clustering is one of the functions that has many interesting applications. ● One of the applications of clustering is for understanding. Understanding is achieved through appropriate grouping. Grouping related documents for browsing, grouping genes and proteins that have similar functionality, or grouping stocks with similar price fluctuations are some examples that help in understanding the commonalities and differences between groups. ● Another use of clustering is in summarization, in other words we reduce the size of large data sets. For example, Google news uses clustering of news articles to help in better presentation of news. In fact by using appropriate features for clustering we can also bring out a personalized presentation. Let us see some real-life examples of clustering. When designing T-shirts, making them to fit each person is too expensive while one-size-fits-all is not a satisfactory policy. We could group people with similar sizes to design "small", "medium" and "large" T-shirts. Example 1: groups people of similar sizes together to make "small", "medium" and "large" T-Shirts. In today's world of online marketing, segmenting customers according to their similarities would help in targeted marketing. Features such as previous products bought, effect of discounts etc.. could be used for such marketing. Another example of clustering is in document clustering where given a collection of text documents, we can cluster them according to their content similarities in order to create a topic hierarchy. As we can see from the varied applications, clustering is one of the most utilized data mining techniques. In image processing it is used to cluster images based on their visual content. In the web scenario it is used to cluster groups of users based on their access patterns on webpages or cluster searchers based on
Unsupervised Learning their search behavior or to cluster web pages based on their content and links. In bioinformatics, clustering can be used to group similar proteins based on similarity of their chemical structure and/or functionality. It has been used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc. Due to the large increase of online documents, text clustering is now becoming very important. 10.6 Summary ● Clustering can be defined as the method of organizing data instances into groups based on their similarity. ●

An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. ●

An outlier is a data point that is isolated from all other data points. ● In general clustering deals with high dimensional data. Another important aspect that influences the effectiveness of clustering is the choice of distance function or the similarity or dissimilarity measure used. ● In partitional clustering the data points are divided into a finite number of partitions which are disjoint subsets of the set of data points. ● The K-means algorithm is a heuristic method where each cluster is represented by the center of the cluster and the algorithm converges when the centroids of the clusters do not change. ● Similarity measure is often described as the inverse of the distance function that is less the distance more is the similarity. ● Clustering is used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc. 10.7 Key Terms ● Inter-Cluster Distance: The distance between the data point with the cluster center. ● Intra-Cluster Distance: The distance between the data point of one cluster with the other data point in the other cluster. ● Centroid: The centroid is the centre point of the object. ● K-mode: For categorical data, where K-means is called as K-mode -the centroid
Unsupervised Learning is represented by most frequent values. ● Cosine similarity: It is a measure of similarity between two vectors obtained by measuring the cosine of the angle between them. 10.8 Check Your Progress Short- Answer Type Q1) Binary attributes have two values or states but no ordering relationships. (True/ False?) Q2) __________ is described as the inverse of the distance function that is less the distance more is the similarity. Q3) Which of the following is TRUE about the outliers in the K-means algorithm? a) The algorithm is sensitive to outliers. b) One method to deal with outliers is to remove some data points in the clustering process that are much further away from the centroids than other data points. c) Random Sampling can be used to deal with outliers. d) All of the above Q4) Selecting correct initial points and centroids is one of the requirements in the K- means algorithm. (True/ False?) Q5) In _________ clustering, each data point is assigned to exactly one subset. Long-Answer Type Q1) What are the steps of the K-means algorithm? Also write the algorithm. Q2) Discuss the issues, strengths and weaknesses of the K-means algorithm. Q3) What are the basic features of Clustering? Q4) Explain the two categories of problems in the unsupervised learning algorithm. Q5) Write a short note on Similarity measures in clustering. References: Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004. Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz and Shai Ben-David, Cambridge University Press, 2014.
Unsupervised Learning Unit 11 – Clustering Methods Structure 11.0 Introduction 11.1 Unit Objectives 11.2 Hierarchical Clustering 11.2.1 Agglomerative Clustering 11.2.2 Divisive Clustering 11.3 Spectral Clustering 11.4 Density-based Clustering 11.5 A High Level View of Clustering 11.6 Summary 11.7 Key Terms 11.8 Check Your Progress 11.0 Introduction In the previous unit, we have discussed the basic introduction to Clustering. It is also mentioned that the basic clustering algorithms can be divided into two types namely hierarchical clustering and partitional clustering. We have already discussed the most popular partitional clustering algorithm, that is, k-means clustering algorithm. Hierarchical clustering

is a method of cluster analysis which seeks to build a hierarchy of clusters (or groups) in a given dataset. The hierarchical clustering produces clusters in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level. At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all of the data. The decision regarding whether two clusters are to be merged or not is taken based on the measure of dissimilarity between the clusters. The distance between two clusters is usually taken as the measure of dissimilarity between the clusters. The basic method of clustering is the hierarchical method which is again categorized in two types: agglomerative and divisive. Agglomerative clustering is a bottom up method where initially we assume that each data point is by itself a cluster. Then we repeatedly combine the two "nearest" clusters into one. On the other hand, divisive clustering is a top down procedure where we start with one cluster and recursively
Unsupervised Learning split the clusters until no more division is possible. We normally carry out point assignments where we maintain a set of clusters and allocate points to the nearest cluster. In this unit, we will illustrate hierarchical clustering in detail. The types of hierarchical clustering is also discussed. Another important concept of clustering is demonstrated in this unit, that is, cluster validation and analysis. 11.1 Unit Objectives After completing this unit, the reader will be able to: ● Illustrate the basic features of hierarchical clustering. ● Discuss the agglomerative and divisive clustering in detail. ● Gain knowledge about cluster validation and analysis. 11.2 Hierarchical Clustering
Hierarchical clustering can be represented by a rooted binary tree. The nodes of the trees represent groups or clusters. The root node represents the entire data set. The terminal nodes each represent one of the individual observations (singleton clusters). Each nonterminal node has two daughter nodes. The distance between merged clusters is monotone increasing with the level of the merger. The height of each node above the level of the terminal nodes in the tree is proportional to the value of the distance between its two daughters.
In the hierarchical clustering approach, we carry out partitioning of the dataset in a sequential manner. The approach constructs nested partitions layer by layer by grouping objects into a tree of clusters. In this context there is no need to know the number of clusters in advance. In general the distance matrix is used as the clustering criteria. One potential disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters K. Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of K. Hierarchical clustering has an added advantage over K-means clustering in that it results in an attractive tree-based representation of the observations, called a dendrogram. Dendrograms are commonly used in computational biology to illustrate the clustering of genes or samples.
Unsupervised Learning
A dendrogram is a tree diagram used to illustrate the arrangement of the clusters produced by hierarchical clustering. The dendrogram may be drawn with the root node at the top and the branches growing vertically downwards (see figure 11.2 (a)). It may also be drawn with the root node at the left and the branches growing horizontally rightwards (see figure 11.2 (b)). In some contexts, the opposite directions may also be more appropriate.
For
example, figure 11.1 is a dendrogram of the dataset {a, b, c, d, e}. Note that the root node represents the entire dataset and the terminal nodes represent the individual observations. However, the dendrograms are presented in a simplified format in which only the terminal nodes (that is, the nodes representing the singleton clusters) are explicitly displayed. Figure 11.2 shows the simplified format of the dendrogram in Figure 11.1. Figure 11.3 shows the distances of the clusters at the various levels. Note that the clusters are at 4 levels. The distance between the clusters {a} and {b} is 15, between {c} and {d} is 7.5, between {
c, d} and {e} is 15 and between {a, b} and {c, d, e} is 25.
Figure 11.1 A dendrogram of the dataset {a, b, c, d, e}
Unsupervised Learning Figure 11.2 Different ways of drawing dendrogram Figure 11.3 A dendrogram of the dataset {a, b, c, d, e} showing the distances (heights) of the clusters at different levels
Hierarchical clustering methods can be further classified as either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom- up (merging) or top-down (splitting) fashion. As we have discussed the hierarchical approach sequentially partitions the data points and constructs a tree of clusters. The following are two sequential clustering strategies for constructing the tree of clusters. The important issues in both cases are cluster distance to be considered and the termination condition to be used. Now an important criteria for merging clusters is the cluster distance. There are three different ways in which this cluster distance can be defined. ● Single Link: In this case the distance between two clusters is the distance between the closest points in the clusters. This method is also called nearest neighbor clustering. The cluster distance d(C i , C j) between the clusters C i and C j in the case of single link is given as minimum distance between the data points x ip and x jq in the two clusters. d(C i , C j) = min {d(x ip , x jq)} ● Average Link: In this case the distance between two clusters is the distance
Unsupervised Learning between the cluster centroids.The cluster distance d(C i , C j) between the clusters C i and C j in the case of single link is given as average distance between the data points x ip and x jq in the two clusters. d(C i , C j) = avg {d(x ip , x jq)} ● Complete Link: In this case the distance between two clusters is the distance between the farthest pair of points. The cluster distance d(C i , C j) between the clusters C i and C j in the case of single link is given as minimum distance between the data points x ip and x jq in the two clusters.
The method is also known as farthest neighbour clustering. d(

C i , C j) = max {d(x ip , x jq)} Figure 11.4 Evaluation of Cluster Measures 11.2.1 Agglomerative Clustering Agglomerative clustering is a bottom-up strategy where initially each data point forms its own (atomic) cluster. We then merge these atomic clusters into larger and larger clusters based on some distance metric. The algorithm terminates when all the data
Unsupervised Learning points are in a single cluster or merging is continued until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this category. In the agglomerative clustering,
we start at the bottom and at each level recursively merge a selected pair of clusters into a single cluster. This produces a grouping at the next higher level with one less cluster. If there are N observations in the dataset, there will be N −1 levels in the hierarchy. The pair chosen for merging consist of the two groups with the smallest "intergroup dissimilarity". For example, the hierarchical clustering shown in figure 11.1 can be constructed by the agglomerative method as shown in figure 11.5. Each non-terminal
node has two daughter nodes. The daughters represent the two groups
that were merged to form the parent. Figure 11.5 Hierarchical clustering using Agglomerative method Algorithm for Agglomerative hierarchical clustering
Unsupervised Learning
Given a set of N items to be clustered and an N × N distance matrix, required to construct a hierarchical clustering of the data using the agglomerative method. Step 1. Start by assigning each item to its own cluster, so that we have N clusters, each containing just one item. Let the distances between the clusters equal the distances between the items they contain. Step 2. Find the closest pair of clusters and merge them into a single cluster, so that now we have one less cluster. Step 3. Compute distances between the new cluster and each of the old clusters. Step 4. Repeat Steps 2 and 3 until all items are clustered into a single cluster of size N. For example, Given the dataset {a, b, c, d, e} and the following distance matrix, construct a dendrogram by complete linkage hierarchical clustering using the agglomerative method. Table 11.1 Example for distance matrix Solution: The complete-linkage clustering uses the "maximum formula", that is, the following formula to compute the distance between two clusters A and B: d(A, B) = max {d(x, y) ∶ x ∈ A, y ∈ B} 1. Dataset: {a, b, c, d, e} Initial clustering (singleton sets) C 1 : {a}, {b}, {c}, {d}, {e}. 2. The following table gives the distances between the various clusters in C 1 : In the above table, the minimum distance is the distance between the clusters {c} and {e}. Also d({c}, {e}) = 2. We merge {c} and {e} to form the cluster {c, e}.
Unsupervised Learning
The
new set of clusters C 2 : {a}, {b}, {d}, {c, e}. 3. Let us compute the distance of {c, e} from other clusters. d({c, e}, {a}) = max {d(c, a), d(e, a)} = max {3, 11} = 11. d({c, e}, {b}) = max {d(c, b), d(e, b)} = max {7, 10} = 10. d({c, e}, {d}) = max {d(c, d), d(e, d)} = max {9, 8} = 9. The following table gives the distances between the various clusters in
C 2 .
In the above table, the minimum distance is the distance between the clusters {b} and {d}. Also d({b}, {d}) = 5. We merge {b} and {d} to form the cluster {b, d}. The new set of clusters C 3 : {a}, {b, d}, {c, e}. 4. Let us compute the distance of {b, d} from other clusters. d({b, d}, {a}) = max {d(b, a), d(d, a)} = max {9, 6} = 9. d({b, d}, {c, e}) = max {d(b, c), d(b, e), d(d, c), d(d, e)} = max {7, 10, 9, 8} = 10. The following table gives the distances between the various clusters in C 3 . In the above table, the minimum distance is the distance between the clusters {a} and {b, d}. Also d({a}, {b, d}) = 9. We merge {a} and {b, d} to form the cluster {a, b, d}. The new set of clusters C 4 : {a, b, d}, {c, e} 5. Only two clusters are left. We merge them form a single cluster containing all data points. We have d({a, b, d}, {c, e}) = max {d(a, c), d(a, e), d(b, c), d(b, e), d(d, c), d(d, e)} =
max{3, 11, 7, 10, 9, 8}
Unsupervised Learning = 11
Figure 11.6 Dendrogram for the data given in Table 11.1 (complete linkage clustering) 11.2.2
Divisive Clustering Divisive clustering
is a top-down strategy which does the reverse of agglomerative hierarchical clustering where initially all data points together form a single cluster. Then the cluster is subdivided into smaller and smaller clusters based on some distance metric. It subdivides the clusters into smaller and smaller pieces, until it satisfies certain termination conditions, such as a desired number of clusters or the diameter of each cluster is within a certain threshold.
The divisive method starts at the top and at each level recursively split one of the existing clusters at that level into two new clusters. If there are N observations in the dataset, the divisive method also will produce N – 1 levels in the hierarchy. The split is chosen to produce two new groups with the largest "between-group dissimilarity". For example, the hierarchical clustering shown in Figure 11.1 can be constructed by the divisive method as shown in Figure 11.7. Each nonterminal node has two daughter nodes. The two daughters represent the two groups resulting from the split of the parent.
Unsupervised Learning Figure 11.7 Hierarchical clustering using Divisive method
Algorithm for Divisive hierarchical clustering Divisive clustering algorithms begin with the entire data set as a single cluster, and recursively divide one of the existing clusters into two daughter clusters at each iteration in a top-down fashion. To apply this procedure, we need a separate algorithm to divide a given dataset into two clusters. The divisive algorithm may be implemented by using the k-means algorithm with k = 2 to perform the splits at each iteration. However, it would not necessarily produce a splitting sequence that possesses the monotonicity property required for dendrogram representation.
Unsupervised Learning
Step 1: Suppose that cluster C l is going to be split into clusters C i and C j . Step 2: Let C i = C l and C j = ∅.
Step 3: For each object x ∈ C i :

a) For the first iteration, compute the average distance of x to all other objects. b) For the remaining iterations, compute D x = average {d(x, y) ∶ y ∈
C i } − average {d(x, y) ∶ y ∈ C j }. D x = (
average of dashed lines) − (average of solid lines) Step 4: a) For the first iteration, move the object with the maximum average distance to
C j .
b) For the remaining iterations, find an object x in
C i for which D x is the largest. If D x &lt; 0 then move x to C j . Step 5: Repeat Steps 3(b) and 4(b) until all differences D x are negative. Then C l is split into C i and C j .
Step 6: Select the smaller cluster with the largest diameter. (The diameter of a cluster is the largest dissimilarity between any two of its objects.) Then divide this cluster, following Steps 1-5. Step 7: Repeat Step 6 until all clusters contain only a single object. For
example,
Given the dataset {a, b, c, d, e} and the distance matrix in Table 11.1, construct a dendrogram by the divisive analysis algorithm. Solution: 1. We have, initially
C l = {
a,
b, c, d, e} 2. We write C i = C l , C j = ∅. 3. Division into clusters (
a) Initial iteration Let us calculate the average dissimilarities of the objects in C i with the other objects in
C i .
Unsupervised Learning Average dissimilarity of
a Similarly we have : Average dissimilarity of b = 7.75 Average dissimilarity of c = 5.25 Average dissimilarity of d = 7.00 Average dissimilarity of e = 7.75 The highest average distance is 7.75 and there are two corresponding objects. We choose one of them, b, arbitrarily. We move b to
C j . We now have C i = {a, c, d, e}, C j = ∅ ∪ {b} = {b}. (
b) Remaining iterations (i) 2-nd iteration D
d is the largest and D d &lt; 0. So we move, d to C j . We now have C i = {a, c,
e},
C j = {b} ∪ {d} = {b, d}. (ii) 3-rd iteration All are negative. So we stop and form the clusters
C i and C j . 4. To divide, C i and C j , we compute their diameters. diameter(C i) =
max {d(a, c), d(a, e), d(c, e)}
Unsupervised Learning = max {3, 11, 2} = 11 diameter(C j) = max {d(b, d)} = 5 The cluster with the largest diameter is
C i . So we now split C i . We repeat the process by taking C l = {a, c, e}. 11.3
Spectral Clustering Instead of clustering in the original space, a possibility is to first map the data to a new space with reduced dimensionality such that similarities are made more apparent and then cluster in there. Any feature selection or extraction method can be used for this purpose, and one such method is the Laplacian eigenmaps, where the aim is to place the data instances in such a way that given pairwise similarities are preserved. After such a mapping, points that are similar are placed nearby, and this is expected to enhance the performance of clustering—for example, by using k-means. This is the idea behind spectral clustering (von Luxburg, 2007). There are hence two steps: 1. In the original space, we define a local neighborhood (by either fixing the number of neighbors or a distance threshold), and then for instances that are in the same neighborhood, we define a similarity measure—for example, using the Gaussian kernel—that is inversely proportional to the distance between them. Remember that instances not in the same local neighborhood are assigned a similarity of 0 and hence can be placed anywhere with respect to each other. Given this Laplacian, instances are positioned in the new space using feature embedding. 2. We run k-means clustering with the new data coordinates in this new space. We remember from section 6.12 that when B is the matrix of pairwise similarities and D is the diagonal degree matrix with d i = ∑ j B ij on the diagonals, the graph Laplacian is defined as L = D − B This is the unnormalized Laplacian. There are two ways to normalize. One is closely related to the random walk (Shi and Malik 2000) and the other constructs a symmetric matrix (Ng, Jordan, and Weiss 2002). They may lead to better performance in clustering:

Unsupervised Learning L rw = I – D –1 B L sym = I – D –1/2 BD –1/2 It is always a good idea to do dimensionality reduction before clustering using Euclidean distance if there are redundant or correlated features. Using Laplacian eigenmaps makes more sense than multidimensional scaling proper or principal components analysis because those two check for the preservation of pairwise similarities between all pairs of instances whereas here with Laplacian eigenmaps, we care about preserving the similarity between neighboring instances only and in a manner that is inversely proportional to the distance between them. This has the effect that instances that are nearby in the original space, probably within the same cluster, will be placed very close in the new space, thus making the work of k-means easier, whereas those that are some distance away, probably belonging to different clusters, will be placed far apart. The graph should always be connected; that is, the local neighborhood should be large enough to connect clusters. Remember that the number of eigenvectors with eigenvalue 0 is the number of components and that it should be 1. It should be noted that though similarities are local, they propagate. Consider three instances, a, b, and c. Let us say a and b lie in the same neighborhood and so do b and c, but not a and c. Still, because a and b will be placed nearby and b and c will be placed nearby, a will lie close to c too, and they will probably be assigned to the same cluster. Consider now a and d that are not in the neighborhood with too many intermediate nodes between them; these two will not be placed nearby and it is very unlikely that they will be assigned to the same cluster. 11.4

Density-based clustering In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas, that are required to separate clusters, are usually considered to be noise and border points. The most popular density based clustering method is DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

Unsupervised Learning

Figure 11.8 Clusters of points and noise points not belonging to any of those clusters Some basic terminologies and notations are discussed below: ●

Let (epsilon) be some constant distance. Let p be an arbitrary data point. The -neighbourhood of p is the set N (p) = {q ∶ d (p, q) &gt; } ● We choose some number m 0 to define points of "high density": We say that a point p is point of high density if N (p) contains at least m 0 points. ● We define a point p as a core point if N (p) has more than m 0 points. ● We define a point p as a border point if N (p) has fewer than m 0 points, but is in the -neighbourhood of a core point. ● A point which is neither a core point nor a border point is called a noise point. Figure 11.9 With m 0 = 4: (a) p is a point of high density (b) p
is a core point (c) p is a border point (d) r is
a noise point ● An object q is directly density-reachable from object p if p is a core object and q is in N (p). ● An object q is indirectly density-reachable from an object p if there is a finite set of objects p 1 , . . . , p r such that p 1 is directly density-reachable form p, p 2 is directly density reachable from p 1 , etc., q is directly density-reachable
from p r .

Unsupervised Learning

Figure 11.10 With m 0 = 4: (a) q is directly density-reachable from p (b) q is indirectly density-reachable from p Table 11.2 DBSCAN Algorithm Let X = {x 1 , x 2 , . . . ,
x n }
be the set of data points. DBSCAN requires two parameters: (eps) and the minimum number of points required to form a cluster (m 0). Step 1: Start with an arbitrary starting point p that has not been visited. Step 2: Extract the -neighborhood N (p) of p. Step 3: If the number of points in N (p) is not greater than m 0 then the point p
is labeled as noise (later this point can become the part of the cluster). Step 4: If
the number of points in N (p) is greater than m 0 then the point p is a core point and is
marked as visited. Select a new cluster-id and mark all objects in N (p) with this cluster-id.
Step 5:
If a point is found to be a part of the cluster then its -neighborhood is also the part of the cluster and the above procedure from step 2 is repeated for all - neighborhood points. This is repeated until all points in the cluster are determined.
Step 6:
A new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise. Step 7: This process continues until all points are marked as visited.

One of the advantages of spectral clustering and hierarchical clustering is that we do not need a vectorial representation of the instances, as long as we can define a similarity/distance measure between pairs of instances. The problem of representing an arbitrary data structure: documents, graphs, web pages, and so on, as a vector such that Euclidean distance is meaningful is always a tedious task and leads to artificial representations, such as the bag of words. Being able to use (dis)similarity measures directly defined on the original structure is always a good idea

Unsupervised Learning 11.5 A High Level View of Clustering So far, we have mainly listed various useful clustering tools. However, some fundamental questions remain unaddressed. First and foremost, what is clustering? What is it that distinguishes a clustering algorithm from any arbitrary function that takes an input space and outputs a partition of that space? Are there any basic properties of clustering that are independent of any specific algorithm or task? One method for addressing such questions is via an axiomatic approach. There have been several attempts to provide an axiomatic definition of clustering. Let us demonstrate this approach by presenting the attempt made by (Kleinberg 2003). Consider a clustering function, F, that takes as input any finite domain X with a dissimilarity function d over its pairs and returns a partition of X. Consider the following three properties of such a function: ● Scale Invariance (SI): For any domain set X, dissimilarity function d, and any α &lt; 0, the following should hold: F (X , d) = F (X , αd) (where (αd)(x, y) = α d(x, y)). ● Richness (Ri): For any finite X and every partition C = (C 1 , . . . C k) of X (into nonempty subsets) there exists some dissimilarity function d over X such that F (X , d) = C. ● Consistency (Co): If d and d′ are dissimilarity functions over X, such that for every x, y ∈ X , if x, y belong to the same cluster in F (X , d) then d′ (x, y) ≤ d(x, y) and if x, y belong to different clusters in F (X , d) then d′ (x, y) ≥ d(x, y), then F (X , d) = F (X , d′). A moment of reflection reveals that the Scale Invariance is a very natural requirement – it would be odd to have the result of a clustering function depend on the units used to measure between-point distances. The Richness requirement basically states that the outcome of the clustering function is fully controlled by the function d, which is also a very intuitive feature. The third requirement, Consistency, is the only requirement that refers to the basic (informal) definition of clustering – we wish that similar points will be clustered together and that dissimilar points will be separated to different clusters, and therefore, if points that already share a cluster become more similar, and points that are already separated become even less similar to each other, the clustering function should have even stronger "support" of its previous clustering decisions.
Unsupervised Learning 11.6 Summary ● Hierarchical clustering
is a method of cluster analysis which seeks to build a hierarchy of clusters (or groups) in a given dataset. ●
The
distance between two clusters is usually taken as the measure of dissimilarity between the clusters. ●
The basic method of clustering is the hierarchical method which is again categorized in two types: agglomerative and divisive. ●
A dendrogram is a tree diagram used to illustrate the arrangement of the clusters produced by hierarchical clustering. ●
After such a mapping, points that are similar are placed nearby, and this is expected to enhance the performance of clustering—for example, by using k- means. This is the idea behind spectral
clustering. ● In density-based clustering, clusters are defined as areas of higher density than the remainder of the data set. 11.7 Key Terms ● Core Point:
We define a point p as a core point if N (p) has more than m 0 points. ● Noise Point: A point which is neither a core point nor a border point is called a noise point. ●
Dendrogram:
A
dendrogram is a tree diagram used to illustrate the arrangement of the clusters produced by hierarchical clustering. ●
Divisive clustering: It is a top-down strategy which does the reverse of agglomerative hierarchical clustering where initially all data points together form a single cluster. 11.8 Check Your Progress Short-Answer Type Q1) Agglomerative clustering is a _________ strategy while Divisive clustering is a ______ strategy. Q2) In how many ways the cluster distance can be defined? Q3) The Complete link method of cluster distance is also known as _________. Q4) Laplacian eigenmaps are used in _______.
Unsupervised Learning Q5) In divisive clustering, initially all data points together form a single cluster. (True/ False?) Long-Answer Type Q1) Explain hierarchical clustering. Q2)
What is a dendrogram? Give an example. Q3) Describe the two methods for hierarchical clustering
in detail. Q4)
In the context of density-based clustering, define high density point, core point, border point and noise point.
Q5) Explain the different types of linkages in clustering. References: Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004. Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz and Shai Ben-David, Cambridge University Press, 2014.
Unsupervised Learning Unit 12 – Clustering Validation Structure 12.0 Introduction 12.1 Unit Objectives 12.2 Cluster Validity 12.3 Cluster Validation Process 12.4 Measures of Cluster Validation 12.5 Internal Measures 12.5.1 Sum of Squares 12.5.2 Silhouette Coefficient 12.6 Trends in Clustering Process 12.7 Summary 12.8 Key Terms 12.9 Check Your Progress 12.0 Introduction Cluster analysis is one of the major tasks in various research areas. However, it may be found under different names in different contexts such as unsupervised learning in pattern recognition, taxonomy in biology, partition in graph theory. The clustering aims at identifying and extracting significant groups in underlying data. Thus based on a certain clustering criterion the data are grouped so that data points in a cluster are more similar to each other than points in different clusters. For cluster analysis, the question is how to evaluate the goodness of the resulting clusters. But as it is said the clusters can be defined from different perspectives. However validation is needed to compare clustering algorithms, solve the problem of determining the number of clusters, comparing two clusters, comparing two sets of clusters, and for avoiding finding patterns in noise. "The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage." -Jain & Dubes We will discuss cluster validation and its features in the upcoming sections of this

Unsupervised Learning unit. The process and measures for cluster validation is also illustrated. 12.1 Unit Objectives After completing this unit, the reader will be able to: ● Understand the Cluster Validity and Cluster Validation Process ● Discuss the different Measures of Cluster Validity ● Illustrate Purity based and Internal measures for determining Cluster Validity. 12.2 Cluster Validation One of the most important issues in cluster analysis is the evaluation of clustering results to find the partitioning that best fits the underlying data. The objective of the clustering methods is to discover significant groups present in a data set. In general, they should search for clusters whose members are close to each other (in other words have a high degree of similarity) and well separated. A problem we face in clustering is to decide the optimal number of clusters that fits a data set. In most algorithms' experimental evaluations 2D-data sets are used in order that the reader is able to visually verify the validity of the results (i.e., how well the clustering algorithm discovered the clusters of the data set). It is clear that visualization of the data set is a crucial verification of the clustering results. In the case of large multidimensional data sets (e.g. more than three dimensions) effective visualization of the data set would be difficult. Moreover the perception of clusters using available visualization tools is a difficult task for humans that are not accustomed to higher dimensional spaces. The various clustering algorithms behave in a different way depending on: a) the features of the data set (geometry and density distribution of clusters), b) the input parameters values The procedure of evaluating the results of a clustering algorithm is known under the term cluster validity. In general terms, there are three approaches to investigate cluster validity: 1. The first is based on external criteria. This implies that we evaluate the results of a clustering algorithm based on a pre-specified structure, which is imposed on a data set and reflects our intuition about the clustering structure of the data set.
Unsupervised Learning 2. The second approach is based on internal criteria. We may evaluate the results of a clustering algorithm in terms of quantities that involve the vectors of the data set themselves (e.g. proximity matrix). 3. The third approach of clustering validity is based on relative criteria. Here the basic idea is the evaluation of a clustering structure by comparing it to other clustering schemes, resulting by the same algorithm but with different parameter values. There are two criteria proposed for clustering evaluation and selection of an optimal clustering scheme (Berry and Linoff, 1996): ● Compactness: The members of each cluster should be as close to each other as possible. A common measure of compactness is the variance, which should be minimized. ● Separation: The clusters themselves should be widely spaced. There are three common approaches measuring the distance between two different clusters: a) Single linkage: It measures the distance between the closest members of the clusters. b) Complete linkage: It measures the distance between the most distant members. c) Comparison of centroids: It measures the distance between the centers of the clusters. The two first approaches are based on statistical tests and their major drawback is their high computational cost. Moreover, the indices related to these approaches aim at measuring the degree to which a data set confirms an a-priori specified scheme. On the other hand, the third approach aims at finding the best clustering scheme that a clustering algorithm can be defined under certain assumptions and parameters. 12.3 Cluster Validation Process Cluster validation involves evaluation of the clustering using an external index by comparing the clustering results to ground truth (externally known results). Evaluation of the quality of clusters without reference to external information using only the data is called as evaluation using internal index. Another aspect is determining the reliability of clusters, that is the confidence level that the clusters
Unsupervised Learning are not formed by chance, normally determined using a statistical framework. Now let us discuss the process of cluster validation. The following are the steps involved: 1. The first step is to find out whether the set of data has the clustering tendency that distinguishes whether non-random structures actually exist in the data or whether the set of data is actually one cluster. 2. The next step is the comparison of the results of a cluster analysis to the externally known results, that is to externally given class labels. 3. The third step is the evaluation of how well the results of a cluster analysis fit the data without reference to external information. Use only the data. 4. Then we carry out the comparison of the results of two different sets of cluster analyses to determine which is better. 5. Finally we have the important task of determining the correct number of clusters for the given data set. In cluster validation, the steps 2, 3, and 4, can be further distinguished depending on whether we want to evaluate the entire clustering or just individual clusters. According to Jain & Dubes, cluster validation refers to procedures that evaluate the results of clustering in a quantitative and objective fashion. In this context evaluation is quantitative when we employ the measures for evaluation while it is objective when we validate of the measures. Figure 12.1 shows that for a given data set X, we can apply the clustering algorithm and given the gold standard in terms of different partitions P and the associated codebook C, determine the validity index. The process is repeated by trying out with different number of clusters. Figure 12.1 Clustering and Evaluation 12.4 Measures of Cluster Validation As we have already discussed, the external index is where we validate against ground
Unsupervised Learning truth, or compare two clusters to find out the similarity between them. Figure 12.2 also illustrates the determination of the internal index where we validate without any external information, and if we do the same with different number of clusters we can solve the problem of determining the number of clusters. Figure 12.2 External and Internal Index For the comparison with Ground truth, let the notation for comparing with ground truth be as follows: N: number of objects in the data set; P = {P 1 ,..., P m }: the set of "ground truth" clusters; C = {C 1 ,..., C n }: the set of clusters reported by a clustering algorithm. Now we create the incidence matrix, which is a N x N matrix where both the rows and columns correspond to objects. P ij = 1 if object O i and object O j belong to the same ground truth cluster in P; P ij = 0 otherwise. Similarly C ij = 1 if object O i and object O j belong to the same cluster in C; C ij =0 otherwise. A pair of data object (O i , O j) falls into one of the following categories: ● SS: C ij =1 and P ij = 1; (agree) ● DD: C ij =0 and P ij =0; (agree) ● SD: C ij =1 and P ij =0; (disagree) ● DS: C ij =0 and P ij =1; (disagree) Now we define two evaluation measures based on this incidence matrix, Rand index and Jaccard coefficient. Rand Index may be dominated by DD, which is about objects not belonging to the same cluster in either the ground truth case or in the clusters obtained through the clustering algorithm.

Unsupervised Learning 12.5 Internal Measures As discussed previously internal measures are used to measure the goodness of a clustering structure without comparing with external information. There are basically two aspects that are considered namely cohesion which measures how closely related the data points in a cluster are and separation which measures how distinct or well- separated the data points of a cluster are from the data points of other clusters (Figure 12.3). These aspects can be measured using sum of squares and sum of squared error. Figure 12.3 Cohesion and Separation 12.5.1 Internal Measure: Sum of Squares When we measure cluster cohesion, we measure homogeneity measured by the within cluster sum of squares. Cohesion is measured by the within-cluster sum of squares: (Exactly the objective function of K-means) Separation is measured by the between-cluster sum of squares: Where, |C i | is the size of cluster i, m is the centroid of the whole data set. The sum of BSS + WSS = constant. In general, a larger number of clusters tend to result in smaller WSS as shown in the example of figure 12.4.
Unsupervised Learning Figure 12.4 Example of WSS an BSS with Different Number of Clusters 12.5.2 Internal Measure: Silhouette Coefficient Silhouette Coefficient combines ideas of both cohesion and separation. The silhouette coefficient contrasts the average distance to elements in the same cluster with the average distance to elements in other clusters. Objects with a high silhouette value are considered well clustered, objects with a low value may be outliers. This index works well with k-means clustering, and is also used to determine the optimal number of clusters. Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1. ● 1 (Good): Means clusters are well apart from each other and clearly distinguished. ● 0 (Indifferent): Means clusters are indifferent, or we can say that the distance between clusters is not significant. ● -1 (Bad): Means clusters are assigned in the wrong way. Where, a(x) is the average distance of x to all other vectors in the same cluster or
Unsupervised Learning average intra-cluster distance. b(x) is the average distance of x to the vectors in other clusters or average inter-cluster distance. s(x) is the Silhouette Coefficient. For all N data points, Silhouette coefficient (SC) will be defined as: 12.6 Trends in Clustering Process Though cluster analysis is subject of thorough research for many years and in a variety of disciplines, there are still several open research issues. We summarize some of the most interesting trends in clustering as follows: ● Discovering and finding representatives of arbitrary shaped clusters One of the requirements in clustering is the handling of arbitrary shaped clusters and there are some efforts in this context. However, there is no well- established method to describe the structure of arbitrary shaped clusters as defined by an algorithm. Considering that clustering is a major tool for data reduction, it is important to find the appropriate representatives of the clusters
Unsupervised Learning describing their shape. Thus, we may effectively describe the underlying data based on clustering results while we achieve a significant compression of the huge amount of stored data (data reduction). ● Non-point clustering The vast majority of algorithms have only considered point objects, though in many cases we have to handle sets of extended objects such as hyper- rectangles. Thus, a method that efficiently handles sets of non-point objects and discovers the inherent clusters presented in them is a subject of further research with applications in diverse domains (such as spatial databases, medicine, biology). ● Handling uncertainty in the clustering process and visualization of results The majority of clustering techniques assumes that the limits of clusters are crisp. Thus each data point may be classified into at most one cluster. Moreover all points classified into a cluster belong to it with the same degree of belief (i.e., all values are treated equally in the clustering process). The result is that, in some cases “interesting” data points fall out of the cluster limits so they are not classified at all. This is unlikely to be an everyday life experience where a value may be classified into more than one category. Thus a further work direction is taking in account the uncertainty inherent in the data. Another interesting direction is the study of techniques that efficiently visualize multidimensional clusters taking also in account uncertainty features. ● Incremental clustering The clusters in a data set may change as insertions/ updates and deletions occur throughout its life cycle. Then it is clear that there is a need to evaluate the clustering scheme defined for a data set so as to update it in a timely manner. However, it is important to exploit the information hidden in the earlier clustering schemes so as to update them in an incremental way. ● Constraint-based clustering Depending on the application domain we may consider different clustering aspects as more significant. It may be important to stress or ignore some aspects of data according to the requirements of the considered application. In recent years, there is a trend that cluster analysis is based on less parameters
Unsupervised Learning but on more constraints. These constraints may exist in data space or in users’ queries. Then a clustering process has to be defined so as to take in account these constraints and define the inherent clusters fitting a dataset. 12.7 Summary ● For cluster analysis, the question is how to evaluate the goodness of the resulting clusters. ● One of the most important issues in cluster analysis is the evaluation of clustering results to find the partitioning that best fits the underlying data. ● The procedure of evaluating the results of a clustering algorithm is known under the term cluster validity. ● Evaluation of the quality of clusters without reference to external information using only the data is called as evaluation using internal index. ● Internal measures are used to measure the goodness of a clustering structure without comparing with external information. 12.8 Key Terms ● Cohesion: It measures how closely related the data points in a cluster are. ● Separation: It measures how distinct or well-separated the data points of a cluster are from the data points of other clusters. ● Silhouette Coefficient or silhouette score: It is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1. 12.9 Check Your Progress Short-Answer Type Q1) Determining the correct number of clusters for the given data set is the _______ step of Cluster Validation Process. Q2) According to Jain & Dubes, cluster validation refers to procedures that evaluate the results of clustering in a _________ and _______ fashion. Q3) The procedure of evaluating the results of a clustering algorithm is known as ________. Q4) Silhouette Coefficient combines ideas of both cohesion and separation. (True/

Unsupervised Learning False?) Long-Answer Type Q1) Silhouette Coefficient combines ideas of both cohesion and separation. Elaborate. Q2) What are the trends in the Clustering Process? Q3) Define Cohesion and Separation. How do they affect the clustering process? Q4) What is Cluster Validation? Explain with an example. Q5) Discuss the different measures of Cluster Validation. References: On Clustering Validation Techniques, Maria Halkidi and Michalis Vazirgiannis, Journal of Intelligent Information Systems (2001). Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004. Advances in Machine Learning and Data Science, Damodar Reddy Edla, Pawan Lingras, Venkata Naresh Babu K, Springer, 2018. Unsupervised Learning Unit 13 – Dimensionality Reduction Structure 13.0 Introduction 13.1 Unit Objectives 13.2 Curse of Dimensionality 13.3 Dimensionality Reduction 13.4 Feature Extraction and Feature Selection 13.4.1 Feature Selection 13.4.2 Feature Extraction 13.5 Principal Component Analysis (PCA) 13.6 Multidimensional Scaling 13.7 Summary 13.8 Key Terms 13.9 Check Your Progress 13.0 Introduction In an application, whether it is classification or regression, observation data that we believe contain information are taken as inputs and fed to the system for decision making. Ideally, we should not need feature selection or extraction as a separate process; the classifier (or regressor) should be able to use whichever features are necessary, discarding the irrelevant. However,
there are several reasons why we are interested in reducing dimensionality
as a separate preprocessing step: ●
In most learning algorithms, the complexity depends on the number of input dimensions, d, as well as on the size of the data sample, N, and for reduced memory and computation, we are interested in reducing the dimensionality of the problem. Decreasing d also decreases the complexity of the inference algorithm during testing. ● When an input is decided to be unnecessary, we save the cost of extracting it. ● Simpler models are more robust on small datasets. Simpler models have less variance, that is, they vary less depending on the particulars of a sample, including noise, outliers, and so forth.
Unsupervised Learning ●
When data can be explained with fewer features, we get a better idea about the process that underlies the data and this allows knowledge extraction.
These fewer features may be interpreted as hidden or latent factors that in combination generate the observed features. ●
When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers.
This unit focuses on the need for dimensionality reduction. The methods of feature reduction and feature selection are also discussed. The basic concept of Principal Component Analysis (PCA) and Multidimensional Scaling are explained briefly. 13.1 Unit Objectives After completing this unit, the reader will be able to: ● Explain the Curse of Dimensionality ● Discuss the concept of Dimensionality Reduction ● Outline the concepts of feature selection and feature reduction ● Discuss in detail the Principal Component Analysis method of Dimensionality Reduction. ● Illustrate the multidimensional scaling. 13.2 Curse of Dimensionality In any machine learning problem, if the number of observables or features is increased then it takes more time to compute, more memory to store inputs and intermediate results and more importantly much more data samples are needed for the learning. From a theoretical point of view, increasing the number of features should lead to better performance. However in practice, the inclusion of more features leads to a decrease in performance. This aspect is called the curse of dimensionality and is basically because the number of training examples required increases exponentially as dimensionality increases. A lot of machine learning methods have at least O(nd 2) complexity, where n is the number of samples and d is the dimensionality. A typical example is the need to estimate covariance matrix for which as d becomes large the number of samples is of the O(nd 2) which involves huge computations. In other words if the number of features that is dimension d is large, the number of samples n, may be too small for accurate
Unsupervised Learning parameter estimation. For example, the covariance matrix has d 2 parameters: If the model parameters are high then there will be overfitting. An interesting paradox is that if n&gt;d 2 we are better off assuming that features are uncorrelated, even if we know this assumption is wrong. We are likely to avoid overfitting because we fit a model with less parameters. Figure 13.1 Model with more and less number of parameters Let us suppose, we want to use the nearest neighbour approach with k = 1 (1NN). Suppose we start with only one feature, this feature is not discriminative, i.e. it does not separate the classes well. Now we decide to use 2 features. For the 1NN method to work well, we need a lot of samples, i.e. samples have to be dense. To maintain the same density as in 1D (9 samples per unit length), how many samples do we need? As we discussed we need 9 2 samples to maintain the same density as in 1D. Of course, when we go from 1 feature to 2, no one gives us more samples, we still have 9. This is way too sparse for 1NN to work well. Things are even more of a problem when we decide to use 3 features. If 9 was dense enough in 1D, in 3D we need 9 3 =729 samples. In general, if n samples are dense enough in 1D - Then in d dimensions we need n d samples. (i)
Unsupervised Learning (ii) (iii) Figure 13.2 (i) 1NN with only One Feature, (ii) 1NN with only Two Features, (iii) 1NN with Three Dimensions Some features (dimensions) bear little useful information, which essentially means that we can drop some features. In dimensionality reduction high-dimensional points are projected to a low dimensional space while preserving the “essence” of the data. In projecting from a higher dimensional space to a lower dimensional space the distances are preserved as well as possible. After this projection the learning problems are solved in low dimensions. Let us assume that we have data with dimension d. Let us reduce the dimensionality to k&gt;d by discarding unimportant features or combining several features in one and then use the resulting k-dimensional data set for learning for classification problem (e.g. parameters of probabilities P(x|C) or learning for regression problems (e.g. parameters for model y = g(x|Thetha)). For a fixed number of samples, as we add features, the graph of classification error is shown in Figure 13.3. We see that there exists an optimal number of features which

Unsupervised Learning results in minimum classification error. Figure 13.3 Optimal Number of Features for Classification Thus for each fixed sample size n, there is an optimal number of features that we should use. In dimensionality reduction we strive to find that set of features that are effective. 13.3 Dimensionality Reduction Most machine learning and data mining techniques may not be effective for high dimensional data since the intrinsic dimension (that is the actual features that decide the classification) may be small, for example the number of genes actually responsible for a disease may be small but the dataset may contain a large number of other genes as features. The dimension-reduced data can be used for visualizing, exploring and understanding the data. In addition cleaning the data will allow simpler models to be built later. Need for Dimensionality Reduction ● Essentially, dimensionality reduction reduces time complexity since there will be less computation and therefore more efficient learning. ● Dimensionality reduction also reduces space complexity since there are less number of parameters resulting in more efficient storage. Moreover simpler models are more robust on small datasets since complex models may result in overfitting especially in the case of smaller datasets. ● The dimensionality reduction helps in data visualization (structure, groups, outliers, etc) specifically when reduction is to 2 or 3 dimensions. Criterion for Reduction There are many criteria that can be used for dimensionality reduction. These include criteria that are mainly geometric based and information theory based. These criteria
Unsupervised Learning need to capture the variation in data since these variations are "signals" or information contained in the data. We need to normalize each variable first and then discover variables or dimensions that are highly correlated or dependent. When variables are highly related they can be combined to form a simpler representation. Process of
Dimensionality Reduction The process of reducing the number of random variables (
features) used to represent the samples under consideration can be carried out by combining, transforming or selecting features. We have to estimate which features can be removed from the data. Several features can be combined together without loss or even with gain of information (e.g. income of all family members for loan application), however we need to estimate which features to combine from the data. The simplest way to carry out dimensionality reduction is to keep just one variable and discard all others. However this is too simplistic and not reasonable. Another simple way is to weigh all variables equally but again this is not reasonable unless all variables have the same variance. Another method is to weigh the features based on some criteria and find the weighted average. However the issue is the choice of the criterion. The basic issues for dimensionality reduction are two fold namely how do we represent the data, whether we use vector space and what is the criteria to be used in carrying out the reduction process. Dimensionality can be reduced basically using two methods: feature extraction and feature selection. We will discuss these methods in the upcoming section. 13.4 Feature Extraction and Feature Selection Dimensionality reduction techniques transform the original dataset having high dimensionality and turn it into a new dataset representing low dimensionality while maintaining as much as possible the original meanings of the data. Generally, dimensionality reduction techniques can be classified into two main groups. In feature selection, information can be lost since some features should be excluded when the process of feature subset choice by doing this information can be reduced. However, in feature extraction, the dimension can be decreased without losing much initial feature dataset. Table 13.1 provides a descriptive summary of the methods of dimension reduction.
Unsupervised Learning Table 13.1 Summary of Dimensionality Reduction Techniques Method Main Concept Pros Cons Feature Extraction Summarize the dataset by creating linear combinations of the features Preserves the original, relative distance between covers latent structure, objects Not sufficient enough in the existing of a huge number of irrelevant features Feature Selection A sublist of relevant features can be selected depending on defined criteria Strong against irrelevant features Latent structure does not cover 13.4.1 Feature Selection Feature selection is utilized to reduce the dimensionality impact on the dataset through finding the subset of features which efficiently define the data. It selects the important and relevant features to the mining task from the input data and removes redundant and irrelevant features. It is useful for detecting a good subset of features that is appropriate for the given problem. The main purpose of feature selection is to construct a subset of features as small as possible but represent the whole input data vital features. Feature selection provides numerous advantages: ● Reduce the size of data ● Decrease needed storage, ● Prediction accuracy improvement ● Overfitting evading ● Reduce executing and training time from easily understanding variables. Feature selection algorithm phase is divided into two-phase such as 1. Subset Generation 2. Subset Evaluation In subset Generation, we need to generate subset from the input dataset and to use Subset Evaluations we have to check whether the generated subset is optimal or not. Figure 13.4 shows the overall method of the feature selection process.
Unsupervised Learning Figure 13.4 Process of Feature Selection Feature selection aims to select a feature subset from the original set of features based on a/the feature's relevance and redundancy. Originally evaluation methods in feature selection are divided into five types: filter, wrapper, embedded, hybrid, and ensemble. ● Filter is considered the earliest method and also known as an open-loop method. It checks the features relying on the intrinsic characteristics prior to the learning tasks. It mainly measures the feature characteristics depending on four different kinds of measurement criteria, i.e. information, dependency, consistency, and distance. In the filter method, the feature selection process is performed independently of the data mining algorithm. It uses statistical standards for evaluating the ranking of the subset. Moreover, this technique is to perform good performance and high-efficiency computing, easily scalable in high dimensional datasets, and outperforms the wrapper technique. The primary downside of this method is that it neglects the integration between the selected subset and the performance of the induction algorithm. ● Wrapper, also known as a close-loop method, wraps the feature selection around the learning algorithm and uses the accuracy of the performance or the error rate of the classification process as a criterion of feature evaluation. By decreasing the estimation error of a specific classifier, it chooses the most discriminative subset of features. The wrapper method performs feature selection based on the performance of the learning algorithm; it selects the most optimal feature for the prediction algorithm. Hence it achieves better

Unsupervised Learning performance and high accuracy compared to the filter algorithm. The main disadvantage of this approach is computing complexity and more exposure to overfitting in comparison to the filter approach. ● Embedded method is a built-in feature selection mechanism that embeds feature selection in the learning algorithm and uses its properties to guide feature evaluation. The embedded method is more effective and more tractable than the wrapper method computationally while retaining similar performance. This is because the embedded method avoids the repeated execution of the classifier and the examination of every feature subset. The embedded method combines the qualities of both filter and wrapper methods. It selects features during the implementation of the mining algorithm, and hence it has less computational expensiveness ● Hybrid and Ensemble methods the recent developments in feature selection can be represented in the hybrid method. Thus, it can be developed either by integrating two various methods (e.g. wrapper and filter), two methods with the same criteria, or two feature selection approaches. In the hybrid method, the advantages of both methods can be inherited by combining their complementary strengths. The combination of filter and wrapper methods is the most common hybrid method. ● Ensemble method is a method that aims at building a group of feature subsets and then producing an aggregated result out of the group. This method is depending on various subsampling techniques where a particular feature selection method is implemented on a variety of subsamples, and the obtained features are merged to create a more stable subset. 13.4.2 Feature Extraction Feature extraction method extracting new features from original dataset, and it is very beneficial when we want to decrease the number of resources required for processing without missing relevant feature dataset. Feature extraction can also decrease the number of additional features for an offered study. Feature extraction produces a remarkable transformation of first features to create more significant features. Feature extraction is a process for creating new features that depend on the original input feature set to decrease the high dimensionality of the feature vector.
Unsupervised Learning The transformation method is done by algebraic transformation, and according to some optimization criteria. Also, feature extraction has the ability to handle essential information during dealing with high dimensional issues. These dimensionality reduction techniques aim to not lose a large amount of information during the feature transformation process by conserving the original relative distance between features and cover the original data potential structure. Feature extraction is less exposed to overfitting and performs good accuracy for the classification in comparison to the feature selection methods. However, the data description is lost occasionally after the transformation, and the cost of this process is expensive in several datasets. Feature extraction algorithms are classified into linear and nonlinear algorithms. However, the ideal feature extraction based dimensionality reduction methods is Principal Component Analysis (PCA), Multidimensional Scaling (MDS), Isometric Mapping (ISOMAP), Locally Linear Embedding (LLE), Linear Discriminant Analysis (LDA), Latent Semantic Indexing (LSI) and clustering methods. Figure 13.5 depicts the overall process of feature extraction method. Figure 13.5 Process of Feature Extraction 13.5 Principal Component Analysis (PCA) Principal Component Analysis (PCA) is a linear technique that performs dimensionality reduction by embedding the data into a linear subspace of lower dimensionality. Although there exist various techniques to do so, PCA is by far the most popular (unsupervised) linear technique. PCA constructs a low-dimensional representation of the data that describes as much of the variance in the data as possible. This is done by finding a linear basis of reduced dimensionality for the data, in which the amount of variance in the data is maximal. In mathematical terms, PCA attempts to find a linear mapping M that maximizes
Unsupervised Learning M T cov (X) M, where cov(X) is the covariance matrix of the data X. It can be shown that this linear mapping is formed by the d principal eigenvectors (i.e., principal components) of the covariance matrix of the zero-mean data. Hence, PCA solves the eigenproblem, cov (X) M = λM The eigenproblem is solved for the d principal eigenvalues λ. The low-dimensional data representations y i of the data points x i are computed by mapping them onto the linear basis M, i.e., Y = (X − X̄) M. PCA has been successfully applied in a large number of domains such as face recognition, coin classification, and seismic series analysis. The main drawback of PCA is that the size of the covariance matrix is proportional to the dimensionality of the data points. As a result, the computation of the eigenvectors might be infeasible for very high-dimensional data. In datasets in which n &gt; D, this drawback may be overcome by computing the eigenvectors of the squared Euclidean distance matrix (X − X̄) (X − X̄) T instead of the eigenvectors of the covariance matrix. Alternatively, iterative techniques such as Simple PCA or probabilistic PCA may be employed. The main idea behind using the PCA method is to seek the most accurate data representation in a lower dimensional space. PCA was introduced by Pearson (1901) and Hotelling (1933) to describe the variation in a set of multivariate data in terms of a set of uncorrelated variables. We typically have a data matrix of n observations on p correlated variables x 1 , x 2 ,... x p . PCA looks for a transformation of the x i into p new variables y i that are uncorrelated. For example, Let us project the 2-D data to 1-D subspace (a line) in such a way that the projection error is minimum. Figure 13.6 (a) shows the cases of two such projections where the right hand side projection shows small projection error and therefore is a good line to project to. We need to note that the good line to use for projection lies in the direction of largest variance. After the data is projected on the best line, need to transform the coordinate system to get 1D representation for vector y (Figure 13.6 (b)). Note that new data y has the same variance as old data x.
Unsupervised Learning (a) (b) Figure 13.6 (a) 2-D to 1-D Projection (b) 1-D Projection of the Data PCA projects the data along the directions where the data varies the most. These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues. The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions. Steps for PCA Let D be the data set D={x 1 , x 2 ,..., x n } where each x i is a d-dimensional vector. We need to reduce the dimension d to k. 1. First we must find the sample mean of all the x i 2. Then we subtract the sample mean from each x i of the data 3. Now we compute the scatter matrix 4. Next we compute eigenvectors e 1 , e 2 ,..., e k corresponding to the k largest

Unsupervised Learning eigenvalues of the scatter matrix S. 5. Let e 1 , e 2 ,..., e k be the columns of matrix 6. The desired y which is the closest approximation to x is Y = E t z The larger the eigenvalue of S, the larger is the variance in the direction of the corresponding eigenvector. Practical uses of PCA ● PCA is useful for pre-processing features before the actual classification is carried out. ● PCA does not take different classes into account; it only considers the properties of different features. ● If two features x i and x j are redundant, then one eigenvalue in A is very small and one dimension can be dropped. ● We do not need to choose between two correlating features. It is better to do the linear transform and then drop the least significant dimension. In this way both of the correlating features are utilized. ● This reduction in dimensionality enables the identification of the strongest patterns in the data. ● PCA allows the capture of most of the variability of the data with a small fraction of the total set of dimensions. ● It eliminates much of the noise in the data making it beneficial for data mining and other data analysis algorithms. Drawback of PCA PCA was designed for accurate data representation and not for data classification. The primary job of PCA is to preserve as much variance in data as possible.Therefore only if the direction of maximum variance is important for classification, it will work. However sometimes, the directions of maximum variance may not be useful for the classification. 13.6 Multidimensional Scaling Multidimensional scaling (MDS) represents a collection of nonlinear techniques that
Unsupervised Learning maps the high-dimensional data representation to a low-dimensional representation while retaining the pairwise distances between the data points as much as possible. The quality of the mapping is expressed in the stress function, a measure of the error between the pairwise distances in the low-dimensional and high-dimensional representation of the data. Two important examples of stress functions (for metric MDS) are the raw stress function and the Sammon cost function. The raw stress function is defined by in which ||x i – x j || is the Euclidean distance between the high-dimensional data points x i and x j , and ||y i – y j || is the Euclidean distance between the low- dimensional data points y i and y j . The Sammon cost function is given by The Sammon cost function differs from the raw stress function in that it puts more emphasis on retaining distances that were originally small. The minimization of the stress function can be performed using various methods, such as the eigendecomposition of a pairwise dissimilarity matrix, the conjugate gradient method, or a pseudo-Newton method. MDS is widely used for the visualization of data, e.g., in fMRI analysis and in molecular modeling. The popularity of MDS has led to the proposal of variants such as SPE, CCA, SNE, and FastMap. In addition, there exist non metric variants of MDS, that aim to preserve ordinal relations in data, instead of pairwise distances 13.7 Summary ●
When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers. ●
Simpler models are more robust on small datasets. Simpler models have less variance, that is, they vary less depending on the particulars of a sample,
Unsupervised Learning including noise, outliers, and so forth. ●
From a theoretical point of view, increasing the number of features should lead to better performance. ● Essentially, dimensionality reduction reduces time complexity since there will be less computation and therefore more efficient learning. ● Dimensionality reduction also reduces space complexity since there are less number of parameters resulting in more efficient storage. Moreover simpler models are more robust on small datasets since complex models may result in overfitting especially in the case of smaller datasets. ● The dimensionality reduction helps in data visualization (structure, groups, outliers, etc) specifically when reduction is to 2 or 3 dimensions. 13.8 Key Terms ● Feature selection: In this technique, information can be lost since some features should be excluded when the process of feature subset choice by doing this information can be reduced. ● Feature extraction: In this technique, the dimension can be decreased without losing much initial feature dataset. ● Wrapper: It is also known as a close-loop method, wraps the feature selection around the learning algorithm. ● Filter: It is considered the earliest method and also known as an open-loop method. It checks the features relying on the intrinsic characteristics prior to the learning tasks. 13.9 Check Your Progress Short-Answer Type Q1) Filter, Wrapper, Embedded, Hybrid, and Ensemble are evaluation methods for ________. Q2) Full form of PCA is ________. Q3) PCA is an unsupervised method in which all principal components are orthogonal to each other. (True/ False?) Q4) Multidimensional scaling (MDS) represents a collection of nonlinear techniques. (True/ False?)
Unsupervised Learning Long-Answer Type Q1) Write a short note on Multidimensional scaling. Q2) List some practical uses of PCA. Q3) Explain Principal Component Analysis step by step with the help of an example. Q4) What is the need of Dimensionality Reduction? Q5) Differentiate between Feature Extraction and Feature Selection. References: A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction, Rizgar Zebari, Adnan Mohsin Abdulazeez, Diyar Zeebaree, Dilovan Zebari, Journal of Applied Science and Technology Trends, May 2020. Dimensionality Reduction: A Comparative Review, L.J.P. van der Maaten, E.O. Postma, H.J. van den Herik, Journal of Machine Learning Research, January 2007. Multivariate Statistical Data Analysis- Principal Component Analysis (PCA), Sidharth Prasad Mishra, Uttam Sarkar, Subhash Taraphder, Sanjay Datta, Devi Prasanna Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram, International Journal of Livestock Research, May 2017. Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004. Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz and Shai Ben-David, Cambridge University Press, 2014.
Reinforcement Learning MODULE: IV REINFORCEMENT LEARNING
Reinforcement Learning

Reinforcement Learning Unit 14 – Basics of Reinforcement Learning Structure 14.0 Introduction 14.1 Unit Objectives 14.2 Elements of Reinforcement Learning 14.3 Model-based Learning 14.3.1 Value Iteration 14.3.2 Policy Iteration 14.4 Temporal Difference Learning 14.4.1 Deterministic Rewards and Actions 14.4.2 Nondeterministic Rewards and Actions 14.5 Q- Learning Algorithm 14.6 Summary 14.7 Key Terms 14.8 Check Your Progress 14.0 Introduction Let's start with an example. If we want to build a machine that learns to play chess. In this case we cannot use a supervised learner for two reasons. First, it is very costly to have a teacher that will take us through many games and suggest the best move for each position. Second, in many cases, there is no such thing as the best move; the goodness of a move depends on the moves that follow. A single move does not count; a sequence of moves is good if after playing them we win the game. The only feedback is at the end of the game when we win or lose the game. On observing the above example, we observe that there is a decision maker, called the agent, that is placed in an environment. In chess, the game-player is the decision maker and the environment is the board. At any time, the environment is in a certain state that is one of a set of possible states—for example, the state of the board.

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.

For example, consider teaching a baby a new thing: you cannot instruct the baby what is to be done, but you can reward (a chocolate) or punish (scold) it depending on whether it does the right/ wrong thing. Here the baby

has to figure out what it did that made it get the reward or punishment,

which is known in
Reinforcement Learning reinforcement learning context as the credit assignment problem. Learning takes place as a result of interaction between an agent and the world. In other words, percept received by an agent should be used not only for understanding, interpreting or prediction, as in the machine learning tasks we have discussed so far, but also for acting.
Reinforcement learning is more general than supervised and unsupervised learning and learning from interaction with the environment to achieve a goal and

getting an agent to act in the world so as to maximize its rewards.

It allows agents to automatically determine the ideal behaviour within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behaviour; this is known as the reinforcement signal. Figure 14.1 depicts the basic concept of reinforcement learning. Figure 14.1 Basic concept of reinforcement learning The motivation behind reinforcement learning is that it allows an agent to learn its behavior based on feedback from the environment. This behavior can be learnt once and for all, or keep on adapting as time goes by. If the problem is modelled with care, some reinforcement learning algorithms can converge to the global optimum; this is the ideal behavior that maximizes the reward. It is a trial-and-error learning paradigm which learns from rewards and punishments. Reinforcement learning is not just an algorithm but a new paradigm in itself. Its objective is to learn about a system from minimal feedback like its behavior, control. It is inspired by behavioral psychology.
Reinforcement Learning Let's differentiate Reinforcement, Supervised, and Unsupervised learning. In supervised learning, the training information contains the desired (target) outputs. In reinforcement learning the training information contains evaluations (rewards/ penalties) and the output are actions of the agent. In supervised learning we get the target output and if we subtract the actual output from the target output we will get the error of the system but that is not the case with reinforcement learning. In reinforcement learning the objective is to get as much reward as possible. In reinforcement learning the agent acts on its environment, it receives some evaluation of its action (reinforcement), but is not told of which action is the correct one to achieve its goal. The training data: S, A, R (State- Action- Reward) and the learning system needs to develop an optimal policy (sequence of decision rules) so as to maximize its long-term reward. 14.1 Unit Objectives After completing this unit, the reader will be able to: ● Explain the basic concept of Reinforcement Learning. ● Discuss the Elements of Reinforcement Learning ● Outline the Model-based Learning. ● Discuss the Q- learning Algorithm. ● Illustrate the concept of Temporal Difference Learning. 14.2 Elements of Reinforcement Learning The basic elements of reinforcement learning are discussed below: ● Policy- what to do. ● Reward- what is good: It defines the goal in a reinforcement learning problem and gives the agent a sense of what is good in an immediate sense. ● Value- what is good because it predicts reward: The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state in other words gives the agent a sense of what is good in the long run. ● Model- what follows what: It is used to predict the states the environment will be in after the agent performs its actions. The agent often uses the model to compute series of potential state–action sequences

Reinforcement Learning Figure 14.2 Elements of Reinforcement Learning We have the environment and a state given to the agent. The agent does an action based on the policy. The action effects the environment. The environment moves to another state and gives you a reward. Figure 14.3 shows this transition model, (i.e.) how action influence states. Given the state and the action, we have the reward R, the immediate value of state-action transition and we have the policy which maps states to actions. Figure 14.3 Transition Model of RL Markov decision process (MDP) To summarize, the learning decision maker is called the agent. The agent interacts with the environment that includes everything outside the agent. The agent has sensors to decide on its state in the environment and takes an action that modifies its state. When the agent takes an action, the environment provides a reward. If time is discrete as t = 0, 1, 2, . . ., and s t ∈ S denotes the state of the agent at time t, where S is the set of all possible states. a t ∈ A(s t) denotes the action that the agent takes at time t where A(s t) is the set of possible actions in state s t . When the agent in state s t takes the action a t , the clock ticks, reward r t+1 ∈ ℜ is received, and the agent
Reinforcement Learning moves to the next state, s t +1. The problem is modeled using a Markov decision process (MDP). The reward and next state are sampled from their respective probability distributions, p(r t +1 |s t , a t) and P (s t+1 |s t , a t) Note that what we have is a Markov system where the state and reward in the next time step depend only on the current state and action. In some applications, reward and next state are deterministic, and for a certain state and action taken, there is one possible reward value and next state. Depending on the application, a certain state may be designated as the initial state and in some applications, there is also an absorbing terminal (goal) state where the search ends; all actions in this terminal state transition to itself with probability 1 and without any reward. The sequence of actions from the start to the terminal state is an episode, or a trial. The policy, π , defines the agent's behavior and is a mapping from the states of the environment to actions: π : S → A . The policy defines the action to be taken in any state s t : a t = π (s t). The value of a policy π , V π (s t), is the expected cumulative reward that will be received while the agent follows the policy, starting from state s t . ● In the finite-horizon or episodic model, the agent tries to maximize the expected reward for the next T steps: ● Certain tasks are continuing, and there is no prior fixed limit to the episode. In the infinite-horizon model, there is no sequence limit, but future rewards are discounted: where 0 ≤ &gt; 1 is the discount rate to keep the return finite. If = 0, then only the immediate reward counts. As approaches 1, rewards further in the future count more, and we say that the agent becomes more farsighted. is less than 1
Reinforcement Learning because there generally is a time limit to the sequence of actions needed to solve the task. The agent may be a robot that runs on a battery. We prefer rewards sooner rather than later because we are not certain how long we will survive. ● For each policy π , there is a V π (s t), and we want to find the optimal policy π ∗ such that In some applications, for example, in control, instead of working with the values of states, V (s t), we prefer to work with the values of state- action pairs, Q(s t , a t). V (s t) denotes how good it is for the agent to be in state s t , whereas Q(s t , a t) denotes how good it is to perform action a t when in state s t . We define Q ∗ (s t , a t) as the value, that is, the expected cumulative reward, of action a t taken in state s t and then obeying the optimal policy afterward. The value of a state is equal to the value of the best possible action: The above equation is known as Bellman's equation 14.3 Model-based Learning We start with model-based learning where we completely know the environment model parameters, p(r t +1 |s t , a t) and P (s t+1 |s t , a t). In such a case, we do not need any exploration and can directly solve for the optimal value function and policy using dynamic programming. The optimal value function is unique and is the solution to
Reinforcement Learning the simultaneous equations given in the previous section. Once we have the optimal value function, the optimal policy is to choose the action that maximizes the value in the next state: 14.3.1 Value Iteration To find the optimal policy, we can use the optimal value function, and there is an iterative algorithm called value iteration that has been shown to converge to the correct V ∗ values. Its pseudocode is given in Table 14.1. Table 14.1 Value iteration algorithm for model-based learning Initialize V (s) to arbitrary values Repeat For all s ∈ S For all a ∈ A Q(s, a) ← E[r |s, a] + ∑ s' ∈ S P (s' |s, a)V (s') V (s) ← max a Q(s, a) Until V (s) converge We say that the values converged if the maximum value difference between two iterations is less than a certain threshold δ: max |V (l+1) (s) − V (l) (s)| &gt; δ s∈S where, l is the iteration counter. Because we care only about the actions with the maximum value, it is possible that the policy converges to the optimal one even before the values converge to their optimal values. Each iteration is O (|S| 2 |A|), but frequently there is only a small number k &gt; |S| of next possible states, so complexity decreases to O (k|S||A|). 14.3.2 Policy Iteration In policy iteration, we store and update the policy rather than doing this indirectly

Reinforcement Learning over the values. The pseudocode is given in Table 14.2. The idea is to start with a policy and improve it repeatedly until there is no change. The value function can be calculated by solving for the linear equations. We then check whether we can improve the policy by taking these into account. This step is guaranteed to improve the policy, and when no improvement is possible, the policy is guaranteed to be optimal. Each iteration of this algorithm takes O (|A||S| 2 + |S| 3) time that is more than that of value iteration, but policy iteration needs fewer iterations than value iteration. Table 14.2 Policy iteration algorithm for model-based learning Initialize a policy π′ arbitrarily Repeat π ← π′ Compute the values using π by solving the linear equations V π (s) = E[r |s, π (s)] + ∑ s′ ∈ S P (s′ |s, π (s))V π (s′) Improve the policy at each state π′ (s) ← arg max a (E[r |s, a] + ∑ s′ ∈ S P (s′ |s, a)V π (s′)) Until π = π′ 14.4 Temporal Difference Learning Model is defined by the reward and next state probability distributions, and as we have already discussed, when we know these, we can solve for the optimal policy using dynamic programming. However, these methods are costly, and we seldom have such perfect knowledge of the environment. The more interesting and realistic application of reinforcement learning is when we do not have the model. This requires exploration of the environment to query the model. We first discuss how this exploration is done and later see model-free learning algorithms for deterministic and nondeterministic cases. Though we are not going to assume a full knowledge of the environment model, we will however require that it be stationary. When we explore and get to see the value of the next state and reward, we use this information to update the value of the current state. These algorithms are called temporal difference algorithms because what we do is look at the difference

Reinforcement Learning between our current estimate of the value of a state (or a state-action pair) and the discounted value of the next state and the reward received. Temporal difference (TD) learning is a prediction-based machine learning method. It has primarily been used for the reinforcement learning problem, and is said to be a combination of Monte Carlo ideas because it learns by sampling the environment according to some policy, and dynamic programming (DP) ideas as it approximates its current estimate based on previously learned estimates. Temporal Difference (TD) learning methods can be used to estimate value functions. If the value functions were to be calculated without estimation, the agent would need to wait until the final reward was received before any state -action pair values can be updated. Once the final reward was received, the path taken to reach the final state would need to be traced back and each value updated accordingly. This Can be expressed formally as: where s t is the state visited at time t, R t is the reward after time t and is a constant parameter. On the other hand, with TD methods, an estimate of the final reward is calculated at each state and the state-action value updated for every step of the way: where r t+1 is the observed reward at time t+1 The TD method is called a bootstrapping method, because the value is updated partly using an existing estimate and not a final reward. The Environment,

p(s t +1 |s t , a t) and P (r t+1 |s t , a t),

is not known; model-free learning. There is need for exploration to sample from

p(s t +1 |s t , a t) and P (r t+1 |s t , a t).

We use the reward received in the next time step to update the value of current state (action). The temporal difference between the value of the current action and the value discounted from the next state. Exploration Strategies At each state (except a terminal state) the agent must select an action. There are several ways in which to decide which action to take. The simplest of these is greedy selection: the agent always selects the action that has the highest state-action value. This method is pure exploitation. There are many more sophisticated strategies used which combine exploitation and exploration. Some of these methods are: ● ε-greedy: In this method most of the time the action with the highest estimated reward is chosen, called the greediest action. Every once in a while, say with a

Reinforcement Learning small probability ε, an action is selected at random. The action is selected uniformly, independent of the action-value estimates. ● Boltzman method: Here we move smoothly from exploration to exploitation. Here we need to choose action a in states with probability P. Here we have the concept of Simulated annealing where T is a temperature. Large T means that each action has approximately the same probability. Small T leads to more greedy behavior. Typically we start with a large T and decrease its value with time. 14.4.1 Deterministic Rewards and Actions In this case there is a single possible reward and next state for a state-action pair. The cumulative reward is given by and is used to update the value of Q(s t , a t). Using any of the strategy explained above we move from state S t to state S t+1 taking action a t which yields the reward r t+1 and this is used to update the previous action as . The q value increases as we try to find paths that give higher cumulative reward and never decrease until optimal value is reached. 14.4.2 Nondeterministic Rewards and Actions When next states and rewards are nondeterministic (there is an opponent or randomness in the environment), then there is a probability distribution for

next state P (s t+1 | s t , a t) and reward P (r t+1 | s t , a t).

As there are different rewards for the same action from a state we keep averages (expected values) instead of assignments. This is known as Q- learning and predicted cumulative reward is given as The value of is the sample of the instances for each state - action pair. In temporal difference learning method Off-policy method is used i.e. policy is not

Reinforcement Learning used and the value of the next best action is used to estimate the temporal difference. In Q-learning based SARSA an On-policy method uses policy to determine temporal difference rather than the next best action. The state value V (s t) can be learned using the temporal difference learning and can be updated using the rule . The is the predicted value and V (s t) is the current estimated value and this temporal difference should be reduced to get the optimal value. 14.5 Q- Learning Algorithm Q-Learning is an Off-Policy algorithm for Temporal Difference learning. Off-policy algorithms can update the estimated value functions using hypothetical actions, those which have not actually been tried. Q-learning is an approach of reinforcement learning that attempts to learn the value of taking each action in each state. This approach is called a model-free approach. Q-learning can be used to find an optimal action-selection policy for any given Markov Decision Process. As we discussed earlier a policy can be defined as a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. Q-learning is able to compare the expected utility of the available actions without requiring a model of the environment. Now while solving MDPs we defined the function Q(i,a) which provides the expected value of taking action a in state i. The Q value incorporates both the immediate reward and expected value of the next state. Q-learning converges to an optimal policy in both deterministic and nondeterministic MDPs. The advantages of Q-learning are that it is easy to work with in practice and is exploration insensitive i.e. optimal value will be reached in the end. As the exploration strategy used has no effect on Q-learning it is the most popular and the most effective model-free algorithm for learning from delayed reinforcement. The disadvantage of Q-learning is that it does not scale well i.e. for generalizing over a huge number of states and actions convergence is quite slow. Only practical in a small number of problems because Q- learning can require many thousands of training iterations to converge in even modest-sized problems and hence in many

Reinforcement Learning situations the memory resources required by this method become too large. The Q-Learning follows the below steps to learn the function: 1. Start with initial Q-function (e.g. all zeros) 2. Take action according to an explore/exploit policy (should converge to greedy policy) 3. Perform TD update Q (s, a) ← Q(s, a) + α(R (s) + βmax Q (s', a') - Q (s, a)) a' Q(s,a) is the current estimate of optimal Q-function. 4. Go to step 2. Table 14.3 Q-learning Algorithm Initialize all Q(s, a) arbitrarily For all episodes Initialize s Repeat Choose a using policy derived from Q, e.g., -greedy Take action a, observe r and s' Update Q(s, a): Q(s, a) ← Q(s, a) + η(r + max a' Q(s', a') – Q(s, a)) s ← s' Until s is terminal state SARSA Algorithm The Sarsa algorithm is an On-Policy algorithm for TD-Learning. The major difference between this algorithm and Q-Learning, is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. The name SARSA actually comes from the fact that the updates are done using the quintuple Q(s, a, r, s', a') where: s, a are the original state and action, r is the reward observed in the following state and s', a' are the new state-action pair. The On-policy SARSA instead of looking for all possible next actions a' and choosing the best, uses the policy derived from Q values to choose a'. It uses the Q value of a' to calculate the temporal difference. On employing the GLIE (greedy in the limit with infinite

Reinforcement Learning exploration) policy where, ● All state-action pairs are visited an infinite number of times. ● The policy converges in the limit to the greedy policy. ● The algorithm converges with probability 1 to the optimal policy and state- action values. Table 14.4 SARSA Algorithm Initialize all Q(s, a) arbitrarily For all episodes Initialize s Choose a using policy derived from Q, e.g., -greedy Repeat Take action a, observe r and s' Choose a' using policy derived from Q, e.g., -greedy Update Q(s, a): Q(s, a) ← Q(s, a) + η(r + Q(s', a') – Q(s, a)) s ← s', a ← a' Until s is terminal state 14.6 Summary ●

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards. ●

Simple reward feedback is required for the agent to learn its behaviour; this is known as the reinforcement signal. ● The training data: S, A, R (State- Action- Reward) and the learning system needs to develop an optimal policy (sequence of decision rules) so as to maximize its long-term reward. ● Temporal difference (TD) learning is a prediction-based machine learning method. ● The TD method is called a bootstrapping method, because the value is updated partly using an existing estimate and not a final reward. 14.7 Key Terms

Reinforcement Learning ● Agent: The learning decision maker is called the agent. ● Environment: The agent interacts with the environment that includes everything outside the agent. ● State: The agent has sensors to decide on its state in the environment and takes an action that modifies its state. ● Reward: When the agent takes an action, the environment provides a reward. 14.8 Check Your Progress Short-Answer Type Q1) Value and Policy Iteration are methods of ___________ learning. Q2) __________ model of reinforcement learning depicts how action influences states. Q3) In ________ system, the state and reward in the next time step depend only on the current state and action. Q4) Which of the following is NOT an element of Reinforcement Learning? a) Reward b) Model of Environment c) Value d) Policy Long-Answer Type Q1) State and explain Q-learning Algorithm. Q2) What is Reinforcement Learning? Discuss the basic elements of Reinforcement learning. Q3) Differentiate between Value and Policy Iteration. Q4) Define Temporal Difference Learning. Q5) State and explain SARSA Algorithm. References: Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004. Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997.

Reinforcement Learning Unit 15 – Genetic Algorithms Structure 15.0 Introduction 15.1 Unit Objectives 15.2 Representing Hypotheses 15.3 Genetic Operators 15.4 Fitness Function and Selection 15.5 Important concepts related to Genetic Algorithms 15.5.1 Crowding 15.5.2 Population Evolution and the Schema Theorem 15.5.3 Convergence and Diversity 15.5.4 Phenotype and Genotype 15.5.5 Exploitation and Exploration 15.6 Genetic Programming 15.7 Summary 15.8 Key Terms 15.9 Check Your Progress 15.0 Introduction Genetic algorithms provide an approach to learning that is based loosely on simulated evolution. Hypotheses are often described by bit strings whose interpretation depends on the application, though hypotheses may also be described by symbolic expressions or even computer programs. The search for an appropriate hypothesis begins with a population, or collection, of initial hypotheses. Members of the current population give rise to the next generation population by means of operations such as random mutation and crossover, which are patterned after processes in biological evolution. At each step, the hypotheses in the current population are evaluated relative to a given measure of fitness, with the most fit hypotheses selected probabilistically as seeds for
Reinforcement Learning producing the next generation. Basically the concept of genetic approach is easy to understand. It is modular and separate from application. It supports the important and complex problem of multi- objective optimization. GA is for noisy environments. Genetic Approach always finds an answer and the answer gets better with time. The approach is inherently parallel and easily distributed. There are many ways to speed up and improve a GA-based application as more knowledge about the problem domain is gained. Genetic approach makes it easy to exploit previous or alternate solutions and forms flexible building blocks for hybrid applications. It has history and has been applied to many optimization problems. One of the main disadvantages of genetic algorithms is that there is no guarantee for optimal solution within a finite time or in other words they lack the killer instinct. It has a comparatively weak theoretical basis. In order to arrive at a solution there may be a need for extensive parameter tuning since the search capability is sensitive to parameter settings and operators adopted. The representation, fitness function, and even the position of bits influence judgment of the achievement of fitness optimum, that is whether the optimum is global or only local. Genetic algorithms are often computationally expensive and slow to reach optimum solutions. Genetic algorithms have been applied successfully to a variety of learning tasks and to other optimization problems. For example, they have been used to learn collections of rules for robot control and to optimize the topology and learning parameters for artificial neural networks. This unit covers both genetic algorithms, in which hypotheses are typically described by bit strings, and genetic programming, in which hypotheses are described by computer programs. 15.1 Unit Objectives After completing this unit, the reader will be able to: ● Explain the basic concept of Genetic Algorithms. ● Discuss the Genetic Operators ● Outline the concept of Genetic Programming. 15.2 Representing Hypotheses

100% **MATCHING BLOCK 131/189** **W**

Genetic algorithms (GAS) provide a learning method motivated by an analogy to

Reinforcement Learning

99% **MATCHING BLOCK 132/189** **W**

biological evolution. Rather than search from general-to-specific hypotheses, or from simple-to-complex, GAS generates successor hypotheses by repeatedly mutating and recombining parts of the best currently known hypotheses. At each step, a collection of hypotheses called the current population is updated by replacing some fraction of the population by offspring of the most fit current hypotheses. The process forms a generate-and-test beam-search of hypotheses, in which variants of the best current hypotheses are most likely to be considered next. The popularity of GAS is motivated by a number of factors including: ● Evolution is known to be a successful, robust method for adaptation within biological systems. ● GAS can search spaces of hypotheses containing complex interacting parts, where the impact of each part on overall hypothesis fitness may be difficult to model. ● Genetic algorithms are easily parallelized and can take advantage of the decreasing costs of powerful computer hardware. The problem addressed by GAS is to search a space of candidate hypotheses to identify the best hypothesis. In GAS the best hypothesis is defined as the one that optimizes a predefined numerical measure for the problem at hand, called the hypothesis fitness. For example, if the learning task is the problem of approximating an unknown function given training examples of its input and output, then fitness could be defined as the accuracy of the hypothesis over this training data. If the task is to learn a strategy for playing chess, fitness could be defined as the number of games won by the individual when playing against other individuals in the current population. Although different implementations of genetic algorithms vary in their details, they typically share the following structure: The algorithm operates by iteratively updating a pool of hypotheses, called the population. On each iteration, all members of the population are evaluated according to the fitness function. A new population is then generated by probabilistically selecting the fit individuals from the current population. Some of these selected individuals are carried forward into the next generation population intact. Others are used as the basis for creating new offspring individuals by applying genetic operations such as crossover and mutation.

Reinforcement Learning A prototypical genetic algorithm is described in Table 15.1.

100% MATCHING BLOCK 133/189 W

The inputs to this algorithm include the fitness function for ranking candidate hypotheses, a threshold defining an acceptable level of fitness for terminating the algorithm, the size of the population to be maintained, and parameters that determine how successor populations are to be generated: the fraction of the population to be replaced at each generation and the mutation rate.

It should be noted that

100% MATCHING BLOCK 134/189 W

in this algorithm each iteration through the main loop produces a new generation of hypotheses based on the current population. First, a certain number of hypotheses from the current population are selected for inclusion in the next generation. These are selected probabilistically, where the probability of selecting hypothesis h i is given by Thus, the probability that a hypothesis will be selected is proportional to its own fitness and is inversely proportional to the fitness of the other competing hypotheses in the current population. Once these members of the current generation have been selected for inclusion in the next generation population, additional members are generated using a crossover operation. Crossover, defined in detail in the next section, takes two parent hypotheses from the current generation and creates two offspring hypotheses by recombining portions of both parents. The parent hypotheses are chosen probabilistically from the current population, again using the probability function given by

the above

100% MATCHING BLOCK 135/189 W

equation. After new members have been created by this crossover operation, the new generation population now contains the desired number of members. At this point, a certain fraction m of these members are chosen at random, and random mutations all performed to alter these members. This GA algorithm thus performs a randomized, parallel beam search for hypotheses that perform well according to the fitness function. In the following subsections, we describe in more detail the representation of hypotheses and genetic operators used in this algorithm.

Reinforcement Learning Table 15.1 Prototypical Genetic

100% MATCHING BLOCK 136/189 W

Algorithm Representing Hypotheses Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover. The hypotheses represented by these bit strings can be quite complex. For example, sets of if-then rules can easily be represented in this way, by choosing an encoding of rules that allocates specific substrings for each rule precondition and postcondition. To see how if-then rules can be encoded by bit strings, first consider how we might use a bit string to describe a constraint on the value of a single attribute. To pick an example, consider the attribute Outlook, which can take on any of the three values Sunny, Overcast, or Rain. One obvious way to represent a constraint on Outlook is

Reinforcement Learning

to use a bit string of length three, in which each bit position corresponds to one of its three possible values. Placing a 1 in some position indicates that the attribute is allowed to take on the corresponding value. For example, the string 010 represents the constraint that Outlook must take on the second of these values, or Outlook = Overcast. Similarly, the string 011 represents the more general constraint that allows two possible values, or (Outlook = Overcast v Rain). Note 11 1 represents the most general possible constraint, indicating that we don't care which of its possible values the attribute takes on. Given this method for representing constraints on a single attribute, conjunctions of constraints on multiple attributes can easily be represented by concatenating the corresponding bit strings. For example, consider a second attribute, Wind, that can take on the value Strong or Weak. A rule precondition such as (Outlook = Overcast ^Rain) A (Wind = Strong) can then be represented by the following bit string of length five: Outlook Wind 01 1 10 Rule postconditions (such as PlayTennis = yes) can be represented in a similar fashion. Thus, an entire rule can be described by concatenating the bit strings describing the rule preconditions, together with the bit string describing the rule postcondition. For example, the rule IF Wind = Strong THEN PlayTennis = yes would be represented by the string Outlook Wind PlayTennis 111 10 10 where the first three bits describe the don't care constraint on Outlook, the next two bits describe the constraint on Wind, and the final two bits describe the rule postcondition (here we assume PlayTennis can take on the values Yes or No). Note the bit string representing the rule contains a substring for each attribute in the hypothesis space, even if that attribute is not constrained by the rule preconditions. This yields a fixed length bit-string representation for rules, in which substrings at specific locations describe constraints on specific attributes. Given this representation for single rules, we can represent sets of rules by similarly concatenating the bit string representations of the individual rules.

Reinforcement Learning

In designing a bit string encoding for some hypothesis space, it is useful to arrange for every syntactically legal bit string to represent a well-defined hypothesis. To illustrate, note in the rule encoding in the above paragraph the bit string 11 1 10 11 represents a rule whose postcondition does not constrain the target attribute PlayTennis. If we wish to avoid considering this hypothesis, we may employ a different encoding (e.g., allocate just one bit to the PlayTennis postcondition to indicate whether the value is Yes or No), alter the genetic operators so that they explicitly avoid constructing such bit strings, or simply assign a very low fitness to such bit strings. In some GAS, hypotheses are represented by symbolic descriptions rather than bit strings. 15.3 Genetic Operators The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population. These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are crossover and mutation. ● The crossover operator produces two new offspring from two parent strings, by copying selected bits from each parent. The bit at position i in each offspring is copied from the bit at position i in one of the two parents. The choice of which parent contributes the bit for position i is determined by an additional string called the crossover mask. To illustrate, consider the single-point crossover operator at the top of Table Consider the topmost of the two offspring in this case. This offspring takes its first five bits from the first parent and its remaining six bits from the second parent, because the crossover mask 11 11 1000000 specifies these choices for each of the bit positions. The second offspring uses the same crossover mask, but switches the roles of the two parents. Therefore, it contains the bits that were not used by the first offspring. In single-point crossover, the crossover mask is always constructed so that it begins with a string containing n contiguous 1

s,

followed by the necessary number of 0s to complete the string. This results in offspring in which the first n bits are contributed by one parent and the remaining bits by the second parent. Each time the single-point crossover operator is applied the crossover point n is chosen at random, and the crossover mask is then created and

Reinforcement Learning

applied. In two-point crossover, offspring are created by substituting intermediate segments of one parent into the middle of the second parent string. Put another way, the crossover mask is a string beginning with n 0 zeros, followed by a contiguous string of n 1 ones, followed by the necessary number of zeros to complete the string. Each time the two-point crossover operator is applied, a mask is generated by randomly choosing the integers n 0 and n 1 .

Uniform crossover combines bits sampled uniformly from the two parents. In this case the crossover mask is generated as a random bit string with each bit chosen at random and independent of the others. ● Mutation introduces randomness into the population and is a type of asexual reproduction. The idea of mutation is to reintroduce divergence into a converging population. However mutation is performed only on a small part of the population, in order to avoid entering an unstable state. In Binary-Coded GAs, each bit in the chromosome is mutated with probability p bm which is known as the mutation rate. In this case for each bit we generate a random number r between 0 and 1 and if r &gt; m r (i.e., mutation rate) then we flip the bit. For real number representation we randomly create a number for the mutated bit. In the case of tree representation there are two types of mutation namely tree mutation where we randomly create a branch replacing the current branch and point mutation where we change this bit value by random number/ random factor.
Reinforcement Learning Figure 15.1 Common operators for genetic algorithms 15.4 Fitness Function and Selection During each successive generation, a proportion of the existing population is selected to breed a new generation. The main idea during selection is to focus on fittest individuals. Individual solutions are selected based on fitness function values, where fitter solutions are typically more likely to be selected. Generally selection methods preferentially select the best solutions. However sometimes other methods rate only a random sample of the population, since rating all the solutions in the search space may be a very time-consuming process. Most of the fitness functions are stochastic that ensure that only a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions.

The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population. If the task is to learn classification rules, then the fitness function typically has a component that scores the classification accuracy of the rule over a set of provided

Reinforcement Learning

training examples. Often other criteria may be included as well, such as the complexity or generality of the rule. More generally, when the bit-string hypothesis is interpreted as a complex procedure (e.g., when the bit string represents a collection of if-then rules that will be chained together to control a robotic device), the fitness function may measure the overall performance of the resulting procedure rather than performance of individual rules. In our prototypical GA shown in above Table 15.1 , the probability that a hypothesis will be selected is given by the ratio of its fitness to the fitness of other members of the current population as seen in

the

equation. This method is sometimes called fitness proportionate selection, or roulette wheel selection. Other methods for using fitness to select hypotheses have also been proposed. For example, in tournament selection, two hypotheses are first chosen at random from the current population. With some predefined probability p the more fit of these two is then selected, and with probability (1 - p) the less fit hypothesis is selected. Tournament selection often yields a more diverse population than fitness proportionate selection. In another method called rank selection, the hypotheses in the current population are first sorted by fitness. The probability that a hypothesis will be selected is then proportional to its rank in this sorted list, rather than its fitness.

Fitness function: The fitness of each hypothesized rule set is based on its classification accuracy over the training data. In particular, the function used to measure fitness is where correct (h) is the percent of all training examples correctly classified by hypothesis h.

Roulette Wheel Selection The most common method of selection is the roulette wheel selection. In roulette wheel selection, individuals are given a probability of being selected that is directly proportionate to their fitness. Two individuals are then chosen randomly based on these probabilities and produce offspring. This method is called the roulette wheel method because each individual is assigned a slice of a circular "roulette wheel", where the area of the slice is proportional to the individual's fitness. The wheel is spun

Reinforcement Learning N times, where N is the number of individuals in the population. The slice underneath the wheel when it stops determines which individual becomes a parent. This spinning introduces the probability aspect. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next step. The selective capability depends on the variance of the fitness in the population. The population converges when the variance of the fitness is low, that is the difference between the fitness of the individuals is small. This is because when the probabilities of the different individuals selected are similar it is difficult to select a better individual from individuals with similar fitness values. Fitness Scaling There are a number of disadvantages associated with selection based on proportion of fitness as is the case with Roulette wheel. These selection methods cannot be used directly for minimization problems, but needs to be transformed to an equivalent maximization problem. There is a domination of super individuals in early generations and slow convergence in later generations. Loss of selection pressure that is search direction as population converges is another problem. Fitness scaling has often been used in early days to combat the above problems. We will discuss two types of fitness scaling namely simple scaling and sigma scaling. a) Simple scaling: In the case of simple scaling the ith individual's fitness is defined as: f scaled (t) = f original (t) - f worst (t), where t is the generation number and f worst (t) is the fitness of the worst individual so far seen. b) Sigma Scaling: In the case of sigma scaling the ith individual's fitness is defined as: This method keeps the selection pressure relatively constant. An individual's expected value is a function of its fitness, the population mean, and the population standard deviation. 15.5 Important concepts related to Genetic Algorithms
Reinforcement Learning There are various important problems and concepts associated with GA. We will discuss them briefly. 15.5.1 Crowding Some Genetic Algorithm applications experience a practical difficulty, known as

crowding. Crowding is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of this individual and very similar individuals take over a large fraction of the population. The negative impact of crowding is that it reduces the diversity of the population, thereby slowing further progress by the GA. Several strategies have been explored for reducing crowding.

Many of these techniques are inspired by the analogy to biological evolution. ●

One approach is to alter the selection function, using criteria such as tournament selection or rank selection in place of fitness proportionate roulette wheel selection. ● A related strategy is fitness sharing, in which the measured fitness of an individual is reduced by the presence of other, similar individuals in the population. ● A third approach is to restrict the kinds of individuals allowed to recombine to form offspring. For example, by allowing only the most similar individuals to recombine, we can encourage the formation of clusters of similar individuals, or multiple "subspecies" within the population. ● A related approach is to spatially distribute individuals and allow only nearby individuals to recombine. 15.5.2

Population Evolution and the Schema Theorem It is interesting to ask whether one can mathematically characterize the evolution over time of the population within a GA. The schema theorem provides one such characterization. It is based on the concept of schemas, or patterns that describe sets of bit strings. To be precise, a schema is any string composed of 0s, 1s, and *'s. Each schema represents the set of bit strings containing the indicated 0s and 1s, with each "*" interpreted as a "don't care." For example, the schema 0*10 represents the set of bit strings that includes exactly 0010 and 01 10.

Reinforcement Learning

96% MATCHING BLOCK 149/189 W

An individual bit string can be viewed as a representative of each of the different schemas that it matches. For example, the bit string 0010 can be thought of as a representative of 2 4 distinct schemas including 00**, 0* 10, ****, etc. Similarly, a population of bit strings can be viewed in terms of the set of schemas that it represents and the number of individuals associated with each of these schemas. The schema theorem characterizes the evolution of the population within a GA in terms of the number of instances representing each schema. Let m(s, t) denote the number of instances of schema s in the population at time t (i.e., during the t th generation). The schema theorem describes the expected value of m (s, t+1) in terms of m(s, t) and other properties of the schema, population, and GA algorithm parameters. The evolution of the population in the GA depends on the selection step, the recombination step, and the mutation step. Let us start by considering just the effect of the selection step. Let f (h) denote the fitness of the individual bit string h and (t) denote the average fitness of all individuals in the population at time t. Let n be the total number of individuals in the population. Let h∈s ∩p , indicate that the individual h is both a representative of schema s and a member of the population at time t. Finally, let ?(

s, t)

94% MATCHING BLOCK 150/189 W

denote the average fitness of instances of schema s in the population at time t. Let’s calculate the expected value of m(s,t+1), which we denote E[m(s,t+1)]. We can calculate E[m(s,t+1)] using the probability distribution for selection given in equation, which can be restated using our current terminology as follows Now if we select one member for the new population according to this probability distribution, then the probability that we will select a representative of schema s is The second step above follows from the fact that by definition, Reinforcement Learning Equation gives the probability that a single hypothesis selected by the GA will be an instance of schema s. Therefore, the expected number of instances of s resulting from the n independent selection steps that create the entire new generation is just n times this probability.

100% MATCHING BLOCK 151/189 W

While the above analysis considered only the selection step of the GA, the crossover and mutation steps must be considered as well. The schema theorem considers only the possible negative influence of these genetic operators

100% MATCHING BLOCK 152/189 W

The full schema theorem thus provides a lower bound on the expected frequency of schema s, as follows: Here, p c is the probability that the single-point crossover operator will be applied to an arbitrary individual, and p m is the probability that an arbitrary bit of an arbitrary individual will be mutated by the mutation operator. o(s) is the number of defined bits in schema s, where 0 and 1 are defined bits, but * is not. d(s) is the distance between the leftmost and rightmost defined bits in s. Finally, l is the length of the individual bit strings in the population.

It should be noted the

86% MATCHING BLOCK 153/189 W

leftmost term in the equation describes the effect of the selection step. The middle term describes the effect of the single-point crossover operator-in particular.

97% MATCHING BLOCK 154/189 W

The rightmost term describes the probability that an arbitrary individual representing schema s will still represent schema s following application of the mutation operator. Note that the effects of single-point crossover and mutation increase with the number of defined bits o(s) in the schema and with the distance d(s) between the defined bits. Thus, the schema theorem can be roughly interpreted as stating that more fit schemas will tend to grow in influence, especially schemas containing a small number of defined bits (i.e., containing a large number of *'s), and especially when these defined bits are near one another within the bit string. The schema theorem is perhaps the Reinforcement Learning most widely cited characterization of population evolution within a GA. 15.5.3

Convergence and Diversity Convergence and premature convergence are issues that are associated with genetic algorithms. GA continues while fitter chromosomes exist in the population. The population will converge so that all individuals are as fit as each other. The converged population will not evolve further if the best solution is found. However, though this process actually converges prematurely, the algorithm terminates. Diversity needs to be maintained in genes. Individuals with the same fitness might keep different gene strings. There are chances that some genes may survive if the environment changes. The chance to break premature convergence is by using mutation operators. 15.5.4 Phenotype and Genotype Phenotype describes what an individual looks like, after fitness function calculation. Genotype is used to describe genes on chromosomes. Individuals with the same phenotype may have totally different genotypes. Genotypes maintain diversity for chance of changes. Populations with the same genotype individuals can still be evolved using the mutation operator. If the mutation operator is too strong then too much diversity will be introduced in genotype and then no population would be able to converge. If the mutation is too weak there is less chance to change and premature convergence becomes easy. 15.5.5 Exploitation and Exploration Exploitation in general takes the current search information from the experience of the last search to guide the search toward the direction that might be close to the best solutions. That is in other words exploitation allows focusing in areas of more promising solutions. Exploitation is achieved through selection operator and crossover operator. Exploration on the other hand widens the search to reach all possible solutions around the search space. In other words exploration allows the entire search space to be explored. Exploration is achieved through mutation operator and crossover operator.
Reinforcement Learning An important aspect of the genetic approach is the balancing between exploitation and exploration. High exploitation leads to premature convergence and high exploration results in non-convergence and sometimes may not lead to a fitter solution. 15.6

98% **MATCHING BLOCK 155/189** W

Genetic Programming Genetic programming (GP) is a form of evolutionary computation in which the individuals in the evolving population are computer programs rather than bit strings. The basic genetic programming approach presents a broad range of simple programs that can be successfully learned by GP. Representing Programs Programs manipulated by a GP are typically represented by trees corresponding to the parse tree of the program. Each function call is represented by a node in the tree, and the arguments to the function are given by its descendant nodes. For example, below Figure 15.2 illustrates this tree representation for the function . To apply genetic programming to a particular domain, the user must define the primitive functions to be considered (e.g., sin, cos, √ , +, -, exponential~), as well as the terminals (e.g., x, y, constants such as 2).

Figure 15.2 Program tree representation in genetic programming.

100% **MATCHING BLOCK 156/189** W

The genetic programming algorithm then uses an evolutionary search to explore the vast space of programs that can be described using these primitives. As in a genetic algorithm, the prototypical genetic programming algorithm maintains a population of individuals (in this case, program trees). On each iteration, it produces a new generation of individuals using selection, crossover, and mutation. The fitness of a given individual program in the population is typically determined by executing the program on a set of training data. Crossover

Reinforcement Learning

100% **MATCHING BLOCK 157/189** W

operations are performed by replacing a randomly chosen subtree of one parent program by a subtree from the other parent program (

as shown in figure 15.3). Figure 15.3 Crossover operation applied to two

93% **MATCHING BLOCK 158/189** W

parent program trees Figure 15.3 illustrates a typical crossover operation. It describes a set of experiments applying a GP to a number of applications. In

the experiments by Koza (1992), 10%

100%	MATCHING BLOCK 159/189	W

of the current population, selected probabilistically according to fitness, is retained unchanged in the next generation. The remainder of the new generation is created by applying crossover to pairs of programs from the current generation, again selected probabilistically according to their fitness. The mutation operator was not used in this particular set of experiments. 15.7

Summary ● Genetic algorithms provide an approach to learning that is based loosely on simulated evolution. ● One of the main disadvantages of genetic algorithms is that there is no guarantee for optimal solution within a finite time or in other words they lack the killer instinct. ●

100%	MATCHING BLOCK 160/189	W

GAS can search spaces of hypotheses containing complex interacting parts,

Reinforcement Learning

100%	MATCHING BLOCK 161/189	W

where the impact of each part on overall hypothesis fitness may be difficult to model. ●

100%	MATCHING BLOCK 162/189	W

Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover. ● The

100%	MATCHING BLOCK 163/189	W

The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population. 15.8

Key Terms ● Tournament selection: In this method,

100%	MATCHING BLOCK 164/189	W

two hypotheses are first chosen at random from the current population. ●

Rank

80%	MATCHING BLOCK 165/189	W

selection: In this method, the hypotheses in the current population are first sorted by fitness. ●

Crowding: It

100%	MATCHING BLOCK 166/189	W

is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of this individual and very similar individuals take over a large fraction of the population. ●

Phenotype: It describes what an individual looks like, after fitness function calculation. ● Genotype: It is used to describe genes on chromosomes. 15.9 Check Your Progress Short-Answer Type Q1) In ________ method,

100%	MATCHING BLOCK 167/189	W

the hypotheses in the current population are first sorted by fitness.

Q2) ________ widens the search to reach all possible solutions around the search space. Q3) _________ combines bits sampled uniformly from the two parents. Q4) Exploitation allows focusing in areas of more promising solutions. (True/ False?) Q5) High exploitation leads to premature convergence and high exploration results in non-convergence and sometimes may not lead to a fitter solution. (True/ False?) Long-Answer Type Q1) Explain how a program is represented in Genetic Programming with an example.

Reinforcement Learning Q2) Differentiate between Exploitation and Exploration. Q3) Discuss the concept of Fitness function and Selection. Q4) Explain the genetic operators in detail. Q5) Differentiate between phenotype and genotype. References: Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997. Machine Learning: a Probabilistic Perspective, Kevin Patrick Murphy, MIT Press, March 2014. Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004.

Reinforcement Learning Unit 16 – Analytical and Ensemble Learning Structure 16.0 Introduction 16.1 Unit Objectives 16.2 Inductive and Analytical Learning Problems 16.3 Explanation- based Learning 16.4 Introduction to Ensemble learning 16.5 Techniques in Ensemble learning 16.5.1 Voting 16.5.2 Bagging 16.5.3 Boosting 16.6 Summary 16.7 Key Terms 16.8 Check Your Progress 16.0 Introduction So far we have learnt about the Inductive learning methods such as neural network and decision tree learning. These methods require a certain number of training examples to achieve a given level of generalization accuracy, as reflected in the theoretical bounds and experimental results. Analytical learning uses prior knowledge and deductive reasoning to augment the information provided by the training examples, so that it is not subject to these same bounds. This unit considers an analytical learning method called explanation-based learning (EBL). In explanation-based learning, prior knowledge is used to analyze, or explain, how each observed training example satisfies the target concept. This explanation is then used to distinguish the relevant features of the training example from the irrelevant, so that examples can be generalized based on logical rather than statistical reasoning. Explanation-based learning has been successfully applied to learning search control rules for a variety of planning and scheduling tasks. This unit considers explanation-based learning when the learner's prior knowledge is correct and complete. Another type of learning process, i.e. Ensemble learning is also introduced in this unit. Ensemble Learning is a technique that creates multiple models and then combines

Reinforcement Learning

them to produce improved results. Ensemble learning usually produces more accurate solutions than a single model would. 16.1

Unit Objectives After completing this unit, the reader will be able to: ● Explain the basic concept of Analytical Learning. ● Discuss the Explanation- based learning model. ● Outline the concept of Ensemble Learning for classifiers. 16.2 Inductive and Analytical Learning Problems The essential difference between analytical and inductive learning methods is that they assume two different formulations of the learning problem: ● In inductive learning, the learner is given a hypothesis space H from which it must select an output hypothesis, and a set of training examples D = {(x 1 , f (x 1)) . . . , . (x n , f (x n))} where f (x i) is the target value for the instance x i . The desired output of the learner is a hypothesis h from H that is consistent with these training examples. ● In analytical learning, the input to the learner includes the same hypothesis space H and training examples D as for inductive learning. In addition, the learner is provided an additional input: A domain theory B consisting of background knowledge that can be used to explain observed training examples. The desired output of the learner is a hypothesis h from H that is consistent with both the training examples D and the domain theory B.

Reinforcement Learning Figure 16.1 Choice of inductive and analytical learning methods. In analytical learning, the learner must output a hypothesis that is consistent with both the training data and the domain theory. We say that hypothesis h is consistent with domain theory B provided B does not entail the negation of h (i.e., B -h). This additional constraint that the output hypothesis must be consistent with B reduces the ambiguity faced by the learner when the data alone cannot resolve among all hypotheses in H. The net effect, provided the domain theory is correct, is to increase the accuracy of the output hypothesis. For example, consider an instance space X in which each instance is a pair of physical objects. Each of the two physical objects in the instance is described by the predicates Color, Volume, Owner, Material, Type, and Density, and the relationship between the two objects is described by the predicate On. Given this instance space, the task is to learn the target concept "pairs of physical objects, such that one can be stacked safely on the other," denoted by the predicate SafeToStack (x,y). Learning this target concept might be useful, for example, to a robot system that has the task of storing various physical objects within a limited workspace. The full definition of this analytical learning task is given in Figure 16.1. As shown in Figure 16.1, we have chosen a hypothesis space H in which each hypothesis is a set of first-order if-then rules, or Horn clauses. For instance, the example Horn clause hypothesis shown in the table asserts that it is SafeToStack any object x on any object y, if the Volume of x is LessThan the Volume of y (in this Horn clause the variables vx and vy represent the volumes of x and y, respectively). Note the Horn clause hypothesis can refer to any of the predicates used to describe the instances, as well as several additional predicates and functions. A typical positive training example, SafeToStack (Obj1, Obj2), is also shown in the table. To formulate this task as an analytical learning problem we must also provide a domain theory sufficient to explain why observed positive examples satisfy the target concept. In this example, the domain theory must similarly explain why certain pairs of objects can be safely stacked on one another. The domain theory shown in the table includes assertions such as "it is safe to stack x on y if y is not Fragile," and "an object x is Fragile if the Material from which x is made is Glass."

Reinforcement Learning Figure 16.2 An analytical learning problem: SafeToStack(x,y) (Source: Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997.) Perfect Domain Theories: PROLOG-EBG A domain theory is said to be correct if each of its assertions is a truthful statement about the world. A domain theory is said to be complete with respect to a given target concept and instance space, if the domain theory covers every positive example in the instance space. In other words, it is complete if every instance that satisfies the target concept can be proven by the domain theory to satisfy it. It should be noted that our definition of completeness does not require that the domain theory be able to prove that negative examples do not satisfy the target concept. However, if we follow the usual PROLOG convention that unprovable assertions are assumed to be false, then
Reinforcement Learning this definition of completeness includes full coverage of both positive and negative examples by the domain theory. PROLOG-EBG conducts a detailed analysis of individual training examples to determine how best to generalize from the specific example to a general Horn clause hypothesis. The following are the key properties of this algorithm. ● Unlike inductive methods, PROLOG-EBG produces justified general hypotheses by using prior knowledge to analyze individual examples. ● The explanation of how the example satisfies the target concept determines which example attributes are relevant: those mentioned by the explanation. ● The further analysis of the explanation, regressing the target concept to determine its weakest preimage with respect to the explanation, allows deriving more general constraints on the values of the relevant features. ● Each learned Horn clause corresponds to a sufficient condition for satisfying the target concept. The set of learned Horn clauses covers the positive training examples encountered by the learner, as well as other instances that share the same explanations. ● The generality of the learned Horn clauses will depend on the formulation of the domain theory and on the sequence in which training examples are considered. ● PROLOG-EBG implicitly assumes that the domain theory is correct and complete. If the domain theory is incorrect or incomplete, the resulting learned concept may also be incorrect. Table 16.1 Comparison of inductive learning and analytical learning Features Inductive Learning Analytical Learning Objective Formulate general hypotheses that fit observed training data Formulate general hypotheses that fit domain theory Justification Statistical inference Deductive inference Advantage Require no prior knowledge Learn from scarce data Disadvantage Can fail if there exist scarce data, or incorrect inductive bias Can be misled when given incorrect or insufficient domain theory
Reinforcement Learning Method DT (Decision Tree), NN (Neural Networks), GA (Genetic Algorithms), ILP (Inductive Logic Programming) AL (Analytical Learning), EBL (Explanation based learning) 16.3 Explanation- based Learning Explanation-based learning is a form of analytical learning in which the learner processes each novel training example by: ● Explaining the observed target value for this example in terms of the domain theory ● Analyzing this explanation to determine the general conditions under which the explanation holds ● Refining its hypothesis to incorporate these general conditions. Explanation-Based learning (EBL) is a technique by which an intelligent system can learn by observing examples. EBL systems are characterized by the ability to create justified generalizations from single training instances. They are also distinguished by their reliance on background knowledge of the domain under study. Although EBL is usually viewed as a method for performing generalization, it can be viewed in other ways as well. In particular, EBL can be seen as a method that performs four different learning tasks: generalization, chunking, operationalization and analogy. Explanation-based learning has only recently emerged as a recognizable area of study. Consequently, most early EBL research was undertaken by investigators who were not working on EBL. EBL may be viewed as a convergence of several distinct lines of research within machine learning. In particular. EBL has developed out of efforts to address each of the following problems: ● Justified Generalization: A logically sound procedure for generalizing from examples. Given some initial background knowledge B and a set of training examples T, justified generalization finds a concept C that includes all the positive examples and excludes all the negative examples. The learned concept C must be a logical consequence of the background knowledge B and the training example set T (Russell, 1986). ● Chunking: In the context of explanation-based learning, chunking is a process
Reinforcement Learning of compiling a linear or tree-structured sequence of operators into a single operator. The Single operator has the same effect as the entire original sequence (Rosenbloom and Newell, 1986). ● Operationalization: A process of translating a non-operational expression into an operational form. The initial non-operational expression may be a set of instructions or a concept. Concepts and instructions are considered to be operational with respect to an agent if they are expressed in terms of actions and data available to the agent. ● Justified Analogy: A logically sound procedure for reasoning by analogy. Given some initial background knowledge B, an analogue example A and a target example B, find a feature F such that F(A) and infer that F(B). The conclusion F(B) must be a logical consequence of F(A) the background knowledge B (Davies and Russell, 1986). 16.4 Introduction to Ensemble learning

90%	**MATCHING BLOCK 169/189**	W

Ensemble Learning is a technique that creates multiple models and then combines them to produce improved results. Ensemble learning usually produces more accurate solutions than a single model would. Ensemble learning methods are applied to regression as well as classification: ● Ensemble learning for regression creates multiple repressors i.e. multiple regression models such as linear, polynomial, etc. ● Ensemble learning for classification creates multiple classifiers i.e. multiple classification models such as logistic, decision trees, KNN, SVM, etc. Figure 16.3 Ensemble learning view

Reinforcement Learning

86%	MATCHING BLOCK 170/189	W

There are also different ways the multiple base-learners are combined to generate the final output: ● Multiexpert combination: These methods have base-learners that work in parallel. These methods can in turn be divided into two:

a)

96%	MATCHING BLOCK 171/189	W

In the global approach, also called learner fusion, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking. b) In the local approach, or learner selection, for example, in

a

97%	MATCHING BLOCK 172/189	W

mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output. ● Multistage combination: These methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. The idea is that the base-learners (or the different representations they use) are sorted in increasing complexity so that a complex base-learner is not used (or its complex representation is not extracted) unless the preceding simpler base-learners are not confident. An example is cascading. Let us say that we have L base-learners. We denote by d j (x) the prediction of base- learner M j given the arbitrary dimensional input x. In the case of multiple representations, each M j uses a different input representation x j . The final prediction is calculated from the predictions of the base-learners: y = f (d 1 , d 2 , . . . , d L |Φ) where f (·) is the combining function with Φ denoting its parameters. Figure 16.4 Base-learners are

d j and their outputs are combined using f (·)
Reinforcement Learning

96%	MATCHING BLOCK 173/189	W

When there are K outputs, for each learner there are d ji (x), i = 1, . . . , K, j = 1, . . . , L, and, combining them, we also generate K values, y i , i = 1, . . . , K and then for example in classification, we choose the class with the maximum y i value:

Choose C i if Figure 16.4 depict the

95%	MATCHING BLOCK 174/189	W

base-learners are d j and their outputs are combined using f (·). This is for a single output; in the case of classification, each base-learner has K outputs that are separately used to calculate y i , and then we choose the maximum. Note that here all learners observe the same input; it may be the case that different learners observe different representations of the same input object or event. 16.5

Techniques in Ensemble learning Various methods are used in ensemble learning to combine the models. They include

100%	MATCHING BLOCK 175/189	W

voting, Error-Correcting Output Codes, Bagging: Random Forest Trees, Boosting: Adaboost, Stacking.

We will discuss some of them in this section. 16.5.1

92%	MATCHING BLOCK 176/189	W

Voting The simplest way to combine multiple classifiers is by voting, which corresponds to taking a linear combination of the learners (see figure 16.4): This is also known as ensembles and linear opinion pools. In the simplest case, all learners are given equal weight and we have simple voting that corresponds to taking an average. Still, taking a (weighted) sum is only one of the possibilities and there are also other combination rules, as shown in table 16.2 (

Kittler et al. 1998).

100% **MATCHING BLOCK 177/189** **W**

If the outputs are not posterior probabilities, these rules require that outputs be normalized to the same scale (

Jain, Nandakumar, and Ross, 2005).
Reinforcement Learning

98% **MATCHING BLOCK 178/189** **W**

Table 16.2 Classifier combination rules An example of the use of these rules is shown in table 16.3, which demonstrates the effects of different rules. Sum rule is the most intuitive and is the most widely used in practice. Median rule is more robust to outliers; minimum and maximum rules are pessimistic and optimistic, respectively. With the product rule, each learner has veto power; regardless of the other ones, if one learner has an output of 0, the overall output goes to 0. Note that after the combination rules, y i do not necessarily sum up to 1. Table 16.3 Example of combination rules on three learners and three classes In weighted sum, d ji is the vote of learner j for class C i and w j is the weight of its vote. Simple voting is a special case where all voters have equal weight, namely,

w j = 1/

100% **MATCHING BLOCK 179/189** **W**

L. In classification, this is called plurality voting where the class having the maximum number of votes is the winner. When there are two classes, this is majority voting where the winning class gets more than half of the votes. If the voters can also supply the additional information of how much they vote for each class (e.g., by the posterior probability), then after normalization, these can be used as weights in a weighted voting scheme. Equivalently, if d ji are the class posterior probabilities, P(C i | x,M j), then we can just

Reinforcement Learning

100% **MATCHING BLOCK 180/189** **W**

sum them up (w j = 1/L) and choose the class with maximum y i . In the case of regression, simple or weighted averaging or median can be used to fuse the outputs of base-regressors. Median is more robust to noise than the average. Another possible way to find w j is to assess the accuracies of the learners (regressor or classifier) on a separate validation set and use that information to compute the weights, so that we give more weights to more accurate learners. Voting schemes can be seen as approximations under a Bayesian framework with weights approximating prior model probabilities, and model decisions approximating model-conditional likelihoods. Simple voting corresponds to a uniform prior. If we have a prior distribution preferring simpler models, this would give larger weights to them. We cannot integrate over all models; we only choose a subset for which we believe P(M j) is high, or we can have another Bayesian step and calculate P(C i | x,M j), the probability of a model given the sample, and sample high probable models from this density. Let us assume that d j are iid with expected value E[d j] and variance Var(d j), then when we take a simple average with w j = 1/L, the expected value and variance of the output are We see that the expected value does not change, so the bias does not change. But variance, and therefore mean square error, decreases as the number of independent voters, L, increases. In the general case, which implies that if learners are positively correlated, variance (and error) increase. We can thus view using different algorithms and input features as efforts to decrease,

Reinforcement Learning if not completely eliminate, the positive correlation. 16.5.2
Bagging
Bagging is a voting method whereby base-learners are made different by training them over slightly different training sets. Generating L slightly different samples from a given sample is done by bootstrap, where given a training set X of size N, we draw N instances randomly from X with replacement. Because sampling is done with replacement, it is possible that some instances are drawn more than once and that certain instances are not drawn at all. When this is done to generate L samples
X j ,
j = 1, . . . , L, these samples are similar because they are all drawn from the same original sample, but they are also slightly different due to chance. The base-learners
d j
are trained with these L samples X j .

A learning algorithm is an unstable algorithm if small changes in the training set causes a large difference in the generated learner, namely, the learning algorithm has high variance.
Bagging, short for bootstrap aggregating, uses bootstrap to generate L training sets, trains L
base-learners
using an unstable learning procedure, and then, during testing, takes an average (Breiman 1996). Bagging can be used both for classification and regression. In the case of regression, to be more robust, one can take the median instead of the average when combining predictions.
We saw before that averaging reduces variance only if the positive correlation is small; an algorithm is stable if different runs of the same algorithm on resampled versions of the same dataset lead to learners with high positive correlation.
Algorithms such as decision trees and multilayer perceptrons are unstable.
Nearest neighbor is stable, but the condensed nearest neighbor is unstable (Alpaydin 1997). If the original training set is large, then we may want to generate smaller sets of size N′ &gt; N from them using bootstrap, since otherwise the bootstrap replicates X j will be too similar, and d j will be highly correlated. 16.5.3
Boosting

In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method. In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the Reinforcement Learning previous learners. The original boosting algorithm combines three weak learners to generate a strong learner. A weak learner has error probability less than 1/2, which makes it better than random guessing on a two-class problem, and a strong learner has arbitrarily small error probability. Boosting Concept Given a large training set, we randomly divide it into three. We use X 1 and train d 1 . We then take X 2 and feed it to d 1 . We take all instances misclassified by d 1 and also as many instances on which d 1 is correct from X 2 , and these together form the training set of d 2 . We then take X 3 and feed it to d 1 and d 2 . The instances on which d 1 and d 2 disagree form the training set of d 3 . During testing, given an instance, we give it to d 1 and d 2 ; if they agree, that is the response, otherwise the response of d 3 is taken as the output.

Schapire (1990) has shown that this

overall system has reduced error rate, and the error rate can arbitrarily be reduced by using such systems recursively, that is, a boosting system of three models used as

d j

in a higher system. Though it is quite successful, the disadvantage of the original boosting method is that it requires a very large training sample. The sample should be divided into three and furthermore, the second and third classifiers are only trained on a subset on which the previous ones err. So unless one has a quite large training set, d 2 and d 3 will not have training sets of reasonable size. AdaBoost Freund and Schapire (1996) proposed a variant, named AdaBoost, short for adaptive boosting, that uses the same training set over and over and thus need not be large, but the classifiers should be simple so that they do not overfit. AdaBoost can also combine an arbitrary number of baselearners, not three. The idea is to modify the probabilities of drawing the instances as a function of the error. Let us say p t j denotes the probability that the instance pair (x t , r t) is drawn to train the

j th

base-learner. Initially, all p t 1 = 1/N. Then we add new base-learners as follows, starting from j = 1: Є j denotes the error rate of d j .

Reinforcement Learning Figure 16.5 AdaBoost Algorithm

99% **MATCHING BLOCK 185/189** **W**

AdaBoost requires that learners are weak, that is, Є j &gt; 1/2,∀j; if not, we stop adding new base learners. Note that this error rate is not on the original problem but on the dataset used at step j. We define βj = Є j /(1 − Є j) &gt; 1, and we set p t j +1 = β j p t j if d j correctly classifies xt ; otherwise, p t j+1 = p t j . Because p t j+1 should be probabilities, there is a normalization where we divide p t j+1 by ∑t

100% **MATCHING BLOCK 186/189** **W**

p t j+1 , so that they sum up to 1. This has the effect that the probability of a correctly classified instance is decreased, and the probability of a misclassified instance increases. Then a new sample of the same size is drawn from the original sample according to these modified probabilities, p t j+1 with replacement, and is used to train d j+1 . This has the effect that d j+1 focuses more on instances misclassified by d j ; that is why the base-learners are chosen to be simple and not accurate, since otherwise the next training sample would contain only a few outlier and noisy instances repeated many times over. For example, with decision trees, decision stumps, which are trees grown only one or two levels, are used. So it is clear that these would have bias but the decrease in variance is larger and the overall error decreases. An algorithm like the linear discriminant has low variance, and we cannot gain by AdaBoosting linear discriminants.

Reinforcement Learning 16.6 Summary ● Analytical learning uses prior knowledge and deductive reasoning to augment the information provided by the training examples, so that it is not subject to these same bounds. ●

79% **MATCHING BLOCK 187/189** **W**

Ensemble Learning is a technique that creates multiple models and then combines them to produce improved results. ●

A hypothesis space H in which each hypothesis is a set of first-order if-then rules, or Horn clauses. ● A domain theory is said to be correct if each of its assertions is a truthful statement about the world. A domain theory is said to be complete with respect to a given target concept and instance space, if the domain theory covers every positive example in the instance space. ● Unlike inductive methods, PROLOG-EBG produces justified general hypotheses by using prior knowledge to analyze individual examples. ● Explanation-Based learning (EBL) is a technique by which an intelligent system can learn by observing examples. 16.7 Key Terms ● Chunking: It is a process of compiling a linear or tree-structured sequence of operators into a single operator. ● Operationalization: A process of translating a non-operational expression into an operational form. ● Justified Analogy: A logically sound procedure for reasoning by analogy. ●
Multistage combination: These

100% **MATCHING BLOCK 188/189** **W**

methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. ●

Bagging: It
is a voting method whereby base-learners are made different by training them over slightly different training sets. 16.8 Check Your Progress Short-Answer Type Q1) Global approach and Local approach are the methods for _________. Reinforcement Learning Q2) Full form of EBL- a) Explanation Biased Learning b) Exploitation Biased Learning c) Explanation Based Learning d) Exploitation Based Learning Q3) A domain theory is said to be ________ if each of its assertions is a truthful statement about the world. Q4) In classification, the single

100% **MATCHING BLOCK 189/189** **W**

class having the maximum number of votes is the winner.

This is called _______. Long-Answer Type Q1) Write a short note on Ensemble Learning. Q2) Discuss the Explanation- based learning model for Analytical Learning. Q3) Explain the concept of Voting, Bagging, and Boosting. Q4) Differentiate between Inductive and Analytical Learning. Q5) State the key properties of PROLOG-EBG Algorithm. References: Machine Learning and Software Engineering, Du Zhang, Jeffrey J.P. Tsai, Software Quality Journal, 11, 87–119, 2003. Explanation-Based Learning: A Survey of Programs and Perspectives, Thomas Ellman, May 1987. Explanation-Based Learning: Its Role in Problem Solving, Ian Witten, Journal of Experimental & Theoretical Artificial Intelligence, January 1989. Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004. Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997.

Reinforcement Learning Unit 17 – Design and Analysis of Machine Learning Experiments Structure 17.0 Introduction 17.1 Unit Objectives 17.2 Factors, Response, and Strategy of Experimentation 17.3 Response Surface Design 17.4 Randomization, Replication, and Blocking 17.5 Guidelines for Machine Learning Experiments 17.6 Summary 17.7 Key Terms 17.8 Check Your Progress 17.0 Introduction We discuss the design of machine learning experiments to assess and compare the performances of learning algorithms in practice and the statistical tests to analyze the results of these experiments. We have already discussed several learning algorithms and saw that, given a certain application, more than one is applicable. Now, we are concerned with two questions: 1. How can we assess the expected error of a learning algorithm on a problem? 2. Given two learning algorithms, how can we say one has less error than the other one, for a given application? The algorithms compared can be different, for example, parametric versus nonparametric, or they can use different hyperparameter settings. The error rate on the training set, by definition, is always smaller than the error rate on a test set containing instances unseen during training. Similarly, training errors cannot be used to compare two algorithms. This is because over the training set, the more complex model having more parameters will almost always give fewer errors than the simple one. So as we have repeatedly discussed, we need a validation set that is different from the
Reinforcement Learning training set. Even over a validation set though, just one run may not be enough. There are two reasons for this: ● The training and validation sets may be small and may contain exceptional instances, like noise and outliers, which may mislead us. ● The learning method may depend on other random factors affecting generalization.
We use a learning algorithm on a dataset and generate a learner. If we do the training once, we have one learner and one validation error. To average over randomness (in training data, initial weights, etc.), we use the same algorithm and generate multiple learners. We test them on multiple validation sets and record a sample of validation errors. (
Of course, all the training and validation sets should be drawn from the same application.)
We base our evaluation of the learning algorithm on the distribution of these validation errors. We can use this distribution for assessing the expected error of the learning algorithm for that problem, or compare it with the error rate distribution of some other learning algorithm.
In general, we compare learning algorithms by their error rates, but it should be kept in mind that in real life, error is only one of the criteria that affect our decision. Some other criteria are (Turney 2000): ● Risks when errors are generalized using loss functions, instead of 0/1
loss. ● Training time and space complexity. ● Testing time and space complexity. ● Interpretability, namely, whether the method allows knowledge extraction which can be checked and validated by experts. ● Easy programmability.
The relative importance of these factors changes depending on the application. For example, if the training is to be done once in the factory, then training time and space complexity are not important; if adaptability during use is required, then they do become important. Most of the learning algorithms use 0/1 loss and take error as the single criterion to be minimized; recently, cost-sensitive learning variants of these algorithms have also been proposed to take other cost criteria into account.
Reinforcement Learning When we train a learner on a dataset using a training set and test its accuracy on some validation set and try to draw conclusions, what we are doing is experimentation. Statistics defines a methodology to design experiments correctly and analyze the collected data in a manner so as to be able to extract significant conclusions (Montgomery 2005). In this unit, we will see how this methodology can be used in the context of machine learning. 17.1 Unit Objectives After completing this unit, the reader will be able to: ● Explain the basic factors, response, and strategy of experimentation. ● Discuss the response surface design. ● Outline the concept of Randomization, Replication, and Blocking. ● Illustrate the guidelines for Machine Learning Experiments. 17.2 Factors, Response, and Strategy of Experimentation As in other branches of science and engineering, in machine learning too, we do experiments to get information about the process under scrutiny. In our case, this is a learner, which, having been trained on a dataset, generates an output for a given input. An experiment is a test or a series of tests where we play with the factors that affect the output. These factors may be the algorithm used, the training set, input features, and so on, and we observe the changes in the response to be able to extract information. The aim may be to identify the most important factors, screen the unimportant ones, or find the configuration of the factors that optimizes the response—for example, classification accuracy on a given test set. The aim is to plan and conduct machine learning experiments and analyze the data resulting from the experiments, to be able to eliminate the effect of chance and obtain conclusions which we can consider statistically significant. In machine learning, we target a learner having the highest generalization accuracy and the minimal complexity (so that its implementation is cheap in time and space) and is robust, that is, minimally affected by external sources of variability.

Reinforcement Learning Figure 17.1 The process generates an output given an input and is affected by controllable and uncontrollable factors. A trained learner can be shown as in figure 17.1; it gives an output, for example, a class code for a test input, and this depends on two types of factors. The controllable factors, as the name suggests, are those we have control on. The most basic is the learning algorithm used. There are also the hyperparameters of the algorithm, for example, the number of hidden units for a multilayer perceptron, k for k-nearest neighbor, C for support vector machines, and so on. The dataset used and the input representation, that is, how the input is coded as a vector, are other controllable factors. There are also uncontrollable factors over which we have no control, adding undesired variability to the process, which we do not want to affect our decisions. Among these are the noise in the data, the particular training subset if we are resampling from a large set, randomness in the optimization process, for example, the initial state in gradient descent with multilayer perceptrons, and so on. We use the output to generate the response variable—for example, average classification error on a test set, or the expected risk using a loss function, or some other measure, such as precision and recall, as we will discuss shortly. There are several strategies of experimentation, as shown in figure 17.2. In the best guess approach, we start at some setting of the factors that we believe is a good configuration. We test the response there and we fiddle with the factors one (or very few) at a time, testing each combination until we get to a state that we consider is good enough. If the experimenter has a good intuition of the process, this may work well;
Reinforcement Learning but note that there is no systematic approach to modify the factors and when we stop, we have no guarantee of finding the best configuration. Figure 17.2 Different strategies of experimentation with two factors and five levels each Another strategy is to modify one factor at a time where we decide on a baseline (default) value for all factors, and then we try different levels for one factor while keeping all other factors at their baseline. The major disadvantage of this is that it assumes that there is no interaction between the factors, which may not always be true. In the PCA/k-NN cascade, each choice for d defines a different input space for k- NN where a different k value may be appropriate. The correct approach is to use a factorial design where factors are varied together, instead of one at a time; this is colloquially called grid search. With F factors at L levels each, searching one factor at a time takes O(L · F) time, whereas a factorial experiment takes O(L F) time. 17.3 Response Surface Design To decrease the number of runs necessary, one possibility is to run a fractional factorial design where we run only a subset, another is to try to use knowledge gathered from previous runs to estimate configurations that seem likely to have high response. In searching one factor at a time, if we can assume that the response is typically quadratic (with a single maximum, assuming we are maximizing a response value, such as the test accuracy), then instead of trying all values, we can have an iterative procedure where starting from some initial runs, we fit a quadratic, find its maximum analytically, take that as the next estimate, run an experiment there, add the resulting data to the sample, and then continue fitting and sampling, until we get
Reinforcement Learning no further improvement. With many factors, this is generalized as the response surface design method where we try to fit a parametric response function to the factors as r = g(f 1 , f 2 , . . . , f F |Φ) where r is the response and f i , i = 1, . . . , F are the factors. This fitted parametric function defined given the parameters Φ is our empirical model estimating the response for a particular configuration of the (controllable) factors; the effect of uncontrollable factors is modeled as noise. g(·) is a (typically quadratic) regression model and after a small number of runs around some baseline (as defined by a so- called design matrix), one can have enough data to fit g(·) on. Then, we can analytically calculate the values of f i where the fitted g is maximum, which we take as our next guess, run an experiment there, get a data instance, add it to the sample, fit g once more, and so on, until there is convergence. Whether this approach will work well or not depends on whether the response can indeed be written as a quadratic function of the factors with a single maximum. 17.4 Randomization, Replication, and Blocking Let us now talk about the three basic principles of experimental design. Randomization requires that the order in which the runs are carried out should be randomly determined so that the results are independent. This is typically a problem in real-world experiments involving physical objects; for example, machines require some time to warm up until they operate in their normal range so tests should be done in random order for time not to bias the results. Ordering generally is not a problem in software experiments. Replication implies that for the same configuration of (controllable) factors, the experiment should be run a number of times to average over the effect of uncontrollable factors. In machine learning, this is typically done by running the same algorithm on a number of resampled versions of the same dataset; this is known as cross-validation. Blocking is used to reduce or eliminate the variability due to nuisance factors that influence the response but in which we are not interested. For example, defects produced in a factory may also depend on the different batches of raw material, and this effect should be isolated from the controllable factors in the factory, such as the

Reinforcement Learning equipment, personnel, and so on. In machine learning experimentation, when we use resampling and use different subsets of the data for different replicates, we need to make sure that for example if we are comparing learning algorithms, they should all use the same set of resampled subsets, otherwise the differences in accuracies would depend not only on the algorithms but also on the different subsets—to be able to measure the difference due to algorithms only, the different training sets in replicated runs should be identical; this is what we mean by blocking. In statistics, if there are two populations, this is called pairing and is used in paired testing. 17.5 Guidelines for Machine Learning Experiments Before we start experimentation, we need to have a good idea about what it is we are studying, how the data is to be collected, and how we are planning to analyze it. The steps in machine learning are the same as for any type of experimentation (Montgomery 2005). It should be noted that at this point, it is not important whether the task is classification or regression, or whether it is an unsupervised or a reinforcement learning application. 1. Aim of the Study The initial step is to state the problem clearly, define what the objectives are. In machine learning, there may be several possibilities. Given two learning algorithms and a particular problem defined by a dataset, we may want to determine which one has less generalization error. These can be two different algorithms, or one can be a proposed improvement of the other, for example, by using a better feature extractor. In the general case, we may have more than two learning algorithms, and we may want to choose the one with the least error, or order them in terms of error, for a given dataset. In an even more general setting, instead of on a single dataset, we may want to compare two or more algorithms on two or more datasets. 2. Selection of the Response Variable The next step is to decide on what we should use as the quality measure. Most frequently, error is used that is the misclassification error for classification and mean square error for regression. We may also use some variant; for example, generalizing from 0/1 to an arbitrary loss, we may use a risk measure. In a
Reinforcement Learning cost-sensitive setting, not only the output but also system parameters, for example, its complexity, are taken into account. 3. Choice of Factors and Levels After selecting the response variable, the next step is to choose what are the factors that depend on the aim of the study. ● If we fix an algorithm and want to find the best hyperparameters, then those are the factors. ● If we are comparing algorithms, the learning algorithm is a factor. ● If we have different datasets, they also become a factor. The levels of a factor should be carefully chosen so as not to miss a good configuration and avoid doing unnecessary experimentation. It is always good to try to normalize factor levels. Though previous expertise is a plus in general, it is also important to investigate all factors and factor levels that may be of importance and not be overly influenced by past experience. 4. Choice of Experimental Design It is always better to do a factorial design unless we are sure that the factors do not interact, because mostly they do. Replication number depends on the dataset size; it can be kept small when the dataset is large; However, too few replicates generate few data and this will make comparing distributions difficult; in the particular case of parametric tests, the assumptions of Gaussianity may not be tenable. Generally, given some dataset, we leave some part as the test set and use the rest for training and validation, probably many times by resampling. 5. Performing the Experiment Before running a large factorial experiment with many factors and levels, it is best if one does a few trial runs for some random settings to check that all is as expected. In a large experiment, it is always a good idea to save intermediate results (or seeds of the random number generator), so that a part of the whole experiment can be rerun when desired. All the results should be reproducible. In running a large experiment with many factors and factor levels, one should be aware of the possible negative effects of software aging. It is important that an experimenter be unbiased during experimentation. In comparing one's favorite algorithm with a competitor, both should be
Reinforcement Learning investigated equally diligently. In large-scale studies, it may even be envisaged that testers be different from developers. One should avoid the temptation to write one's own "library" and in-stead, as much as possible, use code from reliable sources; such code would have been better tested and optimized. 6. Statistical Analysis of the Data This corresponds to analyzing data in a way so that whatever conclusion we get is not subjective or due to chance. We cast the questions that we want to answer in the framework of hypothesis testing and check whether the sample supports the hypothesis. For example, the question "Is A a more accurate algorithm than B?" becomes the hypothesis "Can we say that the average error of learners trained by A is significantly lower than the average error of learners trained by B?" We can use histograms of error distributions, whisker-and-box plots, range plots, and so on. 7. Conclusions and Recommendations Once all data is collected and analyzed, we can draw objective conclusions. One frequently encountered conclusion is the need for further experimentation. Most statistical, and hence machine learning or data mining, studies are iterative. It is for this reason that we never start with all the experimentation. It is suggested that no more than 25 percent of the available resources should be invested in the first experiment (Montgomery 2005). The first runs are for investigation only. That is also why it is a good idea not to start with high expectations, or promises to one's boss or thesis advisor. 17.6 Summary ● The training and validation sets may be small and may contain exceptional instances, like noise and outliers, which may mislead us. ● The learning method may depend on other random factors affecting generalization. ● An experiment is a test or a series of tests where we play with the factors that affect the output. ● In machine learning, we target a learner having the highest generalization accuracy and the minimal complexity. ● Replication implies that for the same configuration of (controllable) factors, the

Reinforcement Learning experiment should be run a number of times to average over the effect of uncontrollable factors. 17.7 Key Terms ● Controllable Factors: The factors which we can control. ● Cross-Validation: Running the same algorithm on a number of resampled versions of the same dataset is known as cross-validation. ● Pairing: In statistics, if there are two populations, this is called pairing. ● Factorial design: The approach where factors are varied together, instead of one at a time. 17.8 Check Your Progress Short-Answer Type Q1) Three basic principles of experimental design are- a) Randomization, Replication, and Blocking b) Replication, Grid search, and Chunking c) Randomization, Blocking, and Chunking d) Replication, Blocking, and Grid search Q2) ________ is used to reduce or eliminate the variability due to nuisance factors that influence the response but in which we are not interested. Q3) The process of running the same algorithm on a number of resampled versions of the same dataset is- Q4) Before running a large factorial experiment with many factors and levels, it is best if one does a few trial runs for some random settings to check that all is as expected. (True/ False?) Long-Answer Type Q1) Briefly explain the three basic principles of Experimental Design. Q2) List the steps involved in Machine Learning Experiments. Q3) Define Factors, Response, and Strategy of Experimentation. Q4) Explain the concept of Response Surface Design. Q5) How factors and levels are chosen in Machine Learning Experiments?
Reinforcement Learning References: Introduction to Machine Learning, Ethem Alpaydin, The MIT Press, Cambridge, Massachusetts, 2004. Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997.

Hit and source - focused comparison, Side by Side

Submitted text	As student entered the text in the submitted document.
Matching text	As the text appears in the source.

1/189	**SUBMITTED TEXT** 34 WORDS	**98% MATCHING TEXT** 34 WORDS
	Type of training experience 3.5 Choosing the Target Function 3.5.1 Choosing a representation for the Target Function 3.5.2 Choosing an approximation algorithm for the Target Function 3.6 Final Design 3.7	Type of training experience 2. Choosing the Target Function 3. Choosing a representation for the Target Function 4. Choosing an approximation algorithm for the Target Function 5. The final Design

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

2/189	**SUBMITTED TEXT** 14 WORDS	**100% MATCHING TEXT** 14 WORDS
	Genetic Programming 15.7 Models of Evolution and Learning 15.8 Parallelizing Genetic Algorithms 15.9	Genetic Programming 101 Models of Evolution and Learning 104 Parallelizing Genetic Algorithms. 105 1

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

3/189	**SUBMITTED TEXT** 69 WORDS	**100% MATCHING TEXT** 69 WORDS
	Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data or both.	Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

4/189	SUBMITTED TEXT 39 WORDS	100% MATCHING TEXT 39 WORDS
	A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks T, as measured by P, improves with experience E. 3	A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

5/189	SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
	Handwriting recognition learning problem ● Task (T): Recognising and classifying handwritten words within images ● Performance	Handwriting recognition learning problem • Task T: Recognising and classifying handwritten words within images • Performance

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

6/189	SUBMITTED TEXT 46 WORDS	98% MATCHING TEXT 46 WORDS
	P): Percent of words correctly classified ● Training experience (E): A dataset of handwritten words with given classifications A robot driving learning problem ● Task (T): Driving on highways using vision sensors ● Performance measure (P): Average distance traveled before an error ● Training experience (	P: Percent of words correctly classified • Training experience E: A dataset of handwritten words with given classifications ii) A robot driving learning problem • Task T: Driving on highways using vision sensors • Performance measure P: Average distance traveled before an error • training experience:

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

7/189	SUBMITTED TEXT 72 WORDS	93% MATCHING TEXT 72 WORDS
	A sequence of images and steering commands recorded while observing a human driver A chess learning problem ● Task (T): Playing chess ● Performance measure (P): Percent of games won against opponents ● Training experience (E): Playing practice games against itself A computer program that learns from experience is called a machine learning program or simply a learning program. Sometimes such a program is also referred to as a learner.	A sequence of images and steering commands recorded while observing a human driver iii) A chess learning problem • Task T: Playing chess • Performance measure P: Percent of games won against opponents • Training experience E: Playing practice games against itself Definition A computer program which learns from experience is called a machine learning program or simply a learning program. Such a program is sometimes also referred to as a learner. 1.2

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

8/189	SUBMITTED TEXT 28 WORDS	44% MATCHING TEXT 28 WORDS
	Basic Components of Learning Process Either for humans or a machine, the learning process includes four components, data storage, abstraction, generalization, and evaluation. Figure 1.2 represents the	Basic components of learning process The learning process, whether by a human or a machine, can be divided into four components, namely, data storage, abstraction, generalization and evaluation. Figure 1.1 illustrates the

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

9/189 SUBMITTED TEXT 46 WORDS	56% MATCHING TEXT 46 WORDS
Learning Process 1. Data storage: Storing and retrieving large amounts of data are important components of the learning process. Both humans and computers utilize data storage as a foundation for advanced reasoning. Humans store the data in the brain and retrieve it using electrochemical signals.	learning process. 2 1. Data storage Facilities for storing and retrieving huge amounts of data are an important component of the learning process. Humans and computers alike utilize data storage as a foundation for advanced reasoning. • In a human being, the data is stored in the brain and is retrieved using electrochemical signals. •

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

10/189 SUBMITTED TEXT 28 WORDS	100% MATCHING TEXT 28 WORDS
computers use hard disk drives, flash memory, random access memory, and similar devices to store data and use cables and other technology to retrieve data. 2. Abstraction:	Computers use hard disk drives, flash memory, random access memory and similar devices to store data and use cables and other technology to retrieve data. 2. Abstraction

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

11/189 SUBMITTED TEXT 21 WORDS	91% MATCHING TEXT 21 WORDS
Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed." 1 1.	machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed."

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

12/189 SUBMITTED TEXT 101 WORDS	77% MATCHING TEXT 101 WORDS
of extracting knowledge about already stored data. This involves creating general concepts about the data with the help of known models and new models. Training is the process of fitting a model to a dataset. Once the model is trained, the data is transformed into an abstract form that summarizes the original information. 3. Generalization: The third component generalization of the learning process describes the process of turning the knowledge about stored data into a form that can be utilized for future action. These actions are to be carried out on tasks that are similar, but not identical, to those	of extracting knowledge about stored data. This involves creating general concepts about the data as a whole. The creation of knowledge involves application of known models creation of new The process of fitting a model to a dataset is known as training. the model has trained, the data is transformed into an abstract form that summarizes the original information. 3. Generalization The third component of the learning process is known as generalisation. The term generalization describes process of turning the knowledge about stored data into a form that can be utilized for future action. These actions are to be carried out on tasks that are similar, but not identical, to those

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

13/189 SUBMITTED TEXT 72 WORDS	92% MATCHING TEXT 72 WORDS
have been seen before. In generalization, the goal is to discover those properties of the data that will be most relevant to future tasks. 4. Evaluation: Evaluation, being the last component of the learning process, is the process of giving feedback to the user to measure the utility of the learned knowledge. This feedback is then utilized to effect improvements in the whole learning process. 1.4 Phases of Machine Learning	have been seen before. In generalization, the goal is to discover those properties of the data that will be most relevant to future tasks. 4. Evaluation Evaluation is the last component of the learning process. It is the process of giving feedback to the user to measure the utility of the learned knowledge. This feedback is then utilised to effect improvements in the whole learning process Applications of machine learning

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

14/189	SUBMITTED TEXT 24 WORDS	52% MATCHING TEXT 24 WORDS
	some parameters and learning depicts the execution of computer programs to optimize these parameters of the models using past experiences or training data.	some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

15/189	SUBMITTED TEXT 31 WORDS	71% MATCHING TEXT 31 WORDS
	Applications of Machine Learning Applying machine learning to large databases is known as data mining. In data mining, a large amount of data is processed to build a simple model	Applications of machine learning Application of machine learning methods to large databases is called data mining. In data mining, a large volume of data is processed to construct a simple model

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

16/189	SUBMITTED TEXT 28 WORDS	61% MATCHING TEXT 28 WORDS
	high prediction accuracy. Following is a list of some of the applications of machine learning. ● Machine learning is used in the retail business to study consumer	high predictive accuracy. The following is a list of some of the typical applications of machine learning. 1. In retail business, machine learning is used to study consumer

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

17/189	SUBMITTED TEXT 39 WORDS	77% MATCHING TEXT 39 WORDS
	finance, banks analyze their past data to build models for use in credit applications, fraud detection, and stock marketing. ● In manufacturing industries, training models are used for optimization, control, and troubleshooting. ● Learning programs are	finance, banks analyze their past data to build models use in credit applications, fraud detection, and the stock market. 3. In manufacturing, learning models are used for optimization, control, and troubleshooting. 3 4. In medicine, learning programs are

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

18/189	SUBMITTED TEXT 13 WORDS	100% MATCHING TEXT 13 WORDS
	used for medical diagnosis. ● In telecommunications, call patterns are analyzed	used for medical diagnosis. 5. In telecommunications, call patterns are analyzed

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

19/189	SUBMITTED TEXT 24 WORDS	87% MATCHING TEXT 24 WORDS
	In artificial intelligence, it is used to train a system to learn and adapt to changes so that the system designer does not	In artificial intelligence, it is used to teach a system to learn and adapt to changes so that the system designer need not

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

20/189	SUBMITTED TEXT 18 WORDS	66% MATCHING TEXT 18 WORDS
	applied in the design of computer-controlled vehicles to drive accurately when driving on various types of roads.	applied in the design of computer-controlled vehicles to steer correctly when driving on a variety of roads. 10.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

21/189	SUBMITTED TEXT 21 WORDS	91% MATCHING TEXT 21 WORDS
	A computer program that learns from experience is called a machine learning program or simply a learning program. ●	A computer program which learns from experience is called a machine learning program or simply a learning program.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

22/189	SUBMITTED TEXT 46 WORDS	97% MATCHING TEXT 46 WORDS
	Supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs. In supervised learning, each example in the training set is a pair consisting of an input object (typically a vector) and an output value.	Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. In supervised learning, each example in the training set is a pair consisting of an input object (typically a vector) and an output value.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

23/189	SUBMITTED TEXT 137 WORDS	97% MATCHING TEXT 137 WORDS
	A supervised learning algorithm analyzes the training data and produces a function, which can be used for mapping new examples. In the optimal case, the function will correctly determine the class labels for unseen instances. Both classification and regression problems are supervised learning problems. A wide range of supervised learning algorithms is available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems. A "supervised learning" is so-called because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers (that is, the correct outputs), the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.	A supervised learning algorithm analyzes the training data and produces a function, which can be used for mapping new examples. In the optimal case, the function will correctly determine the class labels for unseen instances. Both classification and regression 13 problems are supervised learning problems. A wide range of supervised learning algorithms available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised problems. Figure 1.4: learning Remarks A "supervised learning" is so called because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers (that is, the correct outputs), the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

24/189	SUBMITTED TEXT 43 WORDS	92% MATCHING TEXT 43 WORDS
	A training set of examples with the correct responses (targets) is provided and, based on this training set, the algorithm generalizes to respond correctly to all possible inputs. This is also called learning from exemplars. Figure 2.1 Supervised Learning	A training set of examples with the correct responses (targets) is provided and, based on this training set, the algorithm generalises to respond correctly to all possible inputs. This is also called learning from exemplars. Supervised learning

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

25/189	SUBMITTED TEXT 98 WORDS	100% MATCHING TEXT 98 WORDS
	Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions. Since the examples given to the learner are unlabeled, the accuracy of the structure that is output by the algorithm cannot be evaluated. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.	Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions. Since examples given to the learner are unlabeled, the accuracy of the structure that is output by the algorithm cannot be evaluated. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns 14 or grouping data.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

26/189	SUBMITTED TEXT 36 WORDS	95% MATCHING TEXT 36 WORDS
	correct responses, but instead the algorithm tries to identify similarities between the inputs so that inputs that have something in common are categorised together. The statistical approach to unsupervised learning is known as density estimation.	Correct responses are not provided, but instead the algorithm tries to identify similarities between the inputs so that inputs that have something in common are categorised together. The statistical approach to unsupervised learning is known as density estimation.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

27/189	SUBMITTED TEXT 16 WORDS	71% MATCHING TEXT 16 WORDS
	Unsupervised Learning Unsupervised learning is more applicable than supervised learning, since it does not	Unsupervised Learning o Unsupervised learning is intrinsically more difficult than supervised learning as it does not

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

28/189	SUBMITTED TEXT 78 WORDS	100% MATCHING TEXT 78 WORDS
	Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards. A learner (the program) is not told what actions to take as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situations and, through that, all subsequent rewards.	Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards. learner (the program) is not told what actions to take as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situations and, through that, all subsequent rewards.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

29/189	SUBMITTED TEXT 77 WORDS	98% MATCHING TEXT 77 WORDS
	example, consider teaching a dog a new trick: we cannot tell it what to do, but we can reward/punish it if it does the right/ wrong thing. It has to find out what it did that made it get the reward/ punishment. We can use a similar method to train computers to do many tasks, such as playing chess, scheduling jobs, and controlling robot limbs. Reinforcement learning is different from supervised learning. Supervised learning is	Example Consider teaching a dog a new trick: we cannot tell it what to do, but we can reward/punish it if it does the right/wrong thing. It has to find out what it did that made it get the reward/punishment. We can use a similar method to train computers to do many tasks, such as playing backgammon or chess, scheduling jobs, and controlling robot limbs. Reinforcement learning is different from supervised learning. Supervised learning is

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

30/189	SUBMITTED TEXT 26 WORDS	93% MATCHING TEXT 26 WORDS
	Supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs. ● Supervised learning	Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. supervised learning,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

31/189	SUBMITTED TEXT 27 WORDS	97% MATCHING TEXT 27 WORDS
	Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. ● Unsupervised learning	Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. In unsupervised learning

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

32/189	SUBMITTED TEXT 32 WORDS	98% MATCHING TEXT 32 WORDS
	Type of training experience 3.3.2 Choosing the Target Function 3.3.3 Choosing a representation for the Target Function 3.3.4 Choosing an approximation algorithm for the Target Function 3.3.5 Final Design 3.4	Type of training experience 2. Choosing the Target Function 3. Choosing a representation for the Target Function 4. Choosing an approximation algorithm for the Target Function 5. The final Design

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

33/189	SUBMITTED TEXT 37 WORDS	68% MATCHING TEXT 37 WORDS
	Machine learning is concerned with using the right features to build the right models that achieve the right tasks. The collection of all possible outcomes, for a given problem, represents the sample space or instance space.	Machine learning is concerned with using the right features to build the right models that achieve the right tasks. The basic idea of Learning models has divided into three categories. For a given problem, the collection of all possible outcomes represents the sample space or instance space. ?

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

34/189	SUBMITTED TEXT 28 WORDS	100% MATCHING TEXT 28 WORDS
	Using a logical expression (Logical Models) ● Using the Geometry of the instance space (Geometric models) ● Using Probability to classify the instance space (Probabilistic models)	Using a Logical expression. (Logical models) ? Using the Geometry of the instance space. (Geometric models) ? Using Probability to classify the instance space. (Probabilistic models) ?

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

35/189	SUBMITTED TEXT 78 WORDS	98% MATCHING TEXT 78 WORDS
	Logical Models These models use a logical expression to divide the instance space into segments and hence construct grouping models. A logical expression is an expression that returns a Boolean value, i.e., a True or False outcome. Once the data is grouped using a logical expression, the data is divided into homogeneous groupings for the problem we are trying to solve. For example, for a classification problem, all the instances in the group belong to one class.	Logical models models use a logical expression to divide the instance space into segments and hence construct grouping models. A logical expression is an expression that returns a Boolean value, i.e., a True or False outcome. Once the data is grouped using a logical expression, the data is divided into homogeneous groupings for the problem we are trying to solve. For example, for a classification problem, all the instances in the group belong to one class.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

36/189	SUBMITTED TEXT 53 WORDS	94% MATCHING TEXT 53 WORDS
	a logical expression is used to partition the instance space. Two instances are similar when they end up in the same logical segment. In geometric models, we consider models that define similarity by considering the geometry of the instance space. In Geometric models, features could be described as points in two	a logical expression is used to partition the instance space. Two instances are similar when they end up in the same logical segment. In this section, we consider models that define similarity by considering the geometry of the instance space. In Geometric models, features could be described as points in two

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

37/189	SUBMITTED TEXT 42 WORDS	100% MATCHING TEXT 42 WORDS
	dimensions (x- and y-axis) or a three-dimensional space (x, y, and z). Even when features are not intrinsically geometric, they could be modelled in a geometric manner (for example, temperature as a function of time can be modelled in two axes).	dimensions (x- and y-axis) or a three-dimensional space (x, y, and z). Even when features are not 5 intrinsically geometric, they could be modelled in a geometric manner (for example, temperature as a function of time can be modelled in two axes).

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

38/189	SUBMITTED TEXT 242 WORDS	100% MATCHING TEXT 242 WORDS
	In geometric models, there are two ways we could impose similarity: ● We could use geometric concepts like lines or planes to segment (classify) the instance space. These are called Linear models. ● Alternatively, we can use the geometric notion of distance to represent similarity. In this case, if two points are close together, they have similar values for features and thus can be classed as similar. We call such models as Distance-based models. Linear Models Linear models are relatively simple. In this case, the function is represented as a linear combination of its inputs. Thus, if x 1 and x 2 are two scalars or vectors of the same dimension and a and b are arbitrary scalars, then ax 1 + bx 2 represents a linear combination of x 1 and x 2 . In the simplest case where f(x) represents a straight line, we have an equation of the form f(x) = mx + c where c represents the intercept and m represents the slope. Linear models are parametric, which means that they have a fixed form with a small number of numeric parameters that need to be learned from data. For example, in f (x) = mx + c, m and c are the parameters that we are trying to learn from the data. This technique is different from tree or rule models, where the structure of the model	In geometric models, there are two ways we could impose similarity. ? We could use geometric concepts like lines or planes to segment (classify) the instance space. These are called Linear models. ? Alternatively, we can use the geometric notion of distance to represent similarity. In this case, if two points are close together, they have similar values for features and thus can be classed as similar. We call such models as Distance-based models. Linear models Linear models are relatively simple. In this case, the function is represented as a linear combination of its inputs. Thus, if x 1 and x 2 are two scalars or vectors of the same dimension and a and b are arbitrary scalars, then ax 1 + bx 2 represents a linear combination of x 1 and x 2 . In the simplest case where f(x) represents a straight line, we have an equation of the form f (x) = mx + c where c represents the intercept and m represents the slope. Linear models are parametric, which means that they have a fixed form with a small number of numeric parameters that need to be learned from data. For example, in f (x) = mx + c, m and c are the parameters that we are trying to learn from the data. This technique is different from tree or rule models, where the structure of the model (

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

39/189	SUBMITTED TEXT 24 WORDS	93% MATCHING TEXT 24 WORDS
	fixed in advance. Linear models are stable, i.e., small variations in the training data have only a limited impact on the learning model.	fixed in advance. Linear models are stable, i.e., small variations in the training data have only a limited impact on the learned model.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

40/189	SUBMITTED TEXT 132 WORDS	95% MATCHING TEXT 132 WORDS
	As a result of having few parameters, Linear models have low variance and high bias. This implies that Linear models are less likely to overfit the training data than some other models. However, they are more likely to underfit. For example, if we want to learn the boundaries between countries based on labelled data, then linear models are not likely to give a good approximation. Distance-based Models Like Linear models, distance-based models are based on the geometry of data. As the name implies, distance-based models work on the concept of distance. In the context of Machine learning, the concept of distance is not based on merely the physical distance between two points. Instead, we could think of the distance between two points considering the mode of transport between two points.	As a result of having relatively few parameters, Linear models have low variance and high bias. This implies that Linear models are less likely to overfit the training data than some other models. However, they are more likely to underfit. For example, if we want to learn the boundaries between countries based on labelled data, then linear models are not likely to give a good approximation. Distance-based models Distance-based models are the second class of models. Like Linear models, distance- based models are based on the geometry of data. As the name implies, distance-based models work on the concept of distance. In the context of Machine learning, the concept of distance is not based on merely the physical distance between two points. Instead, we could think of the distance between two points considering the mode of transport between two points.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

41/189	SUBMITTED TEXT 45 WORDS	100% MATCHING TEXT 45 WORDS
	in chess, the concept of distance depends on the piece used – for example, a Bishop can move diagonally. Thus, depending on the entity and the mode of travel, the concept of distance can be experienced differently. The distance metrics commonly used are Euclidean	in chess, the concept of distance depends on the piece used – for example, a Bishop can move diagonally. Thus, depending on the entity and the mode of travel, the concept of distance can be experienced differently. The distance metrics commonly used are Euclidean,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

42/189	SUBMITTED TEXT 14 WORDS	100% MATCHING TEXT 14 WORDS
	Probabilistic Models The third family of machine learning algorithms is the probabilistic models.	Probabilistic models The third family of machine learning algorithms is the probabilistic models.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

43/189	SUBMITTED TEXT 229 WORDS	97% MATCHING TEXT 229 WORDS
	The probabilistic models use the idea of probability to classify new entities. Probabilistic models see features and target variables as random variables. The process of modelling represents and manipulates the level of uncertainty with respect to these variables. There are two types of probabilistic models: Predictive and Generative. Predictive probability models use the idea of a conditional probability distribution P (Y | X) from which Y can be predicted from X. Generative models estimate the joint distribution P (Y, X). Once we know the joint distribution for the generative models, we can derive any conditional or marginal distribution involving the same variables. Thus, the generative model is capable of creating new data points and their labels, knowing the joint probability distribution. The joint distribution looks for a relationship between two variables. Once this relationship is inferred, it is possible to infer new data points. Naïve Bayes is an example of a probabilistic classifier. This can be accomplished using the Bayes rule: The Naïve Bayes algorithm is based on the idea of Conditional Probability. Conditional probability is based on finding the probability that something will happen, given that something else has already happened. The task of the algorithm then is to look at the evidence and to determine the likelihood of a specific class and assign a label	the probabilistic models use the idea of probability to classify new entities. Probabilistic models see features and target variables as random variables. The process of modelling represents and manipulates the level of uncertainty with respect to these variables. There are two types of probabilistic models: Predictive and Generative. Predictive probability models use the idea of a conditional probability distribution P (Y |X) from which Y can be predicted from X. Generative models estimate the joint distribution P (Y, X). Once we know the joint distribution for the generative models, we can derive any conditional or marginal distribution involving the same variables. Thus, the generative model is capable of creating new data points and their labels, knowing the joint probability distribution. The joint distribution looks for a relationship between two variables. Once this relationship is inferred, it is possible to infer new data points. Naïve Bayes is an example of a probabilistic classifier. We can do this using the Bayes rule defined as 7 The Naïve Bayes algorithm is based on the idea of Conditional Probability. Conditional probability is based on finding the probability that something will happen, given that something else has already happened. The task of the algorithm then is to look at the evidence and to determine the likelihood of a specific class and assign a label

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

44/189	SUBMITTED TEXT 19 WORDS	52% MATCHING TEXT 19 WORDS
	Models Geometric Models Probabilistic Models Decision Trees, Random Forest Trees, etc. K-nearest neighbors, Linear regression, Support Vector Machine (	models: Geometric models Probabilistic models Logical models E.g. K-nearest neighbors, linear regression, support vector machine,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

45/189	SUBMITTED TEXT 16 WORDS	100% MATCHING TEXT 16 WORDS
	The learning process starts with task T, performance measure P and training experience E and	The learning process starts with task T, performance measure P and training experience E and

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

46/189	SUBMITTED TEXT 100 WORDS	91% MATCHING TEXT 100 WORDS
	objective is to find an unknown target function. The target function is an exact knowledge to be learned from the training experience and its unknown. For example, in a case of credit approval, the learning system will have customer application records as experience and task would be to classify whether the given customer application is eligible for a loan. So in this case, the training examples can be represented as (x 1 ,y 1) (x 2 ,y 2).. (x n ,y n), where X represents customer application details and y represents the status of credit approval.	objective are to find an unknown target function. The target function is an exact knowledge to be learned from the training experience and its unknown. For example, in a case of credit approval, the learning system will have customer application records as experience and task would be to classify whether the given customer application is eligible for a loan. So in this case, the training examples can be represented as 8 (x1,y1)(x2,y2)..(xn,yn) where X represents customer application details and y represents the status of credit approval.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

47/189	SUBMITTED TEXT 37 WORDS	100% MATCHING TEXT 37 WORDS
	So the target function to be learned in the credit approval learning system is a mapping function f:X→y. This function represents the exact knowledge defining the relationship between input variable X and output variable y.	So the target function to be learned in the credit approval learning system is a mapping function f:X →y. This function represents the exact knowledge defining the relationship between input variable X and output variable y.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

48/189	SUBMITTED TEXT 31 WORDS	94% MATCHING TEXT 31 WORDS
	Task (T): To play chess. ● Performance measure (P): Total percent of the game won in the tournament. ● Training experience (E): A set of games played against itself.	Task T: To play checkers 2. Performance measure P: Total percent of the game won in the tournament. 3. Training experience E: A set of games played against itself 1.4.1

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

49/189	SUBMITTED TEXT 13 WORDS	87% MATCHING TEXT 13 WORDS
	needs only to learn how to choose the best move from among	needs only to learn how to choose the best move among

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

50/189	SUBMITTED TEXT 13 WORDS	87% MATCHING TEXT 13 WORDS
	needs only to learn how to choose the best move from among	needs only to learn how to choose the best move among

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

51/189 SUBMITTED TEXT 34 WORDS	95% MATCHING TEXT 34 WORDS
higher scores to better board states. If the system can successfully learn such a target function V, then it can easily use it to select the best move from any current board position.	higher scores to better board states. If the system can successfully learn such a target function V, then it can easily use it to select the best move from any board position. 10

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

52/189 SUBMITTED TEXT 104 WORDS	99% MATCHING TEXT 104 WORDS
Let us therefore define the target value V (b) for an arbitrary board state b in B, as follows: 1. if b is a final board state that is won, then V (b) = 100 2. if b is a final board state that is lost, then V (b) = -100 3. if b is a final board state that is drawn, then V (b) = 0 4. if b is not a final state in the game, then V(b) = V(b'), where b' is the best final board state that can be achieved starting from b and playing optimally until the	Let us therefore define the target value V(b) for an arbitrary board state b in B, as follows: 1. if b is a final board state that is won, then V(b) = 100 2. if b is a final board state that is lost, then V(b) = -100 3. if b is a final board state that is drawn, then V(b) = 0 4. if b is a not a final state in the game, then V (b) = V (b'), where b' is the best final board state that can be achieved starting from b and playing optimally until the

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

53/189 SUBMITTED TEXT 98 WORDS	92% MATCHING TEXT 98 WORDS
The case (4) is a recursive definition and to determine the value of V(b) for a particular board state, it performs the search ahead for the optimal line of play, all the way to the end of the game. So this definition is not efficiently computable by our chess playing program, we say that it is a non-operational definition. The goal of learning in this case is to discover an operational description of V; that is, a description that can be used by the chess playing program to evaluate states and select moves within realistic time bounds.	The (4) is a recursive definition and to determine the value of V(b) for a particular board state, it performs the search ahead for the optimal line of play, all the way to the end of the game. So this definition is not efficiently computable by our checkers playing program, we say that it is a nonoperational definition. The goal of learning, in this case, is to discover an operational description of V ; that is, a description that can be used by the checkers-playing program to evaluate states and select moves within realistic time bounds.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

54/189 SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
It may be very difficult in general to learn such an operational form of V perfectly.	It may be very difficult in general to learn such an operational form of V perfectly.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

55/189 SUBMITTED TEXT 13 WORDS	100% MATCHING TEXT 13 WORDS
expect learning algorithms to acquire only some approximation to the target function,	expect learning algorithms to acquire only some approximation to the target function ^

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

56/189 SUBMITTED TEXT 14 WORDS	88% MATCHING TEXT 14 WORDS
choose a representation that the learning algorithm will use to describe the function	choose a representation that the learning program will use to describe the function ^

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

57/189 SUBMITTED TEXT 18 WORDS	90% MATCHING TEXT 18 WORDS
The function NextMove will be calculated as a linear combination of the following board features: ● x1:	the function ^V will be calculated as a linear combination of the following board features: ? x1(

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

58/189 SUBMITTED TEXT 19 WORDS	66% MATCHING TEXT 19 WORDS
number of black pieces threatened by white (i.e., which can be captured on white's next turn) ● x6:	number of red pieces threatened by black (i.e., which can be taken on black's next turn) ? x6(

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

59/189 SUBMITTED TEXT 16 WORDS	84% MATCHING TEXT 16 WORDS
the learning algorithm. 3.3.4 Choosing an approximation algorithm for the Target Function Training data	the learning program. 1.4.4 Choosing an approximation algorithm for the Target Function Generating training data —

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

60/189 SUBMITTED TEXT 25 WORDS	81% MATCHING TEXT 25 WORDS
a specific board state b and the training value V train (b) for b. In other words, each training example is an ordered pair	a specific board state b and the training value V_train (b) for b. Each training example is an ordered pair &gt;

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

61/189 SUBMITTED TEXT 24 WORDS	67% MATCHING TEXT 24 WORDS
we require a set of training examples, each describing a specific board state b and the training value (Correct Move) y for b.	we need a set of training data, each describing a specific board state b and the training value V_train (b) for b.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

62/189 SUBMITTED TEXT 31 WORDS	94% MATCHING TEXT 31 WORDS
The final design of our chess learning system can be naturally described by four distinct program modules that represent the central components in many learning systems. 1. The performance System:	The final design of our checkers learning system can be naturally described by four distinct program modules that represent the central components in many learning systems. 1. The performance System —

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

63/189	SUBMITTED TEXT 22 WORDS	88% MATCHING TEXT 22 WORDS
	the current hypothesis (currently learned function) and outputs a new problem (i.e., initial board state) for the Performance System to explore.	the current hypothesis (currently learned function) as input and outputs a new problem (an initial board state) for the performance system to explore.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

64/189	SUBMITTED TEXT 27 WORDS	94% MATCHING TEXT 27 WORDS
	using a logical expression (Logical Models), using the Geometry of the instance space (Geometric models), and using Probability to classify the instance space (Probabilistic models). •	Using a Logical expression. (Logical models) ? Using the Geometry of the instance space. (Geometric models) ? Using Probability to classify the instance space. (Probabilistic models) ?

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

65/189	SUBMITTED TEXT 13 WORDS	100% MATCHING TEXT 13 WORDS
	expect learning algorithms to acquire only some approximation to the target function,	expect learning algorithms to acquire only some approximation to the target function ^

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

66/189	SUBMITTED TEXT 22 WORDS	95% MATCHING TEXT 22 WORDS
	supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs.	Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

67/189	SUBMITTED TEXT 49 WORDS	93% MATCHING TEXT 49 WORDS
	where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final Supervised Learning outcomes and the decision nodes are where the data is split.	where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

68/189	SUBMITTED TEXT 14 WORDS	100% MATCHING TEXT 14 WORDS
	The tree can be explained by two entities, namely decision nodes and leaves.	The tree can be explained by two entities, namely decision nodes and leaves.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

69/189	SUBMITTED TEXT 143 WORDS	96% MATCHING TEXT 143 WORDS
	regression. 6.2 Classification Trees A classification tree is an algorithm where the target variable is fixed or categorical. The algorithm is then used to identify the "class" within which a target variable would most likely fall. An example of a classification-type problem would be determining who will or will not subscribe to a digital platform; or who will or will not graduate from high school. These are examples of simple binary classifications where the categorical dependent variable can assume only one of two, mutually exclusive values. In other cases, you might have to predict among a number of different variables. For instance, you may have to predict which type of smartphone a consumer may decide to purchase. In such cases, there are multiple values for the categorical dependent variable. Figure 6.1 represents how a classic classification tree looks like.	Regression Trees 2.1.2.1. Classification Trees A classification tree is an algorithm where the target variable is fixed or categorical. The algorithm is then used to identify the "class" within which a target variable would most likely fall. An example of a classification-type problem would be determining who will or will not subscribe to a digital platform; or who will or will not graduate from high school. These are examples of simple binary classifications where the categorical dependent variable can assume only one of two, mutually exclusive values. In other cases, you might have to predict among a number of different variables. For instance, you may have to predict which type of smartphone a consumer may decide to purchase. In such cases, there are multiple values for the categorical dependent variable. Here's what a classic classification tree looks like 29 2.1.2.2.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

70/189	SUBMITTED TEXT 80 WORDS	100% MATCHING TEXT 80 WORDS
	Regression Trees A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value. As an example of a regression type problem, you may want to predict the selling prices of a residential house, which is a continuous dependent variable. This will depend on both continuous factors like square footage as well as categorical factors like the style of home, area in which the property is located and so on.	Regression Trees A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value. As an example of a regression type problem, you may want to predict the selling prices of a residential house, which is a continuous dependent variable. This will depend on both continuous factors like square footage as well as categorical factors like the style of home, area in which the property is located and so on.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

71/189	SUBMITTED TEXT 140 WORDS	99% MATCHING TEXT 140 WORDS
	The purpose of the analysis conducted by any classification or regression tree is to create a set of if-else conditions that allow for the accurate prediction or classification of a case. ● The Results are Simplistic The interpretation of results summarized in classification or regression trees is usually fairly simple. The simplicity of results helps in the following ways. It allows for the rapid classification of new observations. That's because it is much simpler to evaluate just one or two logical conditions than to compute scores using complex nonlinear equations for each group. It can often result in a simpler model which explains why the observations are either classified or predicted in a certain way. For instance, business problems are much easier to explain with if-then statements than with complex nonlinear equations. ● Classification and Regression Trees are Nonparametric	The purpose of the analysis conducted by any classification or regression tree is to create a set of if-else conditions that allow for the accurate prediction or classification of a case. (i) The Results are Simplistic The interpretation of results summarized in classification or regression trees is usually fairly simple. The simplicity of results helps in the following ways. ? It allows for the rapid classification of new observations. That's because it is much simpler to evaluate just one or two logical conditions than to compute scores using complex nonlinear equations for each group. ? It can often result in a simpler model which explains why the observations are either classified or predicted in a certain way. For instance, business problems are much easier to explain with if-then statements than with complex nonlinear equations. (ii) Classification and Regression Trees are Nonparametric &

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

72/189	SUBMITTED TEXT 14 WORDS	100% MATCHING TEXT 14 WORDS
	Nonlinear The results from classification and regression trees can be summarized in	Nonlinear The results from classification and regression trees can be summarized in

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

73/189	SUBMITTED TEXT 330 WORDS	99% MATCHING TEXT 330 WORDS
	simplistic if-then conditions. This negates the need for the following implicit assumptions. ➔ The predictor variables and the dependent variable are linear. ➔ The predictor variables and the dependent variable follow some specific nonlinear link function. ➔ The predictor variables and the dependent variable are monotonic. Since there is no need for such implicit assumptions, classification and regression tree methods are well suited to data mining. This is because there is very little knowledge or assumptions that can be made beforehand about how the different variables are related. As a result, classification and regression trees can actually reveal relationships between these variables that would not have been possible using other techniques. ● Classification and Regression Trees Implicitly Perform Feature Selection Feature selection or variable screening is an important part of analytics. When we use decision trees, the top few nodes on which the tree is split are the most important variables within the set. As a result, feature selection gets performed automatically and we don't need to do it again. Limitations of Classification and Regression Trees Classification and regression tree tutorials, as well as classification and regression tree ppts, exist in abundance. This is a testament to the popularity of these decision trees and how frequently they are used. However, these decision trees are not without their disadvantages. There are many classification and regression trees examples where the use of a decision tree has not led to the optimal result. Here are some of the limitations of classification and regression trees. ● Overfitting Overfitting occurs when the tree takes into account a lot of noise that exists in the data and comes up with an inaccurate result. ● High variance In this case, a small variance in the data can lead to a very high variance in the prediction, thereby affecting the stability of the outcome. ● Low bias A decision tree that is very complex usually has a low bias. This makes it very	simplistic if-then conditions. This negates the need for the following implicit assumptions. ? The predictor variables and the dependent variable are linear. ? The predictor variables and the dependent variable follow some specific nonlinear link function. ? The predictor variables and the dependent variable are monotonic. Since there is no need for such implicit assumptions, classification and regression tree methods are well suited to data mining. This is because there is very little knowledge or assumptions that can be made beforehand about how the different variables are related. As a result, classification and regression trees can actually reveal relationships between these variables that would not have been possible using other techniques. (iii) Classification and Regression Trees Implicitly Perform Feature Selection Feature selection or variable screening is an important part of analytics. When we use decision trees, the top few nodes on which the tree is split are the most important variables within the set. As a result, feature selection gets performed automatically and we don't need to do it again. Limitations of Classification and Regression Trees 31 Classification and regression tree tutorials, as well as classification and regression tree ppts, exist in abundance. This is a testament to the popularity of these decision trees and how frequently they are used. However, these decision trees are not without their disadvantages. There are many classification and regression trees examples where the use of a decision tree has not led to the optimal result. Here are some of the limitations of classification and regression trees. (i) Overfitting Overfitting occurs when the tree takes into account a lot of noise that exists in the data and comes up with an inaccurate result. (ii) High variance In this case, a small variance in the data can lead to a very high variance in the prediction, thereby affecting the stability of the outcome. (iii) Low bias A decision tree that is very complex usually has a low bias. This makes it very

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

74/189	SUBMITTED TEXT 18 WORDS	62% MATCHING TEXT 18 WORDS
	difficult for the model to incorporate any new data. 6.5 CART Algorithm Classification And Regression Trees (CART)	difficult for the model to incorporate any new data. What is a CART in Machine Learning? A Classification and Regression Tree (CART)

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

75/189	SUBMITTED TEXT 180 WORDS	97% MATCHING TEXT 180 WORDS
	The Classification and regression tree (CART) methodology is one of the oldest and most fundamental algorithms. It is used to predict outcomes based on certain predictor variables. They are excellent for data mining tasks because they require very little data pre-processing. Decision tree models are easy to understand and implement which gives them a strong advantage when compared to other analytical models. 6.6 Regression Analysis Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as temperature, age, salary, price, etc. We can understand the concept of regression analysis using the below example: Suppose there is a marketing company A, who does various advertisements every year and gets sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:	The Classification and regression tree (CART) methodology is one of the oldest and most fundamental algorithms. It is used to predict outcomes based on certain predictor variables. They are excellent for data mining tasks because they require very little data pre-processing. Decision tree models are easy to understand and implement which gives them a strong advantage when compared to other analytical models. 2.2. Regression Regression Analysis in Machine Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as temperature, age, salary, price, etc. We can understand the concept of regression analysis using the below example: Example: Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales: 32

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

76/189	SUBMITTED TEXT 103 WORDS	98% MATCHING TEXT 103 WORDS
	Sales $90 $1000 $120 $1300 $150 $1800 $100 $1200 $130 $1380 $200 ?? Now, the company wants to do the advertisement of $200 in the year 2021 and wants to know the prediction about the sales for this year. So to solve such types of prediction problems in machine learning, we need regression analysis. Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between	sales: 32 Now, the company wants to do the advertisement of $200 in the year 2019 and wants to know the prediction about the sales for this year. So to solve such type of prediction problems in machine learning, we need regression analysis. Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

77/189	SUBMITTED TEXT 45 WORDS	93% MATCHING TEXT 45 WORDS
	variables. In Regression, we plot a graph between the variables which best fits the given data points, using this plot, the machine learning model can make predictions about the data. In simple words, regression shows a line or curve that passes through all the	variables. In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data. In simple words, "Regression shows a line or curve that passes through all the

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

78/189	SUBMITTED TEXT 181 WORDS	87% MATCHING TEXT 181 WORDS
	target-predictor graph in such a way that the vertical distance between the data points and the regression line is minimum. The distance between data points and line tells whether a model has captured a strong relationship or not. Some examples of regression can be as prediction of rain using temperature and other factors; determining Market trends; prediction of road accidents due to rash driving. Some basic terminologies related to the Regression Analysis: ● Dependent Variable: The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called target variable. ● Independent Variable: The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variables, also called as a predictor. ● Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided. ● Multicollinearity: If the independent variables are highly correlated with each other than other variables, then such	target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum." The distance between datapoints and line tells whether a model has captured a strong relationship or not. Some examples of regression can be as: Prediction of rain using temperature and other factors o Determining Market trends o Prediction of road accidents due to rash driving. Terminologies Related to the Regression Analysis: o Dependent Variable: The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called target variable. o Independent Variable: The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a predictor. o Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided. o Multicollinearity: If the independent variables are highly correlated with each other than other variables, then such

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

79/189	SUBMITTED TEXT 16 WORDS	100% MATCHING TEXT 16 WORDS
	condition is called Multicollinearity. It should not be present in the dataset, because it creates	condition is called Multicollinearity. It should not be present in the dataset, because it creates

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

80/189	SUBMITTED TEXT 25 WORDS	97% MATCHING TEXT 25 WORDS
	problem while ranking the most affecting variable. ● Underfitting and Overfitting: If our algorithm works well with the training dataset but not well with	problem while ranking the most affecting variable. o Underfitting and Overfitting: If our algorithm works well with the training dataset but not well with

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

81/189	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	problem is called Overfitting. And if our algorithm does not perform well even with	problem is called Overfitting. And if our algorithm does not perform well even with

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

82/189	SUBMITTED TEXT 37 WORDS	87% MATCHING TEXT 37 WORDS
	problem is called underfitting. As mentioned above, Regression analysis helps in the prediction of a continuous variable. There are various scenarios in the real world where we need some future predictions such as weather conditions, sales	problem is called underfitting. 33 Why do we use Regression Analysis? As mentioned above, Regression analysis helps in the prediction of a continuous variable. There are various scenarios in the real world where we need some future predictions such as weather condition, sales

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

83/189	SUBMITTED TEXT 39 WORDS	95% MATCHING TEXT 39 WORDS
	case we need some technology which can make predictions more accurately. So we need Regression analysis which is a statistical method and used in machine learning and data science. Below are some other reasons for using Regression	case we need some technology which can make predictions more accurately. So for such case we need Regression analysis which is a statistical method and used in machine learning and data science. Below are some other reasons for using Regression

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

84/189	SUBMITTED TEXT 141 WORDS	92% MATCHING TEXT 141 WORDS
	analysis: ● Regression estimates the relationship between the target and the independent variable. ● It is used to find the trends in data. ● It helps to predict real/continuous values. ● By performing the regression, we can confidently determine the most important factor, the least important factor, and how each factor is affecting the other factors. 6.7 Types of Regression There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Some important types of regression are listed below: ● Linear Regression ● Logistic Regression ● Polynomial Regression ● Support Vector Regression ● Decision Tree Regression ● Random Forest Regression ● Ridge Regression ● Lasso Regression	analysis: Regression estimates the relationship between the target and the independent variable. o It is used to find the trends in data. o It helps to predict real/continuous values. o By performing the regression, we can confidently determine the most important factor, the least important factor, and how each factor is affecting the other factors. Types of Regression There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below: o Linear Regression o Logistic Regression o Polynomial Regression o Support Vector Regression o Decision Tree Regression o Random Forest Regression o Ridge Regression o Lasso Regression 2.2.1.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

85/189	SUBMITTED TEXT 207 WORDS	96% MATCHING TEXT 207 WORDS
	Regression 6.7.1 Linear Regression ● Linear regression is a statistical regression method which is used for predictive analysis. ● It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables. ● It is used for solving the regression problem in machine learning. ● Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression. ● If there is only one input variable (x), then such linear regression is called simple linear regression. And if there is more than one input variable, then such linear regression is called multiple linear regression. ● The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of the year of experience. Below is the mathematical equation for Linear regression: Y= aX+b Here, Y = dependent variables (target variables) X= Independent variables (predictor variables) a and b are the linear coefficients Some popular applications of linear regression are: Supervised Learning ● Analyzing trends and sales estimates ● Salary forecasting ● Real estate prediction ● Arriving at ETAs in traffic	Regression 2.2.1. Linear Regression: o Linear regression is a statistical regression method which is used for predictive analysis. o is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables. o It is used for solving the regression problem in machine learning. o Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called regression. 34 o If there is only one input variable (x), then such linear regression is called simple linear regression. And if there is more than one input variable, then such linear regression is called multiple linear regression. The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of the year of experience. Below is the mathematical equation for Linear regression: Y= aX+b Here, Y = dependent variables (target variables), X= Independent variables (predictor variables), a and b are the linear coefficients Some popular applications of linear regression are: Analyzing trends and sales estimates o Salary forecasting o Real estate prediction Arriving at ETAs in traffic. 2.2.2.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

86/189	SUBMITTED TEXT 31 WORDS	100% MATCHING TEXT 31 WORDS
	Linear Regression Line A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:	Linear Regression Line: A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

87/189	SUBMITTED TEXT	202 WORDS	97% MATCHING TEXT	202 WORDS

Positive Linear Relationship: If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship. b) Negative Linear Relationship: If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship. When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error. The different values for weights or the coefficient of lines (a0, a1) gives a different line of regression, so we need to calculate the best values for a0 and a1 to find the best fit line, so to calculate this we use cost function. Cost function- ● The different values for weights or coefficient of lines (a0, a1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. ● Cost function optimizes the regression coefficients or weights. It measures how

Positive Linear Relationship: If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship. 37 o Negative Linear Relationship: If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship. Finding the best fit When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error. The different values for weights or the coefficient of lines (a 0 , a 1) gives a different line of regression, so we need to calculate the best values for a 0 and a 1 to find the best fit line, so to calculate this we use cost function. Cost function- o The different values for weights or coefficient of lines (a 0 , a 1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line. o Cost function optimizes the regression coefficients or weights. It measures how

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

88/189	SUBMITTED TEXT	44 WORDS	98% MATCHING TEXT	44 WORDS

a linear regression model is performing. ● We can use the cost function to find the accuracy of the mapping function, which maps the input variable to the output variable. This mapping function is also known as Hypothesis function. 6.7.2

a linear regression model is performing. o We can use the cost function to find the accuracy of the mapping function, which maps the input variable to the output variable. This mapping function is also known as Hypothesis function.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

89/189	SUBMITTED TEXT	44 WORDS	96% MATCHING TEXT	44 WORDS

Logistic Regression ● Logistic regression is another supervised learning algorithm which is used to solve the classification problems. In classification problems, we have dependent variables in a binary or discrete format such as 0 or 1. ● Logistic regression algorithm works with categorical

Logistic Regression: o Logistic regression is another supervised learning algorithm which is used to solve the classification problems. In classification problems, we have dependent variables in a binary or discrete format such as 0 or 1. o Logistic regression algorithm works with the categorical

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

90/189	SUBMITTED TEXT 184 WORDS	95% MATCHING TEXT 184 WORDS
	such as 0 or 1, Yes or No, True or False, Spam or not spam, etc. ● It is a predictive analysis algorithm which works on the concept of probability. ● Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used. ● Logistic regression uses sigmoid function or logistic function which is a complex cost function. This sigmoid function is used to model the data in logistic regression. The function can be represented as: f(x) = Output between the 0 and 1 value x = input to the function e = base of natural logarithm When we provide the input values (data) to the function, it gives the S-curve as follows: ● It uses the concept of threshold levels, values above the threshold level are rounded up to 1, and values below the threshold level are rounded up to 0. Supervised Learning There are three types of logistic regression: ● Binary (0/1, pass/fail) ● Multi (cats, dogs, lions) ● Ordinal (low, medium, high) 6.8	such as 0 or 1, Yes or No, True or False, Spam or not spam, etc. It is a predictive analysis algorithm which works on the concept of probability. o Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used. o Logistic regression uses sigmoid function or logistic function which is a complex cost function. This sigmoid function is used to model the data in logistic regression. The function can be represented as: 35 o f(x)= Output between the 0 and 1 value. o x= input to the function o e= base of natural logarithm. When we provide the input values (data) to the function, it gives the S-curve as follows: o It uses the concept of threshold levels, values above the threshold level are rounded up to 1, and values below the threshold level are rounded up to 0. There are three types of logistic regression: o Binary(0/1, pass/fail) o Multi(cats, dogs, lions) o Ordinal(low, medium, high)

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

91/189	SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
	A classification tree is an algorithm where the target variable is fixed or categorical. ●	A classification tree is an algorithm where the target variable is fixed or categorical.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

92/189	SUBMITTED TEXT 23 WORDS	100% MATCHING TEXT 23 WORDS
	A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value. ●	A regression tree refers to an algorithm where the target variable is and the algorithm is used to predict it's value.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

93/189	SUBMITTED TEXT 33 WORDS	100% MATCHING TEXT 33 WORDS
	The purpose of the analysis conducted by any classification or regression tree is to create a set of if-else conditions that allow for the accurate prediction or classification of a case. ●	The purpose of the analysis conducted by any classification or regression tree is to create a set of if-else conditions that allow for the accurate prediction or classification of a case. (

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

94/189	SUBMITTED TEXT 27 WORDS	100% MATCHING TEXT 27 WORDS
	Overfitting: Overfitting occurs when the tree takes into account a lot of noise that exists in the data and comes up with an inaccurate result. ●	Overfitting Overfitting occurs when the tree takes into account a lot of noise that exists in the data and comes up with an inaccurate result. (

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

95/189	SUBMITTED TEXT 23 WORDS	100% MATCHING TEXT 23 WORDS
	Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. ●	Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

96/189	SUBMITTED TEXT 39 WORDS	100% MATCHING TEXT 39 WORDS
	The probabilistic models use the idea of probability to classify new entities. Probabilistic models see features and target variables as random variables. The process of modelling represents and manipulates the level of uncertainty with respect to these variables.	the probabilistic models use the idea of probability to classify new entities. Probabilistic models see features and target variables as random variables. The process of modelling represents and manipulates the level of uncertainty with respect to these variables.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

97/189	SUBMITTED TEXT 184 WORDS	91% MATCHING TEXT 184 WORDS
	One type of ANN system is based on a unit called a perceptron, illustrated in Figure 8.4. A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise. More precisely, given inputs x 1 through x n , the output o(x 1 , . . . , x n) computed by the perceptron is where each w i is a real-valued constant, or weight, that determines the contribution Supervised Learning of input x i to the perceptron output. Notice the quantity (-w 0) is a threshold that the weighted combination of inputs w 1 x 1 + . . . + w n x n must surpass in order for the perceptron to output a 1. To simplify notation, we imagine an additional constant input x 0 = 1, allowing us to write the above inequality as , or in vector form as .For brevity, we will sometimes write the perceptron function as	One type of ANN system is based on a unit called a perceptron, illustrated in below Figure: A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise. More precisely, given inputs x l through x n the output o(x l , . . . , x n) computed by the perceptron is where each w i is a real-valued constant, or weight, that determines the contribution of input x i to the perceptron output. Notice the quantity) (0 w ? is a threshold that the weighted combination of inputs xw xw n n ? ?.... 1 1 must surpass in order for the perceptron to output a 1. To simplify notation, we imagine an additional constant input 1 0 ? x , allowing us to write the above inequality as 0 0 ? ? ? n i i i xw , or in vector form as oxw ? ? ? . . For brevity, we will sometimes write the perceptron function as

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

98/189	SUBMITTED TEXT 42 WORDS	100% MATCHING TEXT 42 WORDS
	Learning a perceptron involves choosing values for the weights w 0 , . . . , w n . Therefore, the space H of candidate hypotheses considered in perceptron learning is the set of all possible real-valued weight vectors.	Learning a perceptron involves choosing values for the weights w w n ,...., 0 Therefore, the space H of candidate hypotheses considered in perceptron learning is the set of all possible real-valued weight vectors.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

99/189 **SUBMITTED TEXT** 74 WORDS **100% MATCHING TEXT** 74 WORDS

SUBMITTED TEXT	MATCHING TEXT
the inputs are fed to multiple units, and the outputs of these units are then input to a second, final stage. One way is to represent the boolean function in disjunctive normal form (i.e., as the disjunction (OR) of a set of conjunctions (ANDs) of the inputs and their negations). Note that the input to an AND perceptron can be negated simply by changing the sign of the corresponding input weight.	The inputs are fed to multiple units, and the outputs of these units are then input to a second, final stage. One way is to represent the Boolean function in disjunctive normal form (i.e., as the disjunction (OR) of a set of conjunctions (ANDs) of the inputs and their negations). Note that the input to an AND perceptron can be negated simply by changing the sign of the corresponding input weight.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

100/189 **SUBMITTED TEXT** 122 WORDS **97% MATCHING TEXT** 122 WORDS

SUBMITTED TEXT	MATCHING TEXT
The Perceptron Training Rule Although we are interested in learning networks of many interconnected units, let us begin by understanding how to learn the weights for a single perceptron. Here the precise learning problem is to determine a weight vector that causes the perceptron to produce the correct 1 output for each of the given training examples. Several algorithms are known to solve this learning problem. Here we consider the perceptron rule. These two algorithms are guaranteed to converge to somewhat different acceptable hypotheses, under somewhat different conditions. They are important to ANNs because they provide the basis for learning networks of many units. One way to learn an acceptable weight vector is to begin with random weights, then	The Perceptron Training Rule Although we are interested in learning networks of many interconnected units, let us begin by understanding how to learn the weights for a single perceptron. Here the precise learning problem is to determine a weight vector that causes the perceptron to produce the correct 1? output for each of the given training examples. Several algorithms are known to solve this learning problem. Here we consider two: the perceptron rule and the delta rule. These two algorithms are guaranteed to converge to somewhat different acceptable hypotheses, under somewhat different conditions. They are important to ANNs because they provide the basis for learning networks of many units. One way to learn an acceptable weight vector is to begin with random weights, then

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

101/189 **SUBMITTED TEXT** 193 WORDS **98% MATCHING TEXT** 193 WORDS

SUBMITTED TEXT	MATCHING TEXT
iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example. This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly. Weights are modified at each step according to the perceptron training rule, which revises the weight w i associated with input x i according to the rule Here t is the target output for the current training example, o is the output generated by the perceptron, and η is a positive constant called the learning rate. The role of the learning rate is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases. Why should this update rule converge toward successful weight values? To get an intuitive feel, consider some specific cases. Suppose the training example is correctly classified already by the perceptron. In this case, (t - o) is zero, making Δw i zero, so that no weights are updated. Suppose the	iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example. This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly. Weights are modified at each step according to the perceptron training rule, which revises the weight w i associated with input x i according to the rule Here t is the target output for the current training example, o is the output generated by the 47 perceptron, and ? is a positive constant called the learning rate. The role of the learning rate is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases. Why should this update rule converge toward successful weight values? To get an intuitive feel, consider some specific cases. Suppose the training example is correctly classified already by the perceptron. In this case, (t - o) is zero, making w i ? zero, so that no weights are updated. Suppose the

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

102/189 SUBMITTED TEXT 70 WORDS	99% MATCHING TEXT 70 WORDS
perceptron outputs a -1, when the target output is +1. To make the perceptron output a+1 instead of -1 in this case, the weights must be altered to increase the value of .For example, if x i &lt;0, then increasing w i will bring the perceptron closer to correctly classifying this example. Notice the training rule will increase w, in this case, because (t - o),	perceptron outputs a -1, when the target output is +1. To make the perceptron output a+1 instead of -1 in this case, the weights must be altered to increase the value of ? ? xw. For example, if x i &lt;0, then increasing w i will bring the perceptron closer to correctly classifying this example. Notice the training rule will increase w, in this case, because (t - o), ? ,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

103/189 SUBMITTED TEXT 101 WORDS	95% MATCHING TEXT 101 WORDS
t = 1, and o = - 1, then the weight update will be On the other hand, if t = -1 and o = 1, then weights associated with positive x i will be decreased rather than increased. In fact, the above learning procedure can be proven to converge within a finite number of applications of the perceptron training rule to a weight vector that correctly classifies all training examples, provided the training examples are linearly separable and provided a sufficiently small η is used. If the data are not linearly separable, convergence is not assured.	t = 1, and o = - 1, then the weight update will be w i ? = ? (t - o)x i = 0.1(1 - (-1))0.8 = 0.16. On other hand, if t = -1 and o = 1, then weights associated with positive x i will be decreased rather than increased. In fact, the above learning procedure can be proven to converge within a finite number of applications of the perceptron training rule to a weight vector that correctly classifies all training examples, provided the training examples are linearly separable and provided a sufficiently small ? is used. If the data are not linearly separable, convergence is not assured.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

104/189 SUBMITTED TEXT 215 WORDS	93% MATCHING TEXT 215 WORDS
Artificial Neural Network ANN can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neuron outputs and neuron inputs can be viewed as the directed edges with weights. The ANN receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs. Afterward, each of the inputs is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. Supervised Learning If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.	Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs. each of the is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs summarized inside computing unit. If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

105/189	SUBMITTED TEXT 99 WORDS	100% MATCHING TEXT 99 WORDS
	Input Layer: As the name suggests, it accepts inputs in several different formats provided by the programmer. ● Hidden Layer: The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns. ● Output Layer: The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function. It determines	Input Layer: As the name suggests, it accepts inputs in several different formats provided by the programmer. Hidden Layer: The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns. Output Layer: The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function. It determines

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

106/189	SUBMITTED TEXT 40 WORDS	100% MATCHING TEXT 40 WORDS
	weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer.	weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

107/189	SUBMITTED TEXT 76 WORDS	94% MATCHING TEXT 76 WORDS
	Types of ANN There are various types of ANN depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification. ● Feedback ANN: In this type of ANN, the output returns into the network to accomplish the best-evolved results internally.	Types of Artificial Neural Network: There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification. Feedback ANN: In this type of ANN, the output returns into the network to accomplish the best-evolved results internally.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

108/189	SUBMITTED TEXT 100 WORDS	98% MATCHING TEXT 100 WORDS
	The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs. ● Feed-Forward ANN: A feed-forward network is a basic neural network consisting of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.	The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs. Feed-Forward ANN: A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns. 45

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

109/189	SUBMITTED TEXT 189 WORDS	98% MATCHING TEXT 189 WORDS
	Advantages of ANN ● Parallel processing capability: Artificial neural networks have a numerical value that can perform more than one task simultaneously. ● Storing data on the entire network: Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working. ● Capability to work with incomplete knowledge: After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data. ● Having a memory distribution: For ANN to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output. ● Having fault tolerance: Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.	Advantages of Artificial Neural ANN) Parallel processing capability: Artificial neural networks have a numerical value that can perform more than one task simultaneously. Storing data on the entire network: Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working. Capability to work with incomplete knowledge: After ANN training, the information may produce output even with inadequate data. The loss of 43 performance here relies upon the significance of missing data. Having a memory distribution: For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output. Having fault tolerance: Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

110/189	SUBMITTED TEXT 143 WORDS	98% MATCHING TEXT 143 WORDS
	Disadvantages of ANN ● Assurance of proper network structure: There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error. ● Unrecognized behavior of the network: It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network. ● Hardware dependence: Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent. ● Difficulty of showing the issue to the network: ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities. 8.6	Disadvantages of Artificial Neural Network: Assurance of proper network structure: There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error. Unrecognized behavior of the network: It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network. Hardware dependence: Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent. Difficulty of showing the issue to the network: ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

111/189 SUBMITTED TEXT 115 WORDS	98% MATCHING TEXT 115 WORDS
Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function f(·):R m →R o by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X = x 1 ,x 2 ,...,x m and a target y, it can learn a nonlinear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 8.7 shows a one hidden layer MLP with scalar output.	Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function f(·):R m →R o by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X=x 1 ,x 2 ,...,x m and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure shows a one hidden layer MLP with scalar output.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

112/189 SUBMITTED TEXT 102 WORDS	100% MATCHING TEXT 102 WORDS
one hidden layer MLP with scalar output The leftmost layer, known as the input layer, consists of a set of neurons {x i |x 1 , x 2 ,...,x m } representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation w 1 x 1 + w 2 x 2 +...+ w m x m , followed by a non-linear activation function g(·):R→R - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.	one hidden layer MLP with scalar output. The leftmost layer, known as the input layer, consists of a set of neurons {x i |x 1 ,x 2 ,...,x m } representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation w 1 x 1 +w 2 x 2 +...+w m x m , followed by a non-linear activation function g(·):R→R - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

113/189 SUBMITTED TEXT 79 WORDS	87% MATCHING TEXT 79 WORDS
Advantages of Multilayer Perceptron • Capability to learn non-linear models. • Capability to learn models in real-time. Disadvantages of Multilayer Perceptron • MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy. • MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations. • MLP is sensitive to feature scaling.	advantages of Multi-layer Perceptron are: ? Capability to learn non-linear models. ? Capability to learn models in real-time (on-line learning) using partial_fit. The disadvantages of Multi-layer Perceptron (MLP) include: ? MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy. ? MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations. ? MLP is sensitive to feature scaling. 49 2.4.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

114/189 SUBMITTED TEXT 33 WORDS	100% MATCHING TEXT 33 WORDS
A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise. 8.9	A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

115/189	SUBMITTED TEXT 103 WORDS	99% MATCHING TEXT 103 WORDS
	Support Vector Machine or SVM is one of the most popular supervised learning algorithms, which is used for classification as well as regression problems. However, primarily, it is used for classification problems in machine learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n- dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence	Support Vector Machine or SVM is one of the most popular Supervised Learning which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

116/189	SUBMITTED TEXT 29 WORDS	100% MATCHING TEXT 29 WORDS
	algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:	algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

117/189	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	Support Vector Machine or SVM is one of the most popular supervised learning	Support Vector Machine or SVM is one of the most popular Learning

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

118/189	SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
	Supervised Learning algorithms, which is used for classification as well as regression problems. ●	Supervised Learning algorithms, which is used for Classification as well as Regression problems.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

119/189	SUBMITTED TEXT 20 WORDS	97% MATCHING TEXT 20 WORDS
	SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors. ●	SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

120/189	SUBMITTED TEXT 84 WORDS	97% MATCHING TEXT 84 WORDS
	Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier. ● Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier. ●	Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier. o Non-linear SVM: Non-Linear is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

121/189	SUBMITTED TEXT 44 WORDS	89% MATCHING TEXT 44 WORDS
	learning is used to draw inferences from data sets consisting of input data without labeled responses. In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions.	learning algorithm used to draw inferences from consisting of input data without labeled responses. In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

122/189	SUBMITTED TEXT 39 WORDS	100% MATCHING TEXT 39 WORDS
	Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of	Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to find the underlying structure of

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

123/189	SUBMITTED TEXT 20 WORDS	94% MATCHING TEXT 20 WORDS
	dataset, group that data according to similarities, and represent that dataset in a compressed format. The unsupervised learning algorithm	dataset, group that data according to similarities, and represent that dataset in a compressed format Example: the unsupervised learning algorithm

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

124/189	SUBMITTED TEXT 38 WORDS	95% MATCHING TEXT 38 WORDS
	Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occur together in the dataset. Association rule	Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

125/189	SUBMITTED TEXT 34 WORDS	100% MATCHING TEXT 34 WORDS
	makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.	makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

126/189	SUBMITTED TEXT 23 WORDS	100% MATCHING TEXT 23 WORDS
	An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. •	An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

127/189	SUBMITTED TEXT 21 WORDS	100% MATCHING TEXT 21 WORDS
	Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.	Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

128/189	SUBMITTED TEXT 16 WORDS	80% MATCHING TEXT 16 WORDS
	has to figure out what it did that made it get the reward or punishment,	has to find out what it did that made it get the reward/ punishment.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

129/189	SUBMITTED TEXT 15 WORDS	100% MATCHING TEXT 15 WORDS
	getting an agent to act in the world so as to maximize its rewards.	getting an agent to act in the world so as to maximize its rewards.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

130/189	SUBMITTED TEXT 22 WORDS	100% MATCHING TEXT 22 WORDS
	Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards. •	Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

131/189	SUBMITTED TEXT 14 WORDS	100% MATCHING TEXT 14 WORDS
	Genetic algorithms (GAS) provide a learning method motivated by an analogy to	Genetic algorithms (GAS) provide a learning method motivated by an analogy to

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

132/189	SUBMITTED TEXT 359 WORDS	99% MATCHING TEXT 359 WORDS
	biological evolution. Rather than search from general-to-specific hypotheses, or from simple-to-complex, GAS generates successor hypotheses by repeatedly mutating and recombining parts of the best currently known hypotheses. At each step, a collection of hypotheses called the current population is updated by replacing some fraction of the population by offspring of the most fit current hypotheses. The process forms a generate-and-test beam-search of hypotheses, in which variants of the best current hypotheses are most likely to be considered next. The popularity of GAS is motivated by a number of factors including: ● Evolution is known to be a successful, robust method for adaptation within biological systems. ● GAS can search spaces of hypotheses containing complex interacting parts, where the impact of each part on overall hypothesis fitness may be difficult to model. ● Genetic algorithms are easily parallelized and can take advantage of the decreasing costs of powerful computer hardware. The problem addressed by GAS is to search a space of candidate hypotheses to identify the best hypothesis. In GAS the best hypothesis is defined as the one that optimizes a predefined numerical measure for the problem at hand, called the hypothesis fitness. For example, if the learning task is the problem of approximating an unknown function given training examples of its input and output, then fitness could be defined as the accuracy of the hypothesis over this training data. If the task is to learn a strategy for playing chess, fitness could be defined as the number of games won by the individual when playing against other individuals in the current population. Although different implementations of genetic algorithms vary in their details, they typically share the following structure: The algorithm operates by iteratively updating a pool of hypotheses, called the population. On each iteration, all members of the population are evaluated according to the fitness function. A new population is then generated by probabilistically selecting the fit individuals from the current population. Some of these selected individuals are carried forward into the next generation population intact. Others are used as the basis for creating new offspring individuals by applying genetic operations such as crossover and mutation.	biological evolution. Rather than search from general-to-specific hypotheses, or from simple-to-complex, GAS generate successor hypotheses by repeatedly mutating and recombining parts of the best currently known hypotheses. At each step, a collection of hypotheses called the current population is updated by replacing some fraction of the population by offspring of the most fit current hypotheses. The process forms a generate-and-test beam-search of hypotheses, in which variants of the best current hypotheses are most likely to be considered next. The popularity of GAS is motivated by a number of factors including: ? Evolution is known to be a successful, robust method for adaptation within biological systems. ? GAS can search spaces of hypotheses containing complex interacting parts, where the impact of each part on overall hypothesis fitness may be difficult to model. ? Genetic algorithms are easily parallelized and can take advantage of the decreasing costs of powerful computer hardware. 9.2 Genetic The problem addressed by GAS is to search a space of candidate hypotheses to identify the best hypothesis. In GAS the "best hypothesis" is defined as the one that optimizes a predefined numerical measure for the problem at hand, called b the hypothesis fitness. For example, if the learning task is the problem of approximating an unknown function given training examples of its input and output, then fitness could be defined as the accuracy of the hypothesis over this training data. If the task is to learn a strategy for playing chess, fitness could be defined as the number of games won by the individual when playing against other individuals in the current population. Although different implementations of genetic algorithms vary in their details, they typically share the following structure: The algorithm operates by iteratively updating a pool of hypotheses, called the population. On each iteration, all members of the population are evaluated according to the fitness function. A new population is then generated by probabilistically selecting the fit individuals from the current population. Some of these selected individuals are carried forward into the next generation population intact. Others are used as the basis for creating new offspring individuals by applying genetic operations such as crossover and mutation. 101

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

133/189	SUBMITTED TEXT 60 WORDS	100% MATCHING TEXT 60 WORDS
	The inputs to this algorithm include the fitness function for ranking candidate hypotheses, a threshold defining an acceptable level of fitness for terminating the algorithm, the size of the population to be maintained, and parameters that determine how successor populations are to be generated: the fraction of the population to be replaced at each generation and the mutation rate.	The inputs to this algorithm include the fitness function for ranking candidate hypotheses, a threshold defining an acceptable level of fitness for terminating the algorithm, the size of the population to be maintained, and parameters that determine how successor populations are to be generated: the fraction of the population to be replaced at each generation and the mutation rate.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

134/189	SUBMITTED TEXT 156 WORDS	100% MATCHING TEXT 156 WORDS
	in this algorithm each iteration through the main loop produces a new generation of hypotheses based on the current population. First, a certain number of hypotheses from the current population are selected for inclusion in the next generation. These are selected probabilistically, where the probability of selecting hypothesis h i is given by Thus, the probability that a hypothesis will be selected is proportional to its own fitness and is inversely proportional to the fitness of the other competing hypotheses in the current population. Once these members of the current generation have been selected for inclusion in the next generation population, additional members are generated using a crossover operation. Crossover, defined in detail in the next section, takes two parent hypotheses from the current generation and creates two offspring hypotheses by recombining portions of both parents. The parent hypotheses are chosen probabilistically from the current population, again using the probability function given by	in this algorithm each iteration through the main loop produces a new generation of hypotheses based on the current population. First, a certain number of hypotheses from the current population are selected for inclusion in the next generation. These are selected probabilistically, where the probability of selecting hypothesis h i is given by Thus, the probability that a hypothesis will be selected is proportional to its own fitness and is inversely proportional to the fitness of the other competing hypotheses in the current population. Once these members of the current generation have been selected for inclusion in the next generation population, additional members are generated using a crossover operation. Crossover, defined in detail in the next section, takes two parent hypotheses from the current generation and creates two offspring hypotheses by recombining portions of both parents. The parent hypotheses are chosen probabilistically from the current population, again using the probability function given by

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

135/189	SUBMITTED TEXT 88 WORDS	100% MATCHING TEXT 88 WORDS
	equation. After new members have been created by this crossover operation, the new generation population now contains the desired number of members. At this point, a certain fraction m of these members are chosen at random, and random mutations all performed to alter these members. This GA algorithm thus performs a randomized, parallel beam search for hypotheses that perform well according to the fitness function. In the following subsections, we describe in more detail the representation of hypotheses and genetic operators used in this algorithm.	Equation (9.1). After new members have been created by this crossover operation, the new generation population now contains the 102 desired number of members. At this point, a certain fraction m of these members are chosen at random, and random mutations all performed to alter these members. This GA algorithm thus performs a randomized, parallel beam search for hypotheses that perform well according to the fitness function. In the following subsections, we describe in more detail the representation of hypotheses and genetic operators used in this algorithm.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

136/189	SUBMITTED TEXT 133 WORDS	100% MATCHING TEXT 133 WORDS
	Algorithm Representing Hypotheses Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover. The hypotheses represented by these bit strings can be quite complex. For example, sets of if-then rules can easily be represented in this way, by choosing an encoding of rules that allocates specific substrings for each rule precondition and postcondition. To see how if-then rules can be encoded by bit strings, first consider how we might use a bit string to describe a constraint on the value of a single attribute. To pick an example, consider the attribute Outlook, which can take on any of the three values Sunny, Overcast, or Rain. One obvious way to represent a constraint on Outlook is	algorithm. Representing Hypotheses Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover. The hypotheses represented by these bit strings can be quite complex. For example, sets of if-then rules can easily be represented in this way, by choosing an encoding of rules that allocates specific substrings for each rule precondition and postcondition. To see how if-then rules can be encoded by bit strings, .first consider how we might use a bit string to describe a constraint on the value of a single attribute. To pick an example, consider the attribute Outlook, which can take on any of the three values Sunny, Overcast, or Rain. One obvious way to represent a constraint on Outlook is

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

137/189 **SUBMITTED TEXT** 373 WORDS **100%** **MATCHING TEXT** 373 WORDS

to use a bit string of length three, in which each bit position corresponds to one of its three possible values. Placing a 1 in some position indicates that the attribute is allowed to take on the corresponding value. For example, the string 010 represents the constraint that Outlook must take on the second of these values, or Outlook = Overcast. Similarly, the string 011 represents the more general constraint that allows two possible values, or (Outlook = Overcast v Rain). Note 11 1 represents the most general possible constraint, indicating that we don't care which of its possible values the attribute takes on. Given this method for representing constraints on a single attribute, conjunctions of constraints on multiple attributes can easily be represented by concatenating the corresponding bit strings. For example, consider a second attribute, Wind, that can take on the value Strong or Weak. A rule precondition such as (Outlook = Overcast ^Rain) A (Wind = Strong) can then be represented by the following bit string of length five: Outlook Wind 01 1 10 Rule postconditions (such as PlayTennis = yes) can be represented in a similar fashion. Thus, an entire rule can be described by concatenating the bit strings describing the rule preconditions, together with the bit string describing the rule postcondition. For example, the rule IF Wind = Strong THEN PlayTennis = yes would be represented by the string Outlook Wind PlayTennis 111 10 10 where the first three bits describe the don't care constraint on Outlook, the next two bits describe the constraint on Wind, and the final two bits describe the rule postcondition (here we assume PlayTennis can take on the values Yes or No). Note the bit string representing the rule contains a substring for each attribute in the hypothesis space, even if that attribute is not constrained by the rule preconditions. This yields a fixed length bit-string representation for rules, in which substrings at specific locations describe constraints on specific attributes. Given this representation for single rules, we can represent sets of rules by similarly concatenating the bit string representations of the individual rules.

to use a bit string of length three, in which each bit position corresponds to one of its three possible values. Placing a 1 in some position indicates that the attribute is allowed to take on the corresponding value. For example, the string 010 represents the constraint that Outlook must take on the second of these values, or Outlook = Overcast. Similarly, the string 011 represents the more general constraint that allows two possible values, or (Outlook = Overcast v Rain). Note 11 1 represents the most general possible constraint, indicating that we don't care which of its possible values the attribute takes on. Given this method for representing constraints on a single attribute, conjunctions of constraints on multiple attributes can easily be represented by concatenating the corresponding bit strings. For example, consider a second attribute, Wind, that can take on the value Strong or Weak. A rule precondition such as (Outlook = Overcast ^Rain) A (Wind = Strong) can then be represented by the following bit string of length five: Outlook Wind 01 1 10 Rule postconditions (such as PlayTennis = yes) can be represented in a similar fashion. Thus, an entire rule can be described by concatenating the bit strings describing the rule preconditions, together with the bit string describing the rule postcondition. For example, the rule IF Wind = Strong THEN PlayTennis = yes would be represented by the string Outlook Wind PlayTennis 111 10 10 where the first three bits describe the "don't care" constraint on Outlook, the next two bits describe the constraint on Wind, and the final two bits describe the rule postcondition (here we assume PlayTennis 103 can take on the values Yes or No). Note the bit string representing the rule contains a substring for each attribute in the hypothesis space, even if that attribute is not constrained by the rule preconditions. This yields a fixed length bit-string representation for rules, in which substrings at specific locations describe constraints on specific attributes. Given this representation for single rules, we can represent sets of rules by similarly concatenating the bit string representations of the individual rules.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

138/189	SUBMITTED TEXT 347 WORDS	100% MATCHING TEXT 347 WORDS
	In designing a bit string encoding for some hypothesis space, it is useful to arrange for every syntactically legal bit string to represent a well-defined hypothesis. To illustrate, note in the rule encoding in the above paragraph the bit string 11 1 10 11 represents a rule whose postcondition does not constrain the target attribute PlayTennis. If we wish to avoid considering this hypothesis, we may employ a different encoding (e.g., allocate just one bit to the PlayTennis postcondition to indicate whether the value is Yes or No), alter the genetic operators so that they explicitly avoid constructing such bit strings, or simply assign a very low fitness to such bit strings. In some GAS, hypotheses are represented by symbolic descriptions rather than bit strings. 15.3 Genetic Operators The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population. These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are crossover and mutation. • The crossover operator produces two new offspring from two parent strings, by copying selected bits from each parent. The bit at position i in each offspring is copied from the bit at position i in one of the two parents. The choice of which parent contributes the bit for position i is determined by an additional string called the crossover mask. To illustrate, consider the single-point crossover operator at the top of Table Consider the topmost of the two offspring in this case. This offspring takes its first five bits from the first parent and its remaining six bits from the second parent, because the crossover mask 11 11 1000000 specifies these choices for each of the bit positions. The second offspring uses the same crossover mask, but switches the roles of the two parents. Therefore, it contains the bits that were not used by the first offspring. In single-point crossover, the crossover mask is always constructed so that it begins with a string containing n contiguous 1	In designing a bit string encoding for some hypothesis space, it is useful to arrange for every syntactically legal bit string to represent a well-defined hypothesis. To illustrate, note in the rule encoding in the above paragraph the bit string 11 1 10 11 represents a rule whose postcondition does not constrain the target attribute PlayTennis. If we wish to avoid considering this hypothesis, we may employ a different encoding (e.g., allocate just one bit to the PlayTennis postcondition to indicate whether the value is Yes or No), alter the genetic operators so that they explicitly avoid constructing such bit strings, or simply assign a very low fitness to such bit strings. In some GAS, hypotheses are represented by symbolic descriptions rather than bit strings. Genetic Operators The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population. These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are crossover and mutation. The crossover operator produces two new offspring from two parent strings, by copying selected bits from each parent. The bit at position i in each offspring is copied from the bit at position i in one of the two parents. The choice of which parent contributes the bit for position i is determined by an additional string called the crossover mask. To illustrate, consider the single-point crossover operator at the top of Table Consider the topmost of the two offspring in this case. This offspring takes its first five bits from the first parent and its remaining six bits from the second parent, because the crossover mask 11 11 1000000 specifies these choices for each of the bit positions. The second offspring uses the same crossover mask, but switches the roles of the two parents. Therefore, it contains the bits that were not used by the first offspring. In single-point crossover, the crossover mask is always constructed so that it begins with a string containing n contiguous

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

139/189	SUBMITTED TEXT 60 WORDS	100% MATCHING TEXT 60 WORDS
	followed by the necessary number of 0s to complete the string. This results in offspring in which the first n bits are contributed by one parent and the remaining bits by the second parent. Each time the single-point crossover operator is applied the crossover point n is chosen at random, and the crossover mask is then created and	followed by the necessary number of 0s to complete the string. This results in offspring in which the first n bits are contributed by one parent and the remaining bits by the second parent. Each time the single-point crossover operator is applied the crossover point n is chosen at random, and the crossover mask is then created and

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

140/189	SUBMITTED TEXT 83 WORDS	96% MATCHING TEXT 83 WORDS
	applied. In two-point crossover, offspring are created by substituting intermediate segments of one parent into the middle of the second parent string. Put another way, the crossover mask is a string beginning with n 0 zeros, followed by a contiguous string of n 1 ones, followed by the necessary number of zeros to complete the string. Each time the two-point crossover operator is applied, a mask is generated by randomly choosing the integers n 0 and n 1 .	applied. 104 In two-point crossover, offspring are created by substituting intermediate segments of one parent into the middle of the second parent string. Put another way, the crossover mask is a string beginning with no zeros, followed by a contiguous string of n l ones, followed by the necessary number of zeros to complete the string. Each time the two-point crossover operator is applied, a mask is generated by randomly choosing the integers n o and n

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

141/189	SUBMITTED TEXT 53 WORDS	100% MATCHING TEXT 53 WORDS
	The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population. If the task is to learn classification rules, then the fitness function typically has a component that scores the classification accuracy of the rule over a set of provided	The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population. If the task is to learn classification rules, then the fitness function typically has a component that scores the classification accuracy of the rule over a set of provided

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

142/189	SUBMITTED TEXT 110 WORDS	100% MATCHING TEXT 110 WORDS
	training examples. Often other criteria may be included as well, such as the complexity or generality of the rule. More generally, when the bit-string hypothesis is interpreted as a complex procedure (e.g., when the bit string represents a collection of if-then rules that will be chained together to control a robotic device), the fitness function may measure the overall performance of the resulting procedure rather than performance of individual rules. In our prototypical GA shown in above Table 15.1 , the probability that a hypothesis will be selected is given by the ratio of its fitness to the fitness of other members of the current population as seen in	training examples. Often other criteria may be included as well, such as the complexity or generality of the rule. More generally, when the bit-string hypothesis is interpreted as a complex procedure (e.g., when the bit string represents a collection of if-then rules that will be chained together to control a robotic device), the fitness function may measure the overall performance of the resulting procedure rather than performance of individual rules. In our prototypical GA shown in above Table , the probability that a hypothesis will be selected is given by the ratio of its fitness to the fitness of other members of the current population as seen in

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

143/189	SUBMITTED TEXT 120 WORDS	99% MATCHING TEXT 120 WORDS
	equation. This method is sometimes called fitness proportionate selection, or roulette wheel selection. Other methods for using fitness to select hypotheses have also been proposed. For example, in tournament selection, two hypotheses are first chosen at random from the current population. With some predefined probability p the more fit of these two is then selected, and with probability (1 - p) the less fit hypothesis is selected. Tournament selection often yields a more diverse population than fitness proportionate selection. In another method called rank selection, the hypotheses in the current population are first sorted by fitness. The probability that a hypothesis will be selected is then proportional to its rank in this sorted list, rather than its fitness.	Equation above . This method is sometimes called fitness proportionate selection, or roulette wheel selection. Other methods for using fitness to select hypotheses have also been proposed. For example, in tournament selection, two hypotheses are first chosen at random from the current population. With some predefined probability p the more fit of these two is then selected, and with probability (1 - p) the less fit hypothesis is selected. Tournament selection often yields a more diverse population than fitness proportionate selection. In another method called rank selection, the hypotheses in the current population are first sorted by fitness. The probability that a hypothesis will be selected is then proportional to its rank in this sorted list, rather than its fitness. 105 5.3.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

144/189	SUBMITTED TEXT 45 WORDS	100% MATCHING TEXT 45 WORDS
	Fitness function: The fitness of each hypothesized rule set is based on its classification accuracy over the training data. In particular, the function used to measure fitness is where correct (h) is the percent of all training examples correctly classified by hypothesis h.	Fitness function. The fitness of each hypothesized rule set is based on its classification accuracy over the training data. In particular, the function used to measure fitness is where correct (h) is the percent of all training examples correctly classified by hypothesis h.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

145/189	SUBMITTED TEXT 70 WORDS	100% MATCHING TEXT 70 WORDS
	crowding. Crowding is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of this individual and very similar individuals take over a large fraction of the population. The negative impact of crowding is that it reduces the diversity of the population, thereby slowing further progress by the GA. Several strategies have been explored for reducing crowding.	crowding. Crowding is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of this individual and very similar individuals take over a large fraction of the population. The negative impact of crowding is that it reduces the diversity of the population, thereby slowing further progress by the GA. Several strategies have been explored for reducing crowding.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

146/189	SUBMITTED TEXT 14 WORDS	100% MATCHING TEXT 14 WORDS
	Many of these techniques are inspired by the analogy to biological evolution. •	Many of these techniques are inspired by the analogy to biological evolution.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

147/189	SUBMITTED TEXT	119 WORDS	100% MATCHING TEXT	119 WORDS
	One approach is to alter the selection function, using criteria such as tournament selection or rank selection in place of fitness proportionate roulette wheel selection. ● A related strategy is fitness sharing, in which the measured fitness of an individual is reduced by the presence of other, similar individuals in the population. ● A third approach is to restrict the kinds of individuals allowed to recombine to form offspring. For example, by allowing only the most similar individuals to recombine, we can encourage the formation of clusters of similar individuals, or multiple "subspecies" within the population. ● A related approach is to spatially distribute individuals and allow only nearby individuals to recombine. 15.5.2		One approach is to alter the selection function, using criteria such as tournament selection or rank selection in place of fitness proportionate roulette wheel selection. A related strategy is "fitness sharing," in which the measured fitness of an individual is reduced by the presence of other, similar individuals in the population. A third approach is to restrict the kinds of individuals allowed to recombine to form offspring. For example, by allowing only the most similar individuals to recombine, we can encourage 108 the formation of clusters of similar individuals, or multiple "subspecies" within the population. A related approach is to spatially distribute individuals and allow only nearby individuals to recombine.	

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

148/189	SUBMITTED TEXT	105 WORDS	100% MATCHING TEXT	105 WORDS
	Population Evolution and the Schema Theorem It is interesting to ask whether one can mathematically characterize the evolution over time of the population within a GA. The schema theorem provides one such characterization. It is based on the concept of schemas, or patterns that describe sets of bit strings. To be precise, a schema is any string composed of 0s, 1s, and *'s. Each schema represents the set of bit strings containing the indicated 0s and 1s, with each "*" interpreted as a "don't care." For example, the schema 0*10 represents the set of bit strings that includes exactly 0010 and 01 10.		Population Evolution and the Schema Theorem It is interesting to ask whether one can mathematically characterize the evolution over time of the population within a GA. The schema theorem provides one such characterization. It is based on the concept of schemas, or patterns that describe sets of bit strings. To be precise, a schema is any string composed of 0s, 1s, and *'s. Each schema represents the set of bit strings containing the indicated 0s and 1s, with each "*" interpreted as a "don't care." For example, the schema 0*10 represents the set of bit strings that includes exactly 0010 and 01 10.	

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

149/189	SUBMITTED TEXT 255 WORDS	96% MATCHING TEXT 255 WORDS
	An individual bit string can be viewed as a representative of each of the different schemas that it matches. For example, the bit string 0010 can be thought of as a representative of 2 4 distinct schemas including 00**, 0* 10, ****, etc. Similarly, a population of bit strings can be viewed in terms of the set of schemas that it represents and the number of individuals associated with each of these schemas. The schema theorem characterizes the evolution of the population within a GA in terms of the number of instances representing each schema. Let m(s, t) denote the number of instances of schema s in the population at time t (i.e., during the t th generation). The schema theorem describes the expected value of m (s, t+1) in terms of m(s, t) and other properties of the schema, population, and GA algorithm parameters. The evolution of the population in the GA depends on the selection step, the recombination step, and the mutation step. Let us start by considering just the effect of the selection step. Let f (h) denote the fitness of the individual bit string h and (t) denote the average fitness of all individuals in the population at time t. Let n be the total number of individuals in the population. Let h∈s ∩p , indicate that the individual h is both a representative of schema s and a member of the population at time t. Finally, let ?(	An individual bit string can be viewed as a representative of each of the different schemas that it matches. For example, the bit string 0010 can be thought of as a representative of 2 4 distinct schemas including 00**, 0* 10, ****, etc. Similarly, a population of bit strings can be viewed in terms of the set of schemas that it represents and the number of individuals associated with each of these schema. The schema theorem characterizes the evolution of the population within a GA in terms of the number of instances representing each schema. Let m(s, t) denote the number of instances of schema s in the population at time t (i.e., during the t th generation). The schema theorem describes the expected value of m(s,t+1) in terms of m(s, t) and other properties of the schema, population, and GA algorithm parameters. The evolution of the population in the GA depends on the selection step, the recombination step, and the mutation step. Let us start by considering just the effect of the selection step. Let f (h) denote the fitness of the individual bit string h and ? ?tf ? denote the average fitness of all individuals in the population at time t. Let n be the total number of individuals in the population. Let p t indicate that the individual h is both a representative of schema s and a member of the population at time t. Finally, let ? ?

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

150/189	SUBMITTED TEXT 144 WORDS	94% MATCHING TEXT 144 WORDS
	denote the average fitness of instances of schema s in the population at time t. Let's calculate the expected value of m(s,t+1), which we denote E[m(s,t+1)]. We can calculate E[m(s,t+1)] using the probability distribution for selection given in equation, which can be restated using our current terminology as follows Now if we select one member for the new population according to this probability distribution, then the probability that we will select a representative of schema s is The second step above follows from the fact that by definition, Reinforcement Learning Equation gives the probability that a single hypothesis selected by the GA will be an instance of schema s. Therefore, the expected number of instances of s resulting from the n independent selection steps that create the entire new generation is just n times this probability.	denote the average fitness of instances of schema s in the population at time t. We are interested in calculating the expected value of m(s,t+1), which we denote E[m(s,t+1)]. We can calculate E[m(s,t+1)] using the probability distribution for selection given in Equation, which can be restated using our current terminology as follows Now if we select one member for the new population according to this probability distribution, then the probability that we will select a representative of schema s is 109 The second step above follows from the fact that by definition, Equation gives the probability that a single hypothesis selected by the GA will be an instance of schema s. Therefore, the expected number of instances of s resulting from the n independent selection steps that create the entire new generation is just n times this probability.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

151/189	SUBMITTED TEXT 36 WORDS	100% MATCHING TEXT 36 WORDS
	While the above analysis considered only the selection step of the GA, the crossover and mutation steps must be considered as well. The schema theorem considers only the possible negative influence of these genetic operators	While the above analysis considered only the selection step of the GA, the crossover and mutation steps must be considered as well. The schema theorem considers only the possible negative influence of these genetic operators (

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

152/189	SUBMITTED TEXT 109 WORDS	100% MATCHING TEXT 109 WORDS
	The full schema theorem thus provides a lower bound on the expected frequency of schema s, as follows: Here, p c is the probability that the single-point crossover operator will be applied to an arbitrary individual, and p m is the probability that an arbitrary bit of an arbitrary individual will be mutated by the mutation operator. o(s) is the number of defined bits in schema s, where 0 and 1 are defined bits, but * is not. d(s) is the distance between the leftmost and rightmost defined bits in s. Finally, l is the length of the individual bit strings in the population.	The full schema theorem thus provides a lower bound on the expected frequency of schema s, as follows: Here, p c is the probability that the single-point crossover operator will be applied to an arbitrary individual, and p m , is the probability that an arbitrary bit of an arbitrary individual will be mutated by the mutation operator. o(s) is the number of defined bits in schema s, where 0 and 1 are defined bits, but * is not. d(s) is the distance between the leftmost and rightmost defined bits in s. Finally, l is the length of the individual bit strings in the population.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

153/189	SUBMITTED TEXT 25 WORDS	86% MATCHING TEXT 25 WORDS
	leftmost term in the equation describes the effect of the selection step. The middle term describes the effect of the single-point crossover operator-in particular.	leftmost term in Equation is identical to the term from Equation and describes the effect of the selection step. The middle term describes the effect of the single-point crossover operator-in particular,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

154/189	SUBMITTED TEXT 127 WORDS	97% MATCHING TEXT 127 WORDS
	The rightmost term describes the probability that an arbitrary individual representing schema s will still represent schema s following application of the mutation operator. Note that the effects of single-point crossover and mutation increase with the number of defined bits o(s) in the schema and with the distance d(s) between the defined bits. Thus, the schema theorem can be roughly interpreted as stating that more fit schemas will tend to grow in influence, especially schemas containing a small number of defined bits (i.e., containing a large number of *'s), and especially when these defined bits are near one another within the bit string. The schema theorem is perhaps the Reinforcement Learning most widely cited characterization of population evolution within a GA. 15.5.3	The rightmost term describes the probability that an arbitrary individual representing schema s will still represent schema s following application of the mutation operator. Note that the effects of single-point crossover and mutation increase with the number of defined bits o(s) in the schema and with the distance d(s) between the defined bits. Thus, the schema theorem can be roughly interpreted as stating that more fit schemas will tend to grow in influence, especially schemas containing a small number of defined bits (i.e., containing a large number of *'s), and especially when these defined bits 110 are near one another within the bit string. The schema theorem is perhaps the most widely cited characterization of population evolution within a GA.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

155/189	SUBMITTED TEXT 142 WORDS	98% MATCHING TEXT 142 WORDS
	Genetic Programming Genetic programming (GP) is a form of evolutionary computation in which the individuals in the evolving population are computer programs rather than bit strings. The basic genetic programming approach presents a broad range of simple programs that can be successfully learned by GP. Representing Programs Programs manipulated by a GP are typically represented by trees corresponding to the parse tree of the program. Each function call is represented by a node in the tree, and the arguments to the function are given by its descendant nodes. For example, below Figure 15.2 illustrates this tree representation for the function . To apply genetic programming to a particular domain, the user must define the primitive functions to be considered (e.g., sin, cos, √ , +, -, exponential~), as well as the terminals (e.g., x, y, constants such as 2).	GENETIC PROGRAMMING Genetic programming (GP) is a form of evolutionary computation in which the individuals in the evolving population are computer programs rather than bit strings. The basic genetic programming approach and presents a broad range of simple programs that can be successfully learned by GP. Representing Programs Programs manipulated by a GP are typically represented by trees corresponding to the parse tree of the program. Each function call is represented by a node in the tree, and the arguments to the function are given by its descendant nodes. For example, below Figure illustrates this tree representation for the function sin(x) + ? ? y x ? 2 . To apply genetic programming to a particular domain, the user must define the primitive functions to be considered (e.g., sin, cos, , +, -, exponential~), as well as the terminals (e.g., x, y, constants such as 2).

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

156/189	SUBMITTED TEXT 85 WORDS	100% MATCHING TEXT 85 WORDS
	The genetic programming algorithm then uses an evolutionary search to explore the vast space of programs that can be described using these primitives. As in a genetic algorithm, the prototypical genetic programming algorithm maintains a population of individuals (in this case, program trees). On each iteration, it produces a new generation of individuals using selection, crossover, and mutation. The fitness of a given individual program in the population is typically determined by executing the program on a set of training data. Crossover	The genetic programming algorithm then uses an evolutionary search to explore the vast space of programs that can be described using these primitives. As in a genetic algorithm, the prototypical genetic programming algorithm maintains a population of individuals (in this case, program trees). On each iteration, it produces a new generation of individuals using selection, crossover, and mutation. The fitness of a given individual program in the population is typically determined by executing the program on a set of training data. Crossover

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

157/189	SUBMITTED TEXT 22 WORDS	100% MATCHING TEXT 22 WORDS
	operations are performed by replacing a randomly chosen subtree of one parent program by a subtree from the other parent program (	operations are performed by replacing a randomly chosen subtree of one parent program by a subtree from the other parent program. 111

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

158/189	SUBMITTED TEXT 27 WORDS	93% MATCHING TEXT 27 WORDS
	parent program trees Figure 15.3 illustrates a typical crossover operation. It describes a set of experiments applying a GP to a number of applications. In	parent program. 111 Above Figure illustrates a typical crossover operation. It describes a set of experiments applying a GP to a number of applications. In

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

159/189	SUBMITTED TEXT 57 WORDS	100% MATCHING TEXT 57 WORDS
	of the current population, selected probabilistically according to fitness, is retained unchanged in the next generation. The remainder of the new generation is created by applying crossover to pairs of programs from the current generation, again selected probabilistically according to their fitness. The mutation operator was not used in this particular set of experiments. 15.7	of the current population, selected probabilistically according to fitness, is retained unchanged in the next generation. The remainder of the new generation is created by applying crossover to pairs of programs from the current generation, again selected probabilistically according to their fitness. The mutation operator was not used in this particular set of experiments.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

160/189	SUBMITTED TEXT 12 WORDS	100% MATCHING TEXT 12 WORDS
	GAS can search spaces of hypotheses containing complex interacting parts,	GAS can search spaces of hypotheses containing complex interacting parts,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

161/189	SUBMITTED TEXT 17 WORDS	100% MATCHING TEXT 17 WORDS
	where the impact of each part on overall hypothesis fitness may be difficult to model. ●	where the impact of each part on overall hypothesis fitness may be difficult to model. ?

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

162/189	SUBMITTED TEXT 27 WORDS	100% MATCHING TEXT 27 WORDS
	Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover. ● The	Hypotheses in GAS are often represented by bit strings, so that they can be easily manipulated by genetic operators such as mutation and crossover. The

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

163/189	SUBMITTED TEXT 25 WORDS	100% MATCHING TEXT 25 WORDS
	The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population. 15.8	The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

164/189	SUBMITTED TEXT 14 WORDS	100% MATCHING TEXT 14 WORDS
	two hypotheses are first chosen at random from the current population. ●	two hypotheses are first chosen at random from the current population.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

165/189	SUBMITTED TEXT 17 WORDS	80% MATCHING TEXT 17 WORDS
	selection: In this method, the hypotheses in the current population are first sorted by fitness. ●	selection. In another method called rank selection, the hypotheses in the current population are first sorted by fitness.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

166/189	SUBMITTED TEXT 39 WORDS	100% MATCHING TEXT 39 WORDS
	is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of this individual and very similar individuals take over a large fraction of the population. ●	is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of this individual and very similar individuals take over a large fraction of the population.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

167/189	SUBMITTED TEXT 12 WORDS	100% MATCHING TEXT 12 WORDS
	the hypotheses in the current population are first sorted by fitness.	the hypotheses in the current population are first sorted by fitness.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

168/189	SUBMITTED TEXT 20 WORDS	100% MATCHING TEXT 20 WORDS
	them to produce improved results. Ensemble learning usually produces more accurate solutions than a single model would. 16.1	them to produce improved results. Ensemble learning usually produces more accurate solutions than a single model would. ?

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

169/189	SUBMITTED TEXT 86 WORDS	90% MATCHING TEXT 86 WORDS
	Ensemble Learning is a technique that creates multiple models and then combines them to produce improved results. Ensemble learning usually produces more accurate solutions than a single model would. Ensemble learning methods are applied to regression as well as classification: ● Ensemble learning for regression creates multiple repressors i.e. multiple regression models such as linear, polynomial, etc. ● Ensemble learning for classification creates multiple classifiers i.e. multiple classification models such as logistic, decision trees, KNN, SVM, etc. Figure 16.3 Ensemble learning view	Ensemble Learning is a technique that create multiple models and then combine them them to produce improved results. Ensemble learning usually produces more accurate solutions than a single model would. ? Ensemble learning methods applied to regression as well as classification. o Ensemble learning for regression creates multiple repressors i.e. multiple regression models such as linear, polynomial, etc. o Ensemble learning for classification creates multiple classifiers i.e. multiple classification models such as logistic, decision tress, KNN, SVM, etc. Figure 1: Ensemble learning view

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

170/189	SUBMITTED TEXT 36 WORDS	86% MATCHING TEXT 36 WORDS
	There are also different ways the multiple base-learners are combined to generate the final output: ● Multiexpert combination: These methods have base-learners that work in parallel. These methods can in turn be divided into two:	There are also different ways the multiple base-learners are combined to generate the final output: Figure2: General Idea - Combining Multiple Learners 61 Multiexpert combination Multiexpert combination methods have base-learners that work in parallel. These methods can in turn be divided into two: ?

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

171/189	SUBMITTED TEXT 39 WORDS	96% MATCHING TEXT 39 WORDS
	In the global approach, also called learner fusion, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking. b) In the local approach, or learner selection, for example, in	In the global approach, also called learner fusion, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking. ? In the local approach, or learner selection, for example, in

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

172/189	SUBMITTED TEXT 192 WORDS	97% MATCHING TEXT 192 WORDS
	mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output. ● Multistage combination: These methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. The idea is that the base-learners (or the different representations they use) are sorted in increasing complexity so that a complex base-learner is not used (or its complex representation is not extracted) unless the preceding simpler base-learners are not confident. An example is cascading. Let us say that we have L base-learners. We denote by d j (x) the prediction of base- learner M j given the arbitrary dimensional input x. In the case of multiple representations, each M j uses a different input representation x j . The final prediction is calculated from the predictions of the base-learners: y = f (d 1 , d 2 , . . . , d L |Φ) where f (·) is the combining function with Φ denoting its parameters. Figure 16.4 Base-learners are	mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output. Multistage Multistage combination methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. The idea is that the base-learners (or the different representations they use) are sorted in increasing complexity so that a complex base-learner is not used (or its complex representation is not extracted) unless the preceding simpler base-learners are not confident. An example is cascading. Let us say that we have L base-learners. We denote by dj(x) the prediction of base-learner M j given the arbitrary dimensional input x. In the case of multiple representations, each M j uses a different input representation x j . The final prediction is calculated from the predictions of the base-learners: y = f (d 1 , d 2 , . . . , d L |Φ) where f (·) is the combining function with Φ denoting its parameters. 62 Figure 1: Base-learners are

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

173/189	SUBMITTED TEXT 66 WORDS	96% MATCHING TEXT 66 WORDS
	When there are K outputs, for each learner there are d ji (x), i = 1, . . . , K, j = 1, . . . , L, and, combining them, we also generate K values, y i , i = 1, . . . , K and then for example in classification, we choose the class with the maximum y i value:	When there are K outputs, for each learner there are d ji (x), i = 1, . . . , K, j = 1, . . . , L, and, combining them, we also generate K values, y i , i = 1, . . . , K and then for example in classification, we choose the class with the maximum yi value: 3.2

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

174/189	SUBMITTED TEXT 75 WORDS	95% MATCHING TEXT 75 WORDS
	base-learners are d j and their outputs are combined using f (·). This is for a single output; in the case of classification, each base-learner has K outputs that are separately used to calculate y i , and then we choose the maximum. Note that here all learners observe the same input; it may be the case that different learners observe different representations of the same input object or event. 16.5	Base-learners are dj and their outputs are combined using f (·). This is for a single output; in the case of classification, each base-learner has K outputs that are separately used to calculate y i , and then we choose the maximum. Note that here all learners observe the same input; it may be the case that different learners observe different representations of the same input object or event.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

175/189	SUBMITTED TEXT 12 WORDS	100% MATCHING TEXT 12 WORDS
	voting, Error-Correcting Output Codes, Bagging: Random Forest Trees, Boosting: Adaboost, Stacking.	Voting, Error-Correcting Output Codes, Bagging: Random Forest Trees, Boosting: Adaboost, Stacking.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

176/189	SUBMITTED TEXT 81 WORDS	92% MATCHING TEXT 81 WORDS
	Voting The simplest way to combine multiple classifiers is by voting, which corresponds to taking a linear combination of the learners (see figure 16.4): This is also known as ensembles and linear opinion pools. In the simplest case, all learners are given equal weight and we have simple voting that corresponds to taking an average. Still, taking a (weighted) sum is only one of the possibilities and there are also other combination rules, as shown in table 16.2 (	Voting The simplest way to combine multiple classifiers is by voting, which corresponds to taking a linear combination of the learn ers, Refer This is also known as ensembles and linear opinion pools. In the sim plest case, all learners are given equal weight and we have simple voting that corresponds to taking an average. Still, taking a (weighted) sum is only one of the possibilities and there are also other combination rules, as shown in table 1.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

177/189	SUBMITTED TEXT 19 WORDS	100% MATCHING TEXT 19 WORDS
	If the outputs are not posterior probabilities, these rules require that outputs be normalized to the same scale (	If the outputs are not posterior probabilities, these rules require that outputs be normalized to the same scale

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

178/189	SUBMITTED TEXT	156 WORDS	98% MATCHING TEXT	156 WORDS

Table 16.2 Classifier combination rules An example of the use of these rules is shown in table 16.3, which demonstrates the effects of different rules. Sum rule is the most intuitive and is the most widely used in practice. Median rule is more robust to outliers; minimum and maximum rules are pessimistic and optimistic, respectively. With the product rule, each learner has veto power; regardless of the other ones, if one learner has an output of 0, the overall output goes to 0. Note that after the combination rules, y i do not necessarily sum up to 1. Table 16.3 Example of combination rules on three learners and three classes In weighted sum, d ji is the vote of learner j for class C i and w j is the weight of its vote. Simple voting is a special case where all voters have equal weight, namely,	Table 1 - Classifier combination rules 63 An example of the use of these rules is shown in table 2, which demonstrates the effects of different rules. Sum rule is the most intuitive and is the most widely used in practice. Median rule is more robust to outliers; minimum and maximum rules are pessimistic and optimistic, respectively. With the product rule, each learner has veto power; regardless of the other ones, if one learner has an output of 0, the overall output goes to 0. Note that after the combination rules, yi do not necessarily sum up to 1. Table 2: Example of combination rules on three learners and three classes In weighted sum, d ji is the vote of learner j for class C i and w j is the weight of its vote. Simple voting is a special case where all voters have equal weight, namely,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

179/189	SUBMITTED TEXT	99 WORDS	100% MATCHING TEXT	99 WORDS

L. In classification, this is called plurality voting where the class having the maximum number of votes is the winner. When there are two classes, this is majority voting where the winning class gets more than half of the votes. If the voters can also supply the additional information of how much they vote for each class (e.g., by the posterior probability), then after normalization, these can be used as weights in a weighted voting scheme. Equivalently, if d ji are the class posterior probabilities, P(C i | x,M j), then we can just	L. In classification, this is called plurality voting where the class having the maximum number of votes is the winner. When there are two classes, this is majority voting where the winning class gets more than half of the votes. If the voters can also supply the additional information of how much they vote for each class (e.g., by the posterior probability), then after normalization, these can be used as weights in a weighted voting scheme. Equivalently, if d ji are the class posterior probabilities, P(C i | x,M j), then we can just

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

180/189	SUBMITTED TEXT 296 WORDS	100% MATCHING TEXT 296 WORDS
	sum them up (w j = 1/L) and choose the class with maximum y i . In the case of regression, simple or weighted averaging or median can be used to fuse the outputs of base-regressors. Median is more robust to noise than the average. Another possible way to find w j is to assess the accuracies of the learners (regressor or classifier) on a separate validation set and use that information to compute the weights, so that we give more weights to more accurate learners. Voting schemes can be seen as approximations under a Bayesian framework with weights approximating prior model probabilities, and model decisions approximating model-conditional likelihoods. Simple voting corresponds to a uniform prior. If we have a prior distribution preferring simpler models, this would give larger weights to them. We cannot integrate over all models; we only choose a subset for which we believe P(M j) is high, or we can have another Bayesian step and calculate P(C i | x,M j), the probability of a model given the sample, and sample high probable models from this density. Let us assume that d j are iid with expected value E[d j] and variance Var(d j), then when we take a simple average with w j = 1/L, the expected value and variance of the output are We see that the expected value does not change, so the bias does not change. But variance, and therefore mean square error, decreases as the number of independent voters, L, increases. In the general case, which implies that if learners are positively correlated, variance (and error) increase. We can thus view using different algorithms and input features as efforts to decrease,	sum them up (w j = 1/L) and choose the class with maximum y i . In the case of regression, simple or weighted averaging or median can be used to fuse the outputs of base-regressors. Median is more robust to noise than the average. Another possible way to find w j is to assess the accuracies of the learners (regressor or classifier) on a separate validation set and use that information to compute the weights, so that we give more weights to more accurate learners. Voting schemes can be seen as approximations under a Bayesian framework with weights approximating prior model probabilities, and model decisions approximating model-conditional likelihoods. Simple voting corresponds to a uniform prior. If we have a prior distribution preferring simpler models, this would give larger weights to them. We cannot integrate over all models; we only choose a subset for which we believe P(M j) is high, or we can have another Bayesian step and calculate P(C i | x,M j), the probability of a model given the sample, and sample high probable models from this density. Let us assume that d j are iid with expected value E[d j] and variance Var(d j), then when we take a simple 64 average with w j = 1/L, the expected value and variance of the output are We see that the expected value does not change, so the bias does not change. But variance, and therefore mean square error, decreases as the number of independent voters, L, increases. In the general case, which implies that if learners are positively correlated, variance (and error) increase. We can thus view using different algorithms and input features as efforts to decrease,

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

181/189	SUBMITTED TEXT 231 WORDS	98% MATCHING TEXT 231 WORDS
	In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method. In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the Reinforcement Learning previous learners. The original boosting algorithm combines three weak learners to generate a strong learner. A weak learner has error probability less than 1/2, which makes it better than random guessing on a two-class problem, and a strong learner has arbitrarily small error probability. Boosting Concept Given a large training set, we randomly divide it into three. We use X 1 and train d 1 . We then take X 2 and feed it to d 1 . We take all instances misclassified by d 1 and also as many instances on which d 1 is correct from X 2 , and these together form the training set of d 2 . We then take X 3 and feed it to d 1 and d 2 . The instances on which d 1 and d 2 disagree form the training set of d 3 . During testing, given an instance, we give it to d 1 and d 2 ; if they agree, that is the response, otherwise the response of d 3 is taken as the output.	In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method. In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the previous learners. The original boosting algorithm combines three weak learners to generate a strong learner. A weak learner has error probability less than 1/2, which makes it better than random guessing on a two-class problem, and a strong learner has arbitrarily small error probability. Original Concept Given a large training set, we randomly divide it into three. We use X1 and train d1. We then take X2 and feed it to d1. We take all instances misclassified by d1 and also as many instances on which d1 is correct from X2, and these together form the training set of d2. We then take X3 and feed it to d1 and d2. The instances on which d1 and d2 disagree form the training set of d3. During testing, given an instance, we give it to d1 and d2; if they agree, that is the response, otherwise the response of d3 is taken as the output. 1.

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

182/189 SUBMITTED TEXT 30 WORDS	100% MATCHING TEXT 30 WORDS
overall system has reduced error rate, and the error rate can arbitrarily be reduced by using such systems recursively, that is, a boosting system of three models used as	overall system has reduced error rate, and the error rate can arbitrarily be reduced by using such systems recursively, that is, a boosting system of three models used as

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

183/189 SUBMITTED TEXT 171 WORDS	99% MATCHING TEXT 171 WORDS
in a higher system. Though it is quite successful, the disadvantage of the original boosting method is that it requires a very large training sample. The sample should be divided into three and furthermore, the second and third classifiers are only trained on a subset on which the previous ones err. So unless one has a quite large training set, d 2 and d 3 will not have training sets of reasonable size. AdaBoost Freund and Schapire (1996) proposed a variant, named AdaBoost, short for adaptive boosting, that uses the same training set over and over and thus need not be large, but the classifiers should be simple so that they do not overfit. AdaBoost can also combine an arbitrary number of baselearners, not three. The idea is to modify the probabilities of drawing the instances as a function of the error. Let us say p t j denotes the probability that the instance pair (x t , r t) is drawn to train the	in a higher system. Though it is quite successful, the disadvantage of the original boosting method is that it requires a very large training sample. The sample should be divided into three and furthermore, the second and third classifiers are only trained on a subset on which the previous ones err. So unless one has a quite large training set, d 2 and d 3 will not have training sets of reasonable size. 3.5.1AdaBoost Freund and Schapire (1996) proposed a variant, named AdaBoost, short for adaptive boosting, that uses the same training set over and over and thus need not be large, but the classifiers should be simple so that they do not overfit. AdaBoost can also combine an arbitrary number of baselearners, not three. AdaBoost algorithm 72 The idea is to modify the probabilities of drawing the instances as a function of the error. Let us say p t j denotes the probability that the instance pair (x t , r t) is drawn to train the

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

184/189 SUBMITTED TEXT 36 WORDS	100% MATCHING TEXT 36 WORDS
base-learner. Initially, all p t 1 = 1/N. Then we add new base-learners as follows, starting from j = 1: Є j denotes the error rate of d j .	base-learner. Initially, all p t 1 = 1/N. Then we add new base-learners as follows, starting from j = 1: Є j denotes the error rate of d j .

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

185/189 SUBMITTED TEXT 108 WORDS	99% MATCHING TEXT 108 WORDS
AdaBoost requires that learners are weak, that is, Є j &gt; 1/2,∀j; if not, we stop adding new base learners. Note that this error rate is not on the original problem but on the dataset used at step j. We define βj = Є j /(1 – Є j) &gt; 1, and we set p t j +1 = β j p t j if d j correctly classifies xt ; otherwise, p t j+1 = p t j . Because p t j+1 should be probabilities, there is a normalization where we divide p t j+1 by ∑t	AdaBoost requires that learners are weak, that is, Є j &gt; 1/2,∀j; if not, we stop adding new base- learners. Note that this error rate is not on the original problem but on the dataset used at step j. We define βj = Є j /(1 –Є j) &gt; 1, and we set p t j+1 = β j p t j if d j correctly classifies xt ; otherwise, p t j +1 = p t j. Because p t j+1 should be probabilities, there is a normalization where we divide p t j+1 by p t

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

186/189	SUBMITTED TEXT	174 WORDS	100% MATCHING TEXT	174 WORDS
	p t j+1 , so that they sum up to 1. This has the effect that the probability of a correctly classified instance is decreased, and the probability of a misclassified instance increases. Then a new sample of the same size is drawn from the original sample according to these modified probabilities, p t j+1 with replacement, and is used to train d j+1 . This has the effect that d j+1 focuses more on instances misclassified by d j ; that is why the base-learners are chosen to be simple and not accurate, since otherwise the next training sample would contain only a few outlier and noisy instances repeated many times over. For example, with decision trees, decision stumps, which are trees grown only one or two levels, are used. So it is clear that these would have bias but the decrease in variance is larger and the overall error decreases. An algorithm like the linear discriminant has low variance, and we cannot gain by AdaBoosting linear discriminants.		p t j+1 , so that they sum up to 1. This has the effect that the probability of a correctly classified instance is decreased, and the probability of a misclassified instance increases. Then a new sample of the same size is drawn from the original sample according to these modified probabilities, p t j+1 with replacement, and is used to train d j+1 . This has the effect that d j+1 focuses more on instances misclassified by d j ; that is why the base-learners are chosen to be simple and not accurate, since otherwise the next training sample would contain only a few outlier and noisy instances repeated many times over. For example, with decision trees, decision stumps, which are trees grown only one or two levels, are used. So it is clear that these would have bias but the decrease in variance is larger and the overall error decreases. An algorithm like the linear discriminant has low variance, and we cannot gain by AdaBoosting linear discriminants. . 3.5.2	

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

187/189	SUBMITTED TEXT	19 WORDS	79% MATCHING TEXT	19 WORDS
	Ensemble Learning is a technique that creates multiple models and then combines them to produce improved results. ●		Ensemble Learning is a technique that create multiple models and then combine them them to produce improved results.	

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

188/189	SUBMITTED TEXT	28 WORDS	100% MATCHING TEXT	28 WORDS
	methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. ●		methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough.	

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf

189/189	SUBMITTED TEXT	11 WORDS	100% MATCHING TEXT	11 WORDS
	class having the maximum number of votes is the winner.		class having the maximum number of votes is the winner.	

W https://mrcet.com/downloads/digital_notes/CSE/IV%20Year/MACHINE%20LEARNING(R17A0534).pdf