UNIT I

1

INTRODUCTION TO MACHINE LEARNING

Unit Structure

1.0 INTRODUCTION

A human child learns new things and uncovers the structure of their world year by year as they grow to adulthood. A child's brain and senses perceive the facts of their surroundings and gradually learn the hidden patterns of life which help the child to craft logical rules to identify learned patterns. The learning process of the human brain makes humans the most sophisticated living creature of this world. Learning continuously by discovering hidden patterns and then innovating on those patterns enables us to make ourselves better and better throughout our lifetime. Superficially, we can draw some motivational similarities between the learning process of the human brain and the concepts of machine learning. (jlooper, n.d.)

The human brain perceives things from the real world, processes the perceived information, makes rational decisions, and performs certain actions based on circumstances. When we program a replica of the intelligent behavioural process to a machine, it is called artificial intelligence (AI).

Machine learning (ML) is an important subset of artificial intelligence. ML is concerned with using specialized algorithms to

uncover meaningful information and find hidden patterns from perceived data to support the logical decision-making process**.**

1.1 MACHINE LEARNING

Machine learning, from a systems perspective, is defined as the creation of automated systems that can learn hidden patterns from data to aid in making intelligent decisions.

This motivation is loosely inspired by how the human brain learns certain things based on the data it perceives from the outside world. Machine learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience.

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output.

Two definitions of Machine Learning are offered.

Arthur Samuel described it as: "The field of study that gives computers the ability to learn from data without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition. According to him, "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 in T, as measured by P, improves with experience E."

Example: playing checkers.

T = the task of playing checkers.

P = the probability that the program will win the next game.

E = the experience of playing many games of checkers

Let us now understand, Supervised Machine Learning, Unsupervised Machine Learning and Reinforcement Learning:

Supervised Machine Learning:
Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data,

machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

Supervised learning can be further divided into two types of problems:

1. Regression:
Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Linear Regression, Regression Trees, Non-Linear Regression, Bayesian Linear Regression, Polynomial Regression are some popular Regression algorithms which come under supervised learning.

2. Classification:
Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc. Spam Filtering, Random Forest, Decision Trees, Logistic Regression, Support vector Machines are some examples of classification.

Unsupervised Machine Learning:
There may be many cases in which we do not have labeled data and need to find the hidden patterns from the given dataset. So, to solve such types of cases in machine learning, we need unsupervised learning techniques.
unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

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 dataset, group that data according to similarities, and represent that dataset in a compressed format.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

The unsupervised learning algorithm can be further categorized into two types of problems:

1. Clustering
2. Association

1. Clustering:
Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

2. 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 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.

K-means clustering, KNN (k-nearest neighbors), Hierarchal clustering, Anomaly detection, Neural Networks are the examples of unsupervised learning.

Reinforcement Learning:
Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty. In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.

Since there is no labelled data, so the agent is bound to learn by its experience only. RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc. The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.

The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that "Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that". How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.

1.2 EXAMPLES OF MACHINE LEARNING PROBLEMS:

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:

1. Image Recognition:
Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, Automatic friend tagging suggestion:

Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's face detection and recognition algorithm.

It is based on the Facebook project named "Deep Face," which is responsible for face recognition and person identification in the picture.

2. Speech Recognition:
While using Google, we get an option of "Search by voice," it comes under speech recognition, and it's a popular application of machine learning.

Speech recognition is a process of converting voice instructions into text, and it is also known as "Speech to text", or "Computer speech recognition." At present, machine learning algorithms are widely used by various applications of speech recognition. Google assistant, Siri, Cortana, and Alexa are using speech recognition technology to follow the voice instructions.

3. Traffic prediction:
If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.

It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

- Real Time location of the vehicle form Google Map app and sensors
- Average time has taken on past days at the same time.

Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

4. Product recommendations:
Machine learning is widely used by various e-commerce and entertainment companies such as Amazon, Netflix, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.

Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest. As similar, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning.

5. Self-driving cars:
One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

6. Email Spam and Malware Filtering:
Whenever we receive a new email, it is filtered automatically as important, normal, and spam. We always receive an important mail in our inbox with the important symbol and spam emails in our spam box, and the technology behind this is Machine learning. Below are some spam filters used by Gmail:

- Content Filter
- Header filter
- General blacklists filter
- Rules-based filters
- Permission filters

Some machine learning algorithms such as Multi-Layer Perceptron, Decision tree, and Naïve Bayes classifier are used for email spam filtering and malware detection.

7. Virtual Personal Assistant:
We have various virtual personal assistants such as Google assistant, Alexa, Cortana, Siri. As the name suggests, they help us in finding the

information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, open an email, Scheduling an appointment, etc.

These virtual assistants use machine learning algorithms as an important part. These assistants record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

8. Online Fraud Detection:
Machine learning is making our online transaction safe and secure by detecting fraud transaction. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as fake accounts, fake ids, and steal money in the middle of a transaction. So, to detect this, Feed Forward Neural network helps us by checking whether it is a genuine transaction or a fraud transaction.

For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round. For each genuine transaction, there is a specific pattern which gets change for the fraud transaction hence, it detects it and makes our online transactions more secure.

9. Stock Market trading:
Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's long short-term memory neural network is used for the prediction of stock market trends.

10. Medical Diagnosis:
In medical science, machine learning is used for diseases diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain. It helps in finding brain tumours and other brain-related diseases easily.

11. Automatic Language Translation:
Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.

The technology behind the automatic translation is a sequence-to-sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

1.3 STRUCTURE OF LEARNING

Like all other machine learning models, patterns are a manifestation of underlying structure in the data. Sometimes this structure takes the form of a single hidden or latent variable, much like unobservable but nevertheless explanatory quantities in physics, such as energy. Consider the following matrix:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix}$$

(FLACH, 2012)

Imagine these represent ratings by six different people (in rows), on a scale of 0 to 3, of four different films – say Khosla Ka Ghosla (KG), Drishyam (D), BADLA (B), Hera Phery (HP), (in columns, from left to right). BADLA (B) seems to be the most popular of the four with an average rating of 1.5, and Khosla Ka Ghosla (KG) is the least appreciated with an average rating of 0.5. Try to find a structure in this matrix. (FLACH, 2012)

Try to look for columns or rows that are combinations of other columns or rows. For instance, the third column turns out to be the sum of the first and second columns. Similarly, the fourth row is the sum of the first and second rows. What this means is that the fourth person combines the ratings of the first and second person. Similarly, BADLA (B)'s ratings are the sum of the ratings of the first two films. This is made more explicit by writing the matrix as the following product:

$$\begin{array}{c} \text{genres:} \\ \begin{matrix} \text{D} & \text{C} & \text{DC} & \text{CCo} \\ \text{KG} & \text{D} & \text{B} & \text{HP} \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} \\ \begin{matrix} \text{D} & \text{C} & & \text{CCo-C} \end{matrix} \end{array} = \begin{array}{c} \text{genres:} \\ \begin{matrix} \text{D} & \text{C} & \text{Co} \\ & & \text{(CCo – C)} \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \begin{array}{c} \begin{matrix} & \text{D} & \text{C} & \text{Co} \end{matrix} \\ \begin{matrix} \text{D} \\ \text{C} \\ \text{Co} \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array} \begin{array}{c} \begin{matrix} & \text{KG} & \text{D} & \text{B} & \text{HP} \end{matrix} \\ \begin{matrix} \text{D} \\ \text{C} \\ \text{Co} \end{matrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

(FLACH, 2012)

Notice that the first and third matrix on the right-hand side are now Boolean, and the middle one is diagonal (all off-diagonal entries are zero). Moreover, these matrices have a very natural interpretation in terms of film genres.

The right-most matrix associates films (in columns) with genres (in rows): Khosla Ka Ghosla (KG) and Drishyam (D) belong to two different genres, say drama and crime, BADLA (B) belongs to both, and Hera Phery (HP) is a crime film and also introduces a new genre (say comedy).

The tall, 6-by-3 matrix then expresses people's preferences in terms of genres: the first, fourth and fifth person like drama, the second, fourth and fifth person like crime films, and the third, fifth and sixth person like comedies. Finally, the middle matrix states that the crime genre is twice as important as the other two genres in terms of determining people's preferences.

1.4 LEARNING VERSUS DESIGNING

According to Arthur Samuel "Machine Learning enables a Machine to Automatically learn from Data, improve performance from an Experience and predict things without explicitly programmed." (https://www.tutorialspoint.com/, n.d.)

In Simple Words, when we fed the Training Data to Machine Learning Algorithm, this algorithm will produce a mathematical model and with the help of the mathematical model, the machine will make a prediction and take a decision without being explicitly programmed, as shown in figure 1.1. Also, during training data, the more machine will work with it the more it will get experience and the more it will get experience the more efficient result is produced.

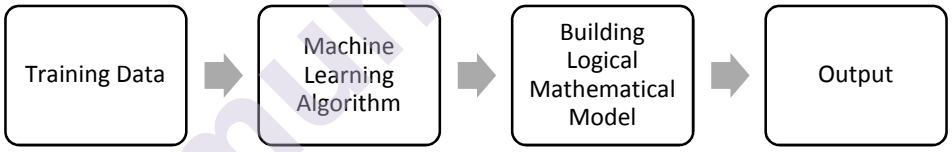


Figure 1.1: Learn from data

Example: In Driverless Car, the training data is fed to Algorithm like how to Drive Car in Highway, Busy and Narrow Street with factors like speed limit, parking, stop at signal etc. After that, a Logical and Mathematical model is created based on that and after that, the car will work according to the logical model. Also, the more data the data is fed the more efficient output is produced.

Designing a Learning System in Machine Learning:
According to Tom Mitchell, "A computer program is said to be learning from experience (E), with respect to some task (T). Thus, the performance measure (P) is the performance at task T, which is measured by P, and it improves with experience E."

Example: In Spam E-Mail detection,

Task, T: To classify mails into Spam or Not Spam.
Performance measure, P: Total percent of mails being correctly classified as being "Spam" or "Not Spam".

Experience, E: Set of Mails with label "Spam"
Steps for Designing Learning System are as shown in figure 1.2 below:

Step 1) Choosing the Training Experience: The very important and first task is to choose the training data or training experience which will be fed to the Machine Learning Algorithm. It is important to note that the data or experience that we fed to the algorithm must have a significant impact on the Success or Failure of the Model. So, training data or experience should be chosen wisely.

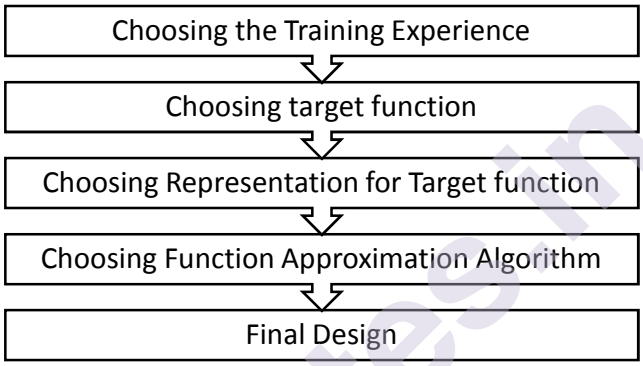


Figure 1.2: Steps for Designing Learning System

Below are the attributes which will impact on Success and Failure of Data:

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. For example, a Machine learning algorithm will get experience while going through a number of different cases and different examples. Thus, Machine Learning Algorithm will get more and more experience by passing through more and more examples and hence its performance will increase.

Step 2) Choosing target function: The next important step is choosing the target function. It means according to the knowledge fed to the algorithm the machine learning will choose NextMove function which will describe what type of legal moves should be taken. For example: While playing chess with the opponent, when opponent will play then the machine learning algorithm will decide what be the number of possible legal moves taken in order to get success.

Step 3) Choosing Representation for Target function: When the machine algorithm will know all the possible legal moves the next step is to choose the optimized move using any representation i.e. using linear Equations, Hierarchical Graph Representation, Tabular form etc. The NextMove function will move the Target move like out of these moves which will provide more success rate. For Example: while playing chess machine have 4 possible moves, so the machine will choose that optimized move which will provide success to it.

Step 4) Choosing Function Approximation Algorithm: An optimized move cannot be chosen just with the training data. The training data had to go through with set of examples and through these examples the training data will approximates which steps are chosen and after that machine will provide feedback on it. For Example: When a training data of Playing chess is fed to algorithm so at that time it is not machine algorithm will fail or get success and again from that failure or success it will measure while next move what step should be chosen and what is its success rate.

Step 5) Final Design: The final design is created at last when system goes from number of examples, failures and success, correct and incorrect decision and what will be the next step etc. Example: DeepBlue is an intelligent computer which is ML-based won chess game against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.

1.5 TRAINING VERSUS TESTING

Training data and test data are two important concepts in machine learning.

Training Data:
The observations in the training set form the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

Test Data:
The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set. If the test set does contain examples from the training set, it will be difficult to assess whether the

algorithm has learned to generalize from the training set or has simply memorized it.

A program that generalizes well will be able to effectively perform a task with new data. In contrast, a program that memorizes the training data by learning an overly complex model could predict the values of the response variable for the training set accurately but will fail to predict the value of the response variable for new examples.

Memorizing the training set is called *over-fitting*. A program that memorizes its observations may not perform its task well, as it could memorize relations and structures that are noise or coincidence. Balancing memorization and generalization, or over-fitting and under-fitting, is a problem common to many machine learning algorithms. *Regularization* may be applied to many models to reduce over-fitting.

In addition to the training and test data, a third set of observations, called a *validation* or *hold-out set,* is sometimes required. The validation set is used to tune variables called *hyper parameters*, which control how the model is learned. The program is still evaluated on the test set to provide an estimate of its performance in the real world; its performance on the validation set should not be used as an estimate of the model's real-world performance since the program has been tuned specifically to the validation data.

It is common to partition a single set of supervised observations into training, validation, and test sets. There are no requirements for the sizes of the partitions, and they may vary according to the amount of data available. It is common to allocate 50 percent or more of the data to the training set, 25 percent to the test set, and the remainder to the validation set.

Some training sets may contain only a few hundred observations; others may include millions. Inexpensive storage, increased network connectivity, the ubiquity of sensor-packed smartphones, and shifting attitudes towards privacy have contributed to the contemporary state of big data, or training sets with millions or billions of examples.

However, machine learning algorithms also follow the maxim "garbage in, garbage out." A student who studies for a test by reading a large, confusing textbook that contains many errors will likely not score better than a student who reads a short but well-written textbook. Similarly, an algorithm trained on a large collection of noisy, irrelevant, or incorrectly labelled data will not perform better than an algorithm trained on a smaller set of data that is more representative of problems in the real world.

Many supervised training sets are prepared manually, or by semi-automated processes. Creating a large collection of supervised data can be

costly in some domains. Fortunately, several datasets are bundled with *scikit-learn*, allowing developers to focus on experimenting with models instead.

During development, and particularly when training data is scarce, a practice called *cross-validation* can be used to train and validate an algorithm on the same data. In cross-validation, the training data is partitioned. The algorithm is trained using all but one of the partitions and tested on the remaining partition. The partitions are then rotated several times so that the algorithm is trained and evaluated on all of the data.

Consider for example that the original dataset is partitioned into five subsets of equal size, labelled A through E. Initially, the model is trained on partitions B through E, and tested on partition A. In the next iteration, the model is trained on partitions A, C, D, and E, and tested on partition B. The partitions are rotated until models have been trained and tested on all of the partitions. Cross-validation provides a more accurate estimate of the model's performance than testing a single partition of the data.

1.6 CHARACTERISTICS OF MACHINE LEARNING TASKS

To understand the actual power of machine learning, we must consider the characteristics of this technology. There are lots of examples that echo the characteristics of machine learning in today's data-rich world. Here are seven key characteristics of machine learning for which companies should prefer it over other technologies:

1. The ability to perform automated data visualization
2. Automation at its best
3. Customer engagement like never before
4. The ability to take efficiency to the next level when merged with IoT
5. The ability to change the mortgage market
6. Accurate data analysis
7. Business intelligence at its best

1. The ability to perform automated data visualization:
A massive amount of data is being generated by businesses and common people on a regular basis. By visualizing notable relationships in data, businesses can not only make better decisions but build confidence as well. Machine learning offers several tools that provide rich snippets of data which can be applied to both unstructured and structured data. With the help of user-friendly automated data visualization platforms in machine learning, businesses can obtain a wealth of new insights to increase productivity in their processes.

2. Automation at its best:

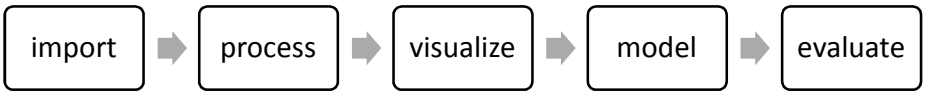


Figure 1.3 Machine Learning workflow

Figure 1.3 shows Machine Learning workflow. One of the biggest characteristics of machine learning is its ability to automate repetitive tasks and thus, increasing productivity. A huge number of organizations are already using machine learning-powered paperwork and email automation. In the financial sector, for example, a huge number of repetitive, data-heavy and predictable tasks are needed to be performed. Because of this, this sector uses different types of machine learning solutions to a great extent. The make accounting tasks faster, more insightful, and more accurate. Some aspects that have been already addressed by machine learning include addressing financial queries with the help of chatbots, making predictions, managing expenses, simplifying invoicing, and automating bank reconciliations.

3. Customer engagement like never before:
For any business, one of the most crucial ways to drive engagement, promote brand loyalty and establish long-lasting customer relationships is by triggering meaningful conversations with its target customer base. Machine learning plays a critical role in enabling businesses and brands to spark more valuable conversations in terms of customer engagement. The technology analyzes particular phrases, words, sentences, idioms, and content formats which resonate with certain audience members. We can think of Pinterest which is successfully using machine learning to personalize suggestions to its users. It uses the technology to source content in which users will be interested, based on objects which they have pinned already.

4. The ability to take efficiency to the next level when merged with IoT:
IoT is being designated as a strategically significant area by many companies. And many others have launched pilot projects to gauge the potential of IoT in the context of business operations. But attaining financial benefits through IoT isn't easy. In order to achieve success, companies, which are offering IoT consulting services and platforms, need to clearly determine the areas that will change with the implementation of IoT strategies. Many of these businesses have failed to address it. In this scenario, machine learning is probably the best technology that can be used to attain higher levels of efficiency. By merging machine learning with IoT, businesses can boost the efficiency of their entire production processes.

5. The ability to change the mortgage market:
It's a fact that fostering a positive credit score usually takes discipline, time, and lots of financial planning for a lot of consumers. When it comes to the lenders, the consumer credit score is one of the biggest measures of creditworthiness that involve a number of factors including payment history, total debt, length of credit history etc. But wouldn't it be great if there is a simplified and better measure? With the help of machine learning, lenders can now obtain a more comprehensive consumer picture. They can now predict whether the customer is a low spender or a high spender and understand his/her tipping point of spending. Apart from mortgage lending, financial institutions are using the same techniques for other types of consumer loans.

6. Accurate data analysis:
Traditionally, data analysis has always been encompassing trial and error method, an approach which becomes impossible when we are working with large and heterogeneous datasets. Machine learning comes as the best solution to all these issues by offering effective alternatives to analyzing massive volumes of data. By developing efficient and fast algorithms, as well as, data-driven models for processing of data in real-time, machine learning is able to generate accurate analysis and results.

7. Business intelligence at its best:
Machine learning characteristics, when merged with big data analytical work, can generate extreme levels of business intelligence with the help of which several different industries are making strategic initiatives. From retail to financial services to healthcare, and many more – machine learning has already become one of the most effective technologies to boost business operations.

1.7 PREDICTIVE AND DESCRIPTIVE TASKS

In the similar fashion, as the distinction between supervised learning from labelled data and unsupervised learning from unlabelled data, we can draw a distinction between whether the model output involves the target variable or not: we call it a predictive model if it does, and a descriptive model if it does not. This leads to the four different machine learning settings summarised in Table 1.1.

	Predictive model	*Descriptive*
model		
Supervised learning	classification, regression	subgroup
discovery		
Unsupervised learning	predictive clustering	descriptive
clustering,		
association rule discovery		

Table 1.1. An overview of different machine learning settings. (FLACH, 2012)

The rows refer to whether the training data is labelled with a target variable, while the columns indicate whether the models learned are used to predict a target variable or rather describe the given data.

The table 1.1 indicates following points:

- The most common setting is supervised learning of predictive models – in fact, this is what people commonly mean when they refer to supervised learning. Typical tasks are classification and regression.
- It is also possible to use labelled training data to build a descriptive model that is not primarily intended to predict the target variable, but instead identifies, say, subsets of the data that behave differently with respect to the target variable. This example of supervised learning of a descriptive model is called subgroup discovery.
- Descriptive models can naturally be learned in an unsupervised setting, and we have just seen a few examples of that (clustering, association rule discovery and matrix decomposition). This is often the implied setting when people talk about unsupervised learning.
- A typical example of unsupervised learning of a predictive model occurs when we cluster data with the intention of using the clusters to assign class labels to new data. We will call this predictive clustering to distinguish it from the previous, descriptive form of clustering.

SUMMARY

This chapter gives brief introduction of Machine Learning. After studying this chapter, you will learn about definition of machine learning, what is supervised, unsupervised and reinforcement learning, applications of machine learning, how a pattern can be found from data, what are training data and test data and predictive and descriptive tasks with respect to supervised and unsupervised learning.

UNIT END QUESTIONS

1. Define and explain Machine Learning. Also explain its examples in brief.
2. Explain supervised learning and unsupervised learning in detail.
3. Write a short note on learning verses designing.
4. Explain training data and test data in detail.
5. What are the characteristics of machine learning tasks? Explain each one in brief.
6. What are predictive and descriptive tasks? Explain with respect to supervised and unsupervised learning.

REFERENCES

- FLACH, P. (2012). MACHINE LEARNING The Art and Science of Algorithms that Make Sense of Data. Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, S˜ao Paulo, Delhi, Mexico City: cambridge university press.
- jlooper, s. l. (n.d.). *microsoft/ ML-For-Beginners*. Retrieved from https://github.com/microsoft/ML-For-Beginners/blob/main/1-Introduction/1-intro-to-ML/README.md
- *https://www.tutorialspoint.com/*. (n.d.). Retrieved from https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_training_test_data.htm. (2019, Nov 3). https://magnimindacademy.com/blog/7-characteristics-of-machine-learning

*****

2

MACHINE LEARNING MODELS

Unit Structure

2.0 INTRODUCTION

Models form the central concept in machine learning as they are what is being learned from the data, in order to solve a given task. There is a considerable – not to say be wildering – range of machine learning models to choose from. One reason for this is the ubiquity of the tasks that machine learning aims to solve: classification, regression, clustering, association discovery, to name but a few. Examples of each of these tasks can be found in virtually every branch of science and engineering. Mathematicians, engineers, psychologists, computer scientists and many others have discovered – and sometimes rediscovered – ways to solve these tasks. They have all brought their specific background to bear, and consequently the principles underlying these models are also diverse. My personal view is that this diversity is a good thing as it helps to make machine learning the powerful and exciting discipline it is. It doesn't, however, make the task of writing a machine learning book any easier! Luckily, a few common themes can be observed, which allow me to discuss machine learning models in a somewhat more systematic way. We will discuss three groups of models: geometric models, probabilistic models, and logical models. These groupings are not meant to be mutually exclusive, and sometimes a particular kind of model has, for instance, both a geometric and a probabilistic interpretation. Nevertheless, it provides a good starting point for our purposes.

2.1 GEOMETRIC MODELS

The *instance space* is the set of all possible or describable instances, whether they are present in our data set or not. Usually this set has some geometric structure. For instance, if all features are numerical, then we can

use each feature as a coordinate in a Cartesian coordinate system. A *geometric model* is constructed directly in instance space, using geometric concepts such as lines, planes and distances. For instance, the linear classifier depicted in Figure 1 on p.5 is a geometric classifier. 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*.

If there exists a linear decision boundary separating the two classes, we say that the data is linearly separable. As we have seen, a linear decision boundary is defined by the equation $w \cdot x = t$, where w is a vector perpendicular to the decision boundary, x points to an arbitrary point on the decision boundary, and t is the decision threshold. A good way to think of the vector w is as pointing from the 'centre of mass' of the negative examples, n, to the centre of mass of the positives p. In other words, w is proportional (or equal) to p−n. One way to calculate these centres of mass is by averaging. For instance, if P is a set of n positive examples, then we can define $\mathbf{P} = \frac{1}{n} \sum_{x \in P} X$ and similarly for n. By setting the decision threshold appropriately, we can intersect the line from n to p half-way (Figure 2.1).

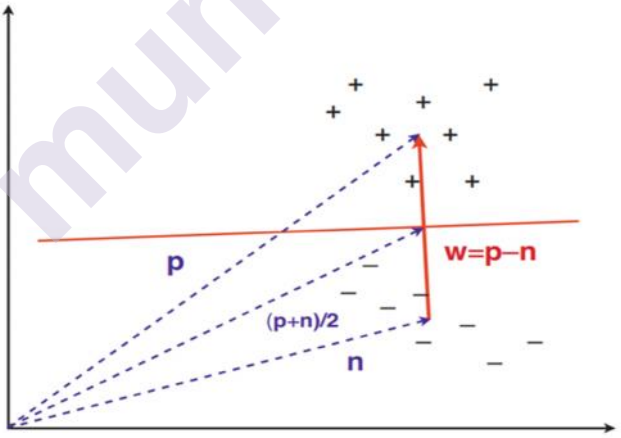


Figure 2.1 The basic linear classifier constructs a decision boundary by half-way intersecting the line between the positive and negative centres of mass. It is described by the equation $\mathbf{w} \cdot \mathbf{x} = t$, with $\mathbf{w} = \mathbf{p} - \mathbf{n}$; the decision threshold can be found by noting that $(\mathbf{p} + \mathbf{n})/2$ is on the decision boundary, and hence $t = (\mathbf{p} - \mathbf{n}) \cdot (\mathbf{p} + \mathbf{n})/2 = (||\mathbf{p}||^2 - ||\mathbf{n}||^2)/2$, where $||\mathbf{x}||$ denotes the length of vector $\mathbf{x}$.

Source: (FLACH, 2012)

We will call this the basic linear classifier. It has the advantage of simplicity, being defined in terms of addition, subtraction and rescaling of

examples only (in other words, w is a linear combination of the examples). However, if those assumptions do not hold, the basic linear classifier can perform poorly – for instance, note that it may not perfectly separate the positives from the negatives, even if the data is linearly separable. Because data is usually noisy, linear separability doesn't occur very often in practice, unless the data is very sparse, as in text classification. Recall that we used a large vocabulary, say 10 000 words, each word corresponding to a Boolean feature indicating whether or not that word occurs in the document. This means that the instance space has 10 000 dimensions, but for any one document no more than a small percentage of the features will be non-zero. As a result there is much 'empty space' between instances, which increases the possibility of linear separability. However, because linearly separable data doesn't uniquely define a decision boundary, we are now faced with a problem: which of the infinitely many decision boundaries should we choose? One natural option is to prefer large margin classifiers, where the margin of a linear classifier is the distance between the decision boundary and the closest instance. Support vector machines are a powerful kind of linear classifier that find a decision boundary whose margin is as large as possible (Figure 2.2).

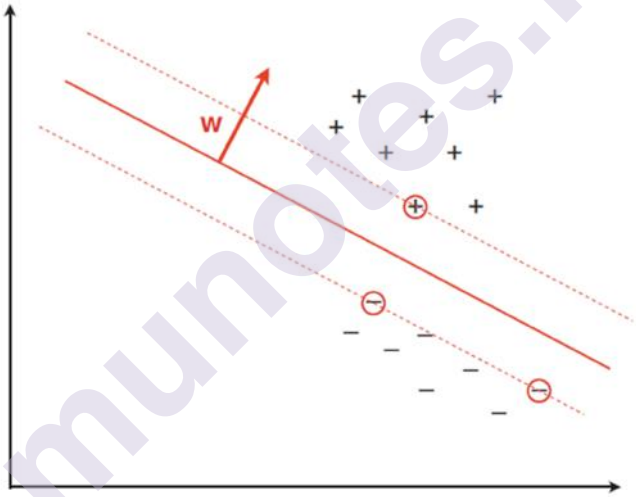


Figure 2.2 The decision boundary learned by a support vector machine from the linearly separable data from Figure 2.1 The decision boundary maximises the margin, which is indicated by the dotted lines. The circled data points are the support vectors.

Source: (FLACH, 2012)

A very useful geometric concept in machine learning is the distance. If the distance between two instances is small then the instances are similar in terms of their feature values, and so nearby instances would be expected to receive the same classification or belong to the same cluster.

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: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. For n points, the general formula is: $\sqrt{\sum_{i=1}^{n}(x - y)2}$

nearest-neighbour classifier:
A very simple distance-based classifier works as follows:
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.

Suppose we want to cluster our data into K clusters, and we have an initial guess of how the data should be clustered. We then calculate the means of each initial cluster and reassign each point to the nearest cluster mean. Unless our initial guess was a lucky one, this will have changed some of the clusters, so we repeat these two steps (calculating the cluster means and reassigning points to clusters) until no change occurs.

It remains to be decided how we construct our initial guess. This is usually done randomly: either by randomly partitioning the data set into K 'clusters' or by randomly guessing K 'cluster centres'. Instead of using Euclidean distance, which May not work exactly the way it should, for outliers, other distances can be used such as Manhattan distance, which sums the distances along each coordinate: $\sum_{i-1}^{n} |x - y|$.

2.2 LOGICAL MODELS

For a given problem, the collection of all possible outcomes represents the sample space or instance space. The basic idea for creating a taxonomy of algorithms is that we divide the instance space by using one of three ways:

- Using a Logical expression.
- Using the Geometry of the instance space.
- Using Probability to classify the instance space.

The outcome of the transformation of the instance space by a machine learning algorithm using the above techniques should be exhaustive (cover all possible outcomes) and mutually exclusive (non-overlapping).

Logical models can also be expressed as Tree models and Rule models

Logical 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.

There are mainly two kinds of logical models: Tree models and Rule models.

Rule models consist of a collection of implications or IF-THEN rules. For tree-based models, the 'if-part' defines a segment and the 'then-part' defines the behaviour of the model for this segment. Rule models follow the same reasoning.

Tree models can be seen as a particular type of rule model where the if-parts of the rules are organised in a tree structure. Both Tree models and Rule models use the same approach to supervised learning. The approach can be summarised in two strategies: we could first find the body of the rule (the concept) that covers a sufficiently homogeneous set of examples and then find a label to represent the body. Alternately, we could approach it from the other direction, i.e., first select a class we want to learn and then find rules that cover examples of the class.

The models of this type can be easily translated into rules that are understandable by humans, such as ·if Bonus = 1 then Class = Y = spam·. Such rules are easily organized in a tree structure, such as the one in Figure 2.3, which is called a feature tree. The idea of such a tree is that features are used to iteratively partition the instance space.

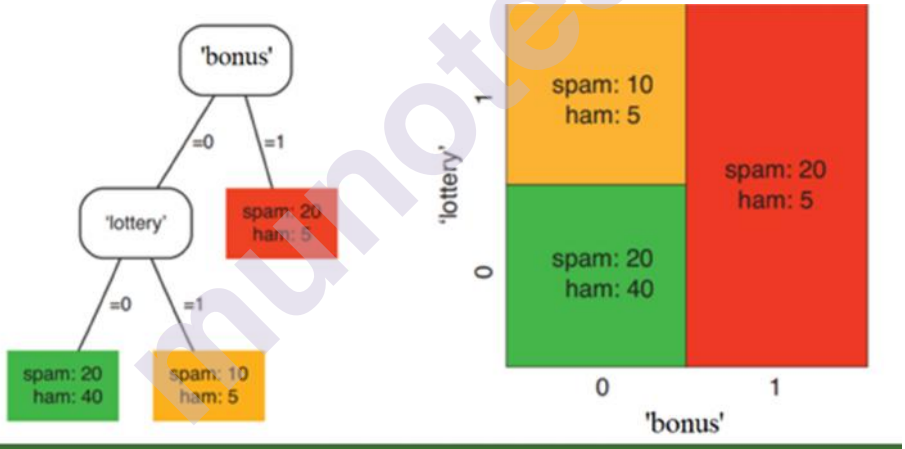


Figure 2.3:(tree learning) (left) A feature tree combining two Boolean features (right) A feature tree partitions the instance space into rectangular regions, one for each leaf

Source: (FLACH, 2012)

The leaves of the tree therefore correspond to rectangular areas in the instance space, which we will call instance space segments, or segments for short. 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 labelled with classes are commonly called *decision trees*. A complete feature tree, which contains all features, one at each level of the tree is shown in figure 2.4.

A feature list is a binary feature tree which always branches in the same direction, either left or right. The tree in Figure 2.3 is a left-branching feature list. Such feature lists can be written as nested if–then–else statements that will be familiar to anyone with a bit of programming experience. For instance, if we were to label the leaves in Figure 2.3 by majority class, we obtain the following decision list as per the Rule learning:

if bonus = 1 then Class = Y = spam
else if lottery = 1 then Class = Y = spam
else Class = Y = ham

Both tree learning and rule learning are implemented in top-down fashion. Select a feature from the instance space, which best splits the entire training sets into different number of subsets. Each subset can then further derive into subsets. Finally, all belongs to each node of a class. In tree learning, we follow divide and conquer approach.

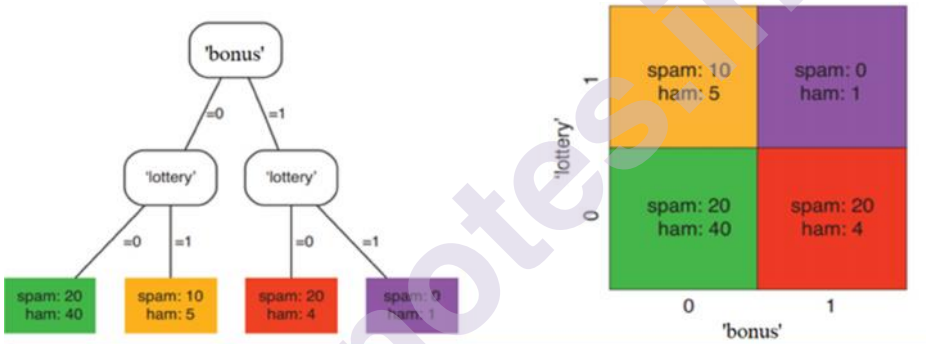


Figure: 2.4
(left) A complete feature tree built from two Boolean features. (right) The corresponding instance space partition is the finest partition that can be achieved with those two features.

Source: (FLACH, 2012)

In rule based, first write a rule, based on some condition and then step by step, we add more conditions to rule by using some set of examples from the training dataset. Now remove those examples from the dataset. Here, we find the class for each feature, ultimately. Here, we follow separate and conquer.

Logical models often have different, equivalent formulations. For instance, two alternative formulations for this model are:

if bonus = 1 ∨ lottery = 1 then Class = Y = spam·
else Class = Y = ham·
if bonus = 0 ∧ lottery = 0 then Class = Y = ham·
else Class = Y = spam·

We can also represent the same model as un-nested rules:

if bonus = 1 then Class = Y = spam·
if bonus = 0 ∧ lottery = 1 then Class = Y = spam·
if bonus = 0 ∧ lottery = 0 then Class = Y = ham.

Here, every path from root to a leaf is translated into a rule. As a result, although rules from the same sub-tree share conditions (such as bonus=0), every pair of rules will have at least some mutually exclusive conditions (such as lottery = 1 in the second rule and lottery = 0 in the third). However, this is not always the case: rules can have a certain overlap. Before learning more on logical models let us understand the terminologies – grouping and grading.

Grouping and grading:
Grouping is breaking the instance space into groups or segments, the number of which is determined at training time. Figure 2.4 shows the example of Grouping.

Grading models are able to distinguish between arbitrary instances, when working in cartesian instance space. The basic linear classifier constructs a decision boundary by half-way intersecting the line between the positive (p) and negative (n) centers of mass. It is described by the equation w· x = t (x is any arbitrary point), as shown in Figure 2.5 – example of Grading.

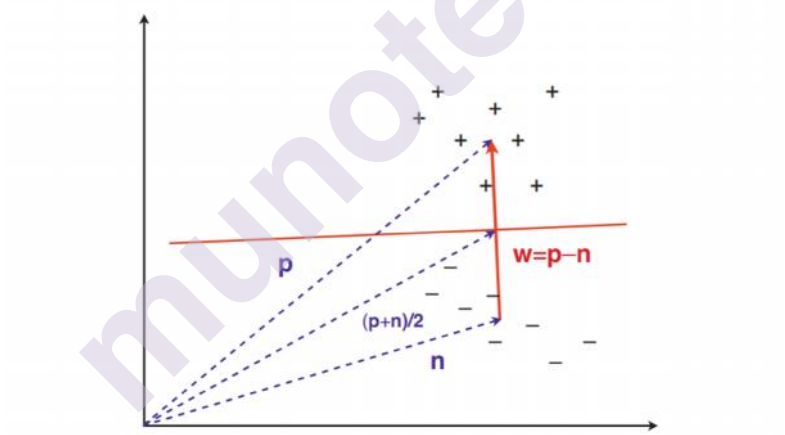


Figure 2.5 – example of Grading Source: (FLACH, 2012)

Let us now continue understanding logical models. An interesting aspect of logical models, which sets them aside from most geometric and probabilistic models, is that they can, to some extent, provide explanations for their predictions.

For example, a prediction assigned by a decision tree could be explained by reading off the conditions that led to the prediction from root to leaf. The model itself can also easily be inspected by humans, which is why they are sometimes called declarative. Declarative models do not need to be restricted to the simple rules that we have considered so far.

The logical rule learning system Progol found the following set of conditions to predict whether a molecular compound is carcinogenic (causes cancer):

1. it tests positive in the Salmonella assay; or
2. it tests positive for sex-linked recessive lethal mutation in Drosophila; or
3. it tests negative for chromosome aberration; or
4. it has a carbon in a six-membered aromatic ring with a partial charge of −0.13; or
5. it has a primary amine group and no secondary or tertiary amines; or
6. it has an aromatic (or resonant) hydrogen with partial charge ≥ 0.168; or
7. it has a hydroxy oxygen with a partial charge ≥ −0.616 and an aromatic (or resonant) hydrogen; or
8. it has a bromine; or
9. it has a tetrahedral carbon with a partial charge ≤ −0.144 and tests positive on Progol's mutagenicity rules.

The first three conditions concerned certain tests that were carried out for all molecules and whose results were recorded in the data as Boolean features. In contrast, the remaining six rules all refer to the structure of the molecule and were constructed entirely by Progol.

For instance, rule 4 predicts that a molecule is carcinogenic if it contains a carbon atom with certain properties. This condition is different from the first three in that it is not a pre-recorded feature in the data, but a new feature that is constructed by Progol during the learning process because it helps to explain the data.

Statisticians work very often with different conditional probabilities, given by the likelihood function P(X|Y). For example, if somebody was to send me a spam e-mail, how likely would it be that it contains exactly the words of the e-mail I'm looking at? And how likely if it were a ham e-mail instead?

With so many words to choose from, the probability of any particular combination of words would be very small indeed. What really matters is not the magnitude of these likelihoods, but their ratio: how much more likely is it to observe this combination of words in a spam e-mail than it is in a non-spam e-mail.

For instance, suppose that for a particular e-mail described by X we have $P(X|Y = \text{spam}) = 3.5 \cdot 10^{-5}$ and $P(X|Y = \text{ham}) = 7.4 \cdot 10^{-6}$, then observing X in a spam e-mail is much more likely than it is in a ham e-mail.

This suggests the following decision rule:

predict spam if the likelihood ratio is larger than 1 and ham otherwise.
So, which one should we use: posterior probabilities or likelihoods? As it turns out, we can easily transform one into the other using Bayes' rule, a simple property of conditional probabilities which states that

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Where, P(Y |X) is conditional probability

P(X |Y) is likelihood function

P(Y) is prior probability without observing data X and

P(X) is probability of features independent of Y

The first decision rule above suggested that we predict the class with maximum posterior probability, which using Bayes' rule can be written in terms of the likelihood function.

2.3 PROBABILISTIC MODELS

The third type of models are probabilistic in nature, like the Bayesian classifier we considered earlier. 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.

Since X is known for a particular instance but Y may not be, we are particularly interested in the conditional probabilities P(Y |X). For instance, Y could indicate whether the e-mail is spam, and X could indicate whether the e-mail contains the words 'bonus' and 'lottery'. The probability of interest is then P(Y | bonus, lottery), with bonus and lottery two Boolean variables which together constitute the feature vector X. For a particular e-mail we know the feature values and so we might write P(Y |bonus = 1,lottery = 0) if the e-mail contains the word 'bonus' but not the word 'lottery'. This is called a posterior probability because it is used after the features X are observed.

Table 2.1 shows an example of how these probabilities might be distributed. From this distribution you can conclude that, if an e-mail doesn't contain the word 'Bonus', then observing the word 'lottery' increases the probability of the e-mail being spam from 0.31 to 0.65; but if the e-mail does contain the word 'Bonus', then observing the word 'lottery' as well decreases the spam probability from 0.80 to 0.40.

Bonus	Lottery	P(Y = spam | Bonus,Lottery)	P(Y = ham | Bonus, Lottery)
0	0	0.31	0.69
0	1	0.65	0.35
1	0	0.80	0.20
1	1	0.40	0.60

Table 2.1. An example posterior distribution. 'Bonus' and 'lottery' are two Boolean features; Y is the class variable, with values 'spam' and 'ham'. In each row the most likely class is indicated in blue. Source: (FLACH, 2012)

Even though this example table is small, it will grow unfeasibly large very quickly, with n Boolean variables 2^n cases have to be distinguished. We therefore don't normally have access to the full joint distribution and have to approximate it using additional assumptions, as we will see below.

Assuming that X and Y are the only variables we know and care about, the posterior distribution P(Y |X) helps us to answer many questions of interest. For instance, to classify a new e-mail we determine whether the words 'Bonus' and 'lottery' occur in it, look up the corresponding probability P(Y = spam | Bonus, Lottery), and predict spam if this probability exceeds 0.5 and ham otherwise. Such a recipe to predict a value of Y on the basis of the values of X and the posterior distribution P(Y |X) is called a decision rule.

2.4 FEATURES

MACHINE LEARNING IS ALL ABOUT using the right features to build the right models that achieve the right tasks – this is the slogan, visualised in Figure 2.6. In essence, *features* define a 'language' in which we describe the relevant objects in our domain. We should not normally have to go back to the domain objects themselves once we have a suitable feature representation, which is why features play such an important role in machine learning.

A *task* is an abstract representation of a problem we want to solve regarding those domain objects: the most common form of these is classifying them into two or more classes. Many of these tasks can be represented as a mapping from data points to outputs.

This mapping or *model* is itself produced as the output of a machine learning algorithm applied to training data; there is a wide variety of models to choose from. No matter what variety of machine learning models you may encounter, you will find that they are designed to solve one of only a small number of tasks and use only a few different types of

features. One could say that *models lend the machine learning field diversity, but tasks and features give it unity*.

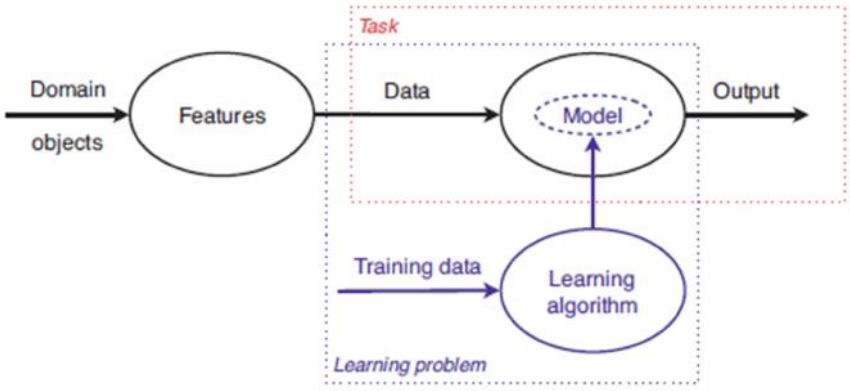


Figure 2.6. An overview of how machine learning is used to address a given task. A task (upper box) requires an appropriate mapping – a model – from data described by features to outputs. Obtaining such a mapping from training data is what constitutes a learning problem (lower box). Source: (FLACH, 2012)

Features determine much of the success of a machine learning application, because a model is only as good as its features. A feature can be thought of as a kind of measurement that can be easily performed on any instance.

Mathematically, they are functions, that map from the instance space to some set of feature values called the domain of the feature. Since measurements are often numerical, the most common feature domain is the set of real numbers. Other typical feature domains include the set of integers, for instance when the feature counts something, such as the number of occurrences of a particular word; the Booleans, if our feature is a statement that can be true or false for a particular instance, such as 'this e-mail is addressed to Beena Kapadia'; and arbitrary finite sets, such as a set of colours, or a set of shapes.

Suppose we have a number of learning models that we want to describe in terms of a number of properties:

- the extent to which the models are geometric, probabilistic or logical
- whether they are grouping or grading models
- the extent to which they can handle discrete and/or real-valued features
- whether they are used in supervised or unsupervised learning; and
- the extent to which they can handle multi-class problems.

The first two properties could be expressed by discrete features with three and two values, respectively; or if the distinctions are more gradual, each aspect could be rated on some numerical scale.

2.5 FEATURE TYPES

There are mainly three kinds of features – Quantitative, Ordinal and Categorical.

Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	√	×	median	quantiles	n/a
Quantitative	√	√	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Table 2.1. Kinds of features, their properties and allowable statistics. Each kind inherits the statistics from the kinds above it in the table. For instance, the mode is a statistic of central tendency that can be computed for any kind of feature. Source: (FLACH, 2012)

Quantitative:
They have a meaningful numerical scale and order. They most often involve a mapping into the reals or continuous. Even if a feature maps into a subset of the reals, such as age expressed in years, the various statistics such as mean or standard deviation still require the full scale of the reals.

Ordinal:
Features with an ordering but without scale are called ordinal features. The domain of an ordinal feature is some totally ordered set, such as the set of characters or strings. Even if the domain of a feature is the set of integers, denoting the feature as ordinal means that we have to dispense with the scale, as we did with house numbers. Another common example are features that express a rank order: first, second, third, and so on. Ordinal features allow the mode and median as central tendency statistics, and quantiles as dispersion statistics.

Categorical:
Features without ordering or scale are called categorical features (or sometimes 'nominal' features). They do not allow any statistical summary except the mode. One subspecies of the categorical features is the Boolean feature, which maps into the truth values true and false. The situation is summarised in Table 2.1.

Models treat these different kinds of feature in distinct ways. First, consider tree models such as decision trees. A split on a categorical feature will have as many children as there are feature values. Ordinal and quantitative features, on the other hand, give rise to a binary split, by selecting a value v0 such that all instances with a feature value less than or equal to v0 go to one child, and the remaining instances to the other child.
It follows that tree models are insensitive to the scale of quantitative features. For example, whether a temperature feature is measured on the Celsius scale or on the Fahrenheit scale will not affect the learned tree. Neither will switching from a linear scale to a logarithmic scale have any effect: the split threshold will simply be logv0 instead of v0. In general, tree models are insensitive to monotonic transformations on the scale of a feature, which are those transformations that do not affect the relative order of the feature values. In effect, tree models ignore the scale of

quantitative features, treating them as ordinal. The same holds for rule models.

Now let's consider the naive Bayes classifier. We have seen that this model works by estimating a likelihood function P(X|Y) for each feature X given the class Y. For categorical and ordinal features with k values this involves estimating P(X = v1|Y), . . . ,P(X = vk |Y). In effect, ordinal features are treated as categorical ones, ignoring the order.

Quantitative features cannot be handled at all, unless they are discretised into a finite number of bins and thus converted to categorical. Alternatively, we could assume a parametric form for P(X|Y), for instance a normal distribution.

While naive Bayes only really handles categorical features, many geometric models go in the other direction: they can only handle quantitative features. Linear models are a case in point: the very notion of linearity assumes a Euclidean instance space in which features act as Cartesian coordinates, and thus need to be quantitative. Distance-based models such as k-nearest neighbour and K-means require quantitative features if their distance metric is Euclidean distance, but we can adapt the distance metric to incorporate categorical features by setting the distance to 0 for equal values and 1 for unequal values.

In a similar vein, for ordinal features we can count the number of values between two feature values (if we encode the ordinal feature by means of integers, this would simply be their difference). This means that distance-based methods can accommodate all feature types by using an appropriate distance metric. Similar techniques can be used to extend support vector machines and other kernel-based methods to categorical and ordinal features.

2.6 FEATURE CONSTRUCTION AND TRANSFORMATION

There is a lot of scope in machine learning for playing around with features. In the spam filter example, and text classification more generally, the messages or documents don't come with built-in features; rather, they need to be constructed by the developer of the machine learning application. This feature construction process is absolutely crucial for the success of a machine learning application.

Indexing an e-mail by the words that occur in it (called a bag of words representation as it disregards the order of the words in the e-mail) is a carefully engineered representation that manages to amplify the 'signal' and attenuate the 'noise' in spam e-mail filtering and related classification tasks. However, it is easy to conceive of problems where this would be exactly the wrong thing to do: for instance if we aim to train a classifier to

distinguish between grammatical and ungrammatical sentences, word order is clearly signal rather than noise, and a different representation is called for.

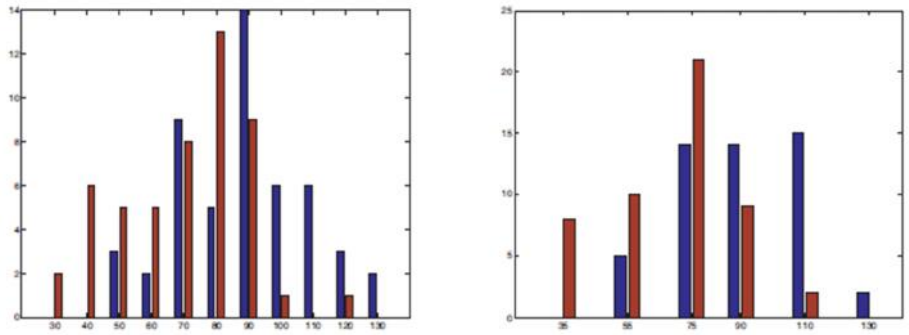


Figure 2.7. (left) Artificial data depicting a histogram of body weight measurements of people with (blue) and without (red) diabetes, with eleven fixed intervals of 10 kilograms width each. (right) By joining the first and second, third and fourth, fifth and sixth, and the eighth, ninth and tenth intervals, we obtain a discretisation such that the proportion of diabetes cases increases from left to right. This discretisation makes the feature more useful in predicting diabetes. (FLACH, 2012)

It is often natural to build a model in terms of the given features. However, we are free to change the features as we see fit, or even to introduce new features. For instance, real-valued features often contain unnecessary detail that can be removed by discretisation. Imagine you want to analyse the body weight of a relatively small group of, say, 100 people, by drawing a histogram.

If you measure everybody's weight in kilograms with one position after the decimal point (i.e., your precision is 100 grams), then your histogram will be sparse and spiky. It is hard to draw any general conclusions from such a histogram. It would be much more useful to discretise the body weight measurements into intervals of 10 kilograms. If we are in a classification context, say we're trying to relate body weight to diabetes, we could then associate each bar of the histogram with the proportion of people having diabetes among the people whose weight falls in that interval. In fact, we can even choose the intervals such that this proportion is monotonically increasing as shown in Figure 2.7.

The previous example gives another illustration of how, for a particular task such as classification, we can improve the signal-to-noise ratio of a feature. In more extreme cases of feature construction, we transform the entire instance space. Consider Figure 2.6: the data on the left is clearly not linearly separable, but by mapping the instance space into a new 'feature space' consisting of the squares of the original features we see that the data becomes almost linearly separable. In fact, by adding in a third feature we can perform a remarkable trick: we can build this feature space classifier without actually constructing the feature space.

2.7 FEATURE SELECTION

Once we have constructed new features it is often a good idea to select a suitable subset of them prior to learning. Not only will this speed up learning as fewer candidate features need to be considered, it also helps to guard against overfitting.

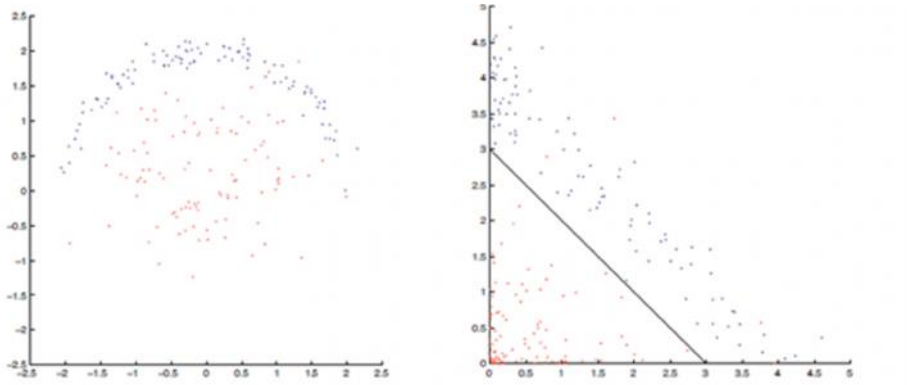


Figure 2.9 (left) A linear classifier would perform poorly on this data. (right) By transforming the original (x, y) data into $(x', y') = (x^2, y^2)$, the data becomes more 'linear', and a linear decision boundary $x' + y' = 3$ separates the data fairly well. In the original space this corresponds to a circle with radius $\sqrt{3}$ around the origin.
(FLACH, 2012)

There are two main approaches to feature selection, The filter approach and the relief approach.

The filter approach scores the features on a particular metric and the top-scoring features are selected. Many of the metrics we have seen so far can be used for feature scoring, including information gain, the $\chi 2$ statistic, the correlation coefficient, to name just a few.

An interesting variation is provided by the Relief feature selection method, which repeatedly samples a random instance x and finds its nearest hit h (instance of the same class) as well as its nearest miss m (instance of opposite class). The i -th feature's score is then decreased by $Dis(xi , hi)^2$ and increased by $Dis(xi , mi)^2$, where Dis is some distance measure (e.g., Euclidean distance for quantitative features, Hamming distance for categorical features). The intuition is that we want to move closer to the nearest hit while differentiating from the nearest miss.

One drawback of a simple filter approach is that no account is taken of redundancy between features. Imagine, for the sake of the argument, duplicating a promising feature in the data set: both copies score equally high and will be selected, whereas the second one provides no added value in the context of the first one.
Secondly, feature filters do not detect dependencies between features as they are solely based on marginal distributions. For example, consider two Boolean features such that half the positives have the value true for both features and the other half have the value false for both, whereas all negatives have opposite values (again distributed half-half over the two

possibilities). It follows that each feature in isolation has zero information gain and hence is unlikely to be selected by a feature filter, despite their combination being a perfect classifier. One could say that feature filters are good at picking out possible root features for a decision tree, but not necessarily good at selecting features that are useful further down the tree.

To detect features that are useful in the context of other features, we need to evaluate sets of features; this usually goes under the name of wrapper approaches. The idea is that feature selection is 'wrapped' in a search procedure that usually involves training and evaluating a model with a candidate set of features.

Forward selection methods start with an empty set of features and add features to the set one at a time, as long as they improve the performance of the model. Backward elimination starts with the full set of features and aims at improving performance by removing features one at a time. Since there are an exponential number of subsets of features it is usually not feasible to search all possible subsets, and most approaches apply a 'greedy' search algorithm that never reconsiders the choices it makes.

SUMMARY

After studying this chapter, you will understand different modes like Geometric Models, Logical Models and Probabilistic Models. You will understand about features usage and why it is very important in model designing. You will also understand about different Feature types, how they can be Constructed and why their Transformation required and how it can be done. You will also understand how Feature Selection plays an important role in designing a model and how to do it.

UNIT END QUESTIONS

1. How a linear classifier construct decision boundary using linear separable data? Explain it in detail with respect to geometric models of Machine Learning.
2. Explain the working of decision boundary learned by Support Vector Machine from linear separable data with respect to geometric models of Machine Learning.
3. Describe logical models.
4. Write a short note on probabilistic models.
5. Machine learning is all about using the right features to build the right models that achieve the right tasks – justify this sentence.
6. What are various types of features available? Explain each one in brief.

7. Why are feature construction and feature transformation required? How to achieve them?
8. What are the approaches to feature selection? Explain each one in detail.

REFERENCES

- FLACH, P. (2012). MACHINE LEARNING The Art and Science of Algorithms that Make Sense of Data. Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, S˜ao Paulo, Delhi, Mexico City: cambridge university press.

*****

UNIT II

3

CLASSIFICATION AND REGRESSION

Unit structure

3.0 OBJECTIVES

A learner will learn:

- Machine learning methods like classification
- Will also explore the classification algorithms with practical approach.
- Concept of multiclass classification

3.1 INTRODUCTION

Classification may be defined as the process of predicting class or category from observed values or given data points. The categorized output can have the form such as “Red” or “Blue” or “spam” or “no spam”.

Conceptually, classification is the task of approximating a mapping function (f) from input variables (X) that tends to output variables (Y). It is basically belonging to the supervised machine learning in which targets are also provided along with the input data set.

An example of classification problem can be the spam detection in emails. There can be only two categories of output, “spam” and “no spam”; hence this is a binary type classification.

To implement this classification, we first need to train the classifier. For this example, “spam” and “no spam” emails would be used as the training data. After successfully train the classifier, it can be used to detect an unknown email.

3.1.1Types of Learners in Classification:

There exist two types of learners in classification problems –

a. Lazy Learners:

These learners wait for the testing data to be appeared after storing the training data. Classification is done only after getting the testing data. They spend less time on training but more time on predicting. Examples of lazy learners are K-nearest neighbor and case-based reasoning.

b. Eager Learners

As opposite to lazy learners, eager learners construct classification model without waiting for the testing data to be appeared after storing the training data. They spend more time on training but less time on predicting. Examples of eager learners are Decision Trees, Naïve Bayes and Artificial Neural Networks (ANN).’

3.2 CLASSIFICATION

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. Classification is the most common task in machine learning. A classifier is $\hat{c}: X \rightarrow C$, where $C = \{C1,C2,...,Ck\}$ is a finite and usually small set of class labels. We will sometimes also use C_i to indicate the set of examples of that class. We use the ‘hat’ to indicate that $\hat{c}(x)$ is an estimate of the true but unknown function $c(x)$. Examples for a classifier take the form $(x,c(x))$, where $x \in X$ is an instance and $c(x)$ is the true class of the instance. Learning a classifier involves constructing the function $\hat{c}$ such that it matches c as closely as possible (and not just on the training set, but ideally on the entire instance space X).

Classification in machine learning and statistics is a supervised learning approach in which the computer program learns from the data given to it and make new observations or classifications. Classification is a process of categorizing a given set of data into classes, It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories. The classification predictive modelling is the task of approximating the mapping function from input variables to discrete output variables. The main goal is to identify which class/category the new data will fall into.

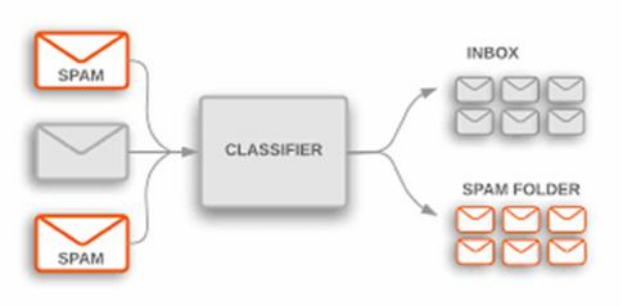


Fig 3.1: Email classification example

The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier. **Examples:** YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier. **Example:** Classifications of types of crops, Classification of types of music.

3.3 ASSESSING CLASSIFICATION PERFORMANCE

The contingency table is used to assess the performance of classification. The table is also known as confusion matrix. The table is constituted with rows and columns. Each row refers to actual classes as recorded in the test set, and each column to classes as predicted by the classifier. In the given table 3.1 , the first row states that the test set contains 50 positives, 30 of which were correctly predicted and 20 incorrectly. The last column and the last row give the marginals (i.e., column and row sums). Marginals are important because they allow us to assess statistical significance.

	Predicted (+ve)	Predicted (-ve)	
Actual (+ve)	30	20	50
Actual (-ve)	10	40	50
	40	60	100

Table 3.1: Confusion Matrix

	+ve	-ve	
+ve	20	30	50
-ve	20	30	50
	40	60	100

Table 3.2: two-class contingency table

The table 3.2, has the same marginals, but the classifier clearly makes a random choice as to which predictions are positive and which are negative

– as a result the distribution of actual positives and negatives in either predicted class is the same.

From a contingency table we can calculate a range of performance indicators. The simplest of these is accuracy, which is the proportion of correctly classified test instances. In the notation introduced at the beginning of this chapter, accuracy over a test set Te is defined as shown in equation 3.1:

$$acc = \frac{1}{|Te|} \sum_{x \in Te} I[\hat{c}(x) = c(x)]$$

…….(3.1)

As stated in the equation 3.1, the function I[·] denotes the indicator function, which is 1 if its argument evaluates to true, and 0 otherwise. In this case it is a convenient way to count the number of test instances that are classified correctly by the classifier (i.e., the estimated class label cˆ(x) is equal to the true class label c(x)). Alternatively, we can calculate the error rate as the proportion of incorrectly classified instances, here 0.30 and 0.50, respectively. Clearly, accuracy and error rate sum to 1.

Test set accuracy can be seen as an estimate of the probability that an arbitrary instance x ∈ X is classified correctly: more precisely, it estimates the probability.

$$P_{\mathcal{X}}(\hat{c}(x) = c(x))$$

We have access to the true classes of a small fraction of the instance space and so an estimate is all we can hope to get. It is therefore important that the test set is as representative as possible. This is usually formalised by the assumption that the occurrence of instances in the world. Correctly classified positives and negatives are referred to as true positives and true negatives, respectively. Incorrectly classified positives are, perhaps somewhat confusingly, called false negatives; similarly, misclassified negatives are called false positives. The positive/negative refers to the classifier's prediction, and true/false refers to whether the prediction is correct or not. So, a false positive is something that was incorrectly predicted as positive, and therefore an actual negative (e.g., a ham email misclassified as spam, or a healthy patient misclassified as having the disease in question).

The true positive rate is the proportion of positives correctly classified, and can be defined mathematically as given in equation 3.2:

$$tpr = \frac{\sum_{x \in Te} I[\hat{c}(x) = c(x) = \oplus]}{\sum_{x \in Te} I[c(x) = \oplus]}$$

…….(3.2)

True positive rate is an estimate of the probability that an arbitrary positive is classified correctly, that is, an estimate of P_X (cˆ(x) = +velc(x) = +ve). Analogously, the true negative rate is the proportion of negatives correctly classified and estimates P_X (cˆ(x) = -velc(x) = -ve). These rates, which are sometimes called sensitivity and specificity, can be seen as per-class accuracies. In the contingency table, the true positive and negative rates can be calculated by dividing the number on the descending (good) diagonal by the row total. In table 3.2, we have a true positive rate of 60%, a true negative rate of 80%, a false negative rate of 40% and a false positive rate of 20%. We have a true positive rate of 40%, a true negative rate of 60%, a false negative rate of 60% and a false positive rate of 40%. Notice that the accuracy in both cases is the average of the true positive rate and the true negative rate .

Example 2.1 (Accuracy as a weighted average). Suppose a classifier's predictions on a test set are as in the following table:

	Predicted (+ve)	Predicted (-ve)	
Actual (+ve)	60	15	75
Actual (-ve)	10	15	25
	70	30	100

From this table, we see that the true positive rate is tpr = 60/75 = 0.80 and the true negative rate is tnr = 15/25 = 0.60. The overall accuracy is acc = (60 + 15)/100 = 0.75, which is no longer the average of true positive and negative rates. However, taking into account the proportion of positives pos = 0.75 and the proportion of negatives neg = 1−pos = 0.25, we see that acc = pos·tpr +neg ·tnr

This equation holds in general: if the numbers of positives and negatives are equal, we obtain the unweighted average from the earlier example (acc = (tpr + tnr)/2). The given equation has a neat intuition: good performance on either class contributes to good classification accuracy, but the more prevalent class contributes more strongly. In order to achieve good accuracy, a classifier should concentrate on the majority class, particularly if the class distribution is highly unbalanced. However, it is often the case that the majority class is also the least interesting class.

3.4 CLASS PROBABILITY ESTIMATION

A class probability estimator – or probability estimator in short – is a scoring classifier that outputs probability vectors over classes, i.e., a mapping pˆ : X → [0,1]k . We write pˆ (x) = pˆ1(x),...,pˆk (x) , where pˆi(x) is the probability assigned to class Ci for instance x, and k i=1 pˆi(x) = 1. If we have only two classes, the probability associated with one class is 1 minus the probability of the other class; in that case, we use pˆ(x) to denote the estimated probability of the positive class for instance x. As

with scoring classifiers, we usually do not have direct access to the true probabilities pi(x).

First, assume a situation in which any two instances are similar to each other. We then have PC (c(x) = ⊕|x ∼ x) = PC (c(x) = ⊕) which is simply estimated by the proportion pos of positives in our data set (I am going to drop the subscript C from now on). In other words, in this scenario we predict p̂(x) = pos regardless

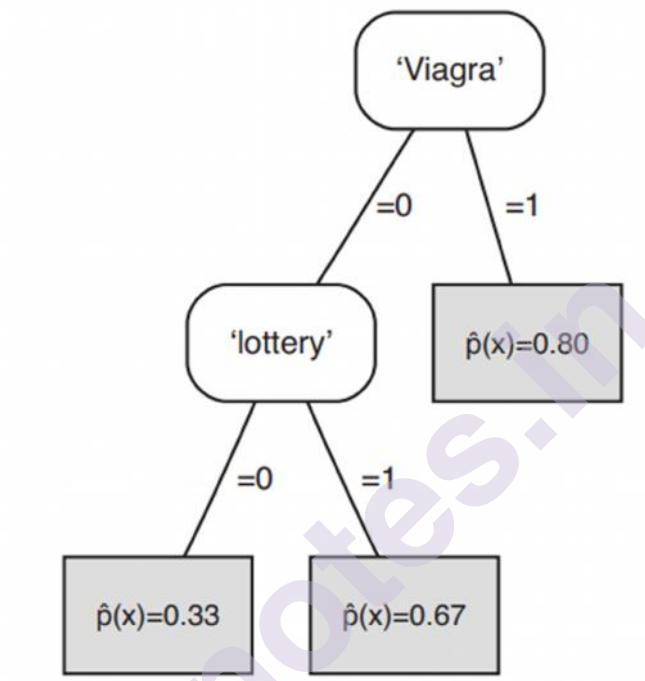


Figure 3.2. A probability estimation tree

of whether we know anything about x's true class. At the other extreme, consider a situation in which no two instances are similar unless they are the same, i.e., x ∼ x if x = x, and x ∼ x otherwise. In this case we have P(c(x) = +velx∼ x) = P(c(x) = +ve), which – because x is fixed – is 1 if c(x) = +ve and 0 otherwise. Put differently, we predict p̂(x) = 1 for all known positives and p̂(x) = 0 for all known negatives, but we can't generalise this to unseen instances.

A feature tree allows us to strike a balance between these extreme and simplistic scenarios, using the similarity relation ∼T associated with feature tree T : x ∼T x if, and only if, x and x are assigned to the same leaf of the tree. In each leaf we then predict the proportion of positives assigned to that leaf.

3.5 ASSESSING CLASS PROBABILITY ESTIMATES

As with classifiers, we can now ask the question of how good these class probability estimators are. A slight complication here is that, as already

remarked, we do not have access to the true probabilities. One trick that is often applied is to define a binary vector (I[c(x) = C1],...,I[c(x) = Ck]), which has the i-th bit set to 1 if x's true class is Ci and all other bits set to 0, and use these as the 'true' probabilities. We can then define the squared error (SE) of the predicted probability vector pˆ (x) = (pˆ1(x),...,pˆk (x)) as :

$$\mathrm{SE}(x) = \frac{1}{2}\sum_{i=1}^{k}(\hat{p}_i(x) - I[c(x) = C_i])^2 \qquad \ldots\ldots. (3.3)$$

and the mean squared error (MSE) as the average squared error over all instances in the test set:

$$\mathrm{MSE}(Te) = \frac{1}{|Te|}\sum_{x \in Te}\mathrm{SE}(x) \qquad \ldots\ldots\ldots (3.4)$$

This definition of error in probability estimates is often used in forecasting theory where it is called the Brier score.

Example 2.6 (Squared error).:
Suppose one model predicts (0.70,0.10,0.20) for a particular example x in a three-class task, while another appears much more certain by predicting (0.99, 0, 0.01). If the first class is the actual class, the second prediction is clearly better than the first: the SE of the first prediction is ((0.70 − 1)2+(0.10−0)2+(0.20−0)2)/2 = 0.07, while for the second prediction it is ((0.99− 1)2+(0−0)2+(0.01−0)2)/2 = 0.0001. The first model gets punished more because, although mostly right, it isn't quite sure of it. However, if the third class is the actual class, the situation is reversed: now the SE of the first prediction is ((0.70−0)2 +(0.10−0)2 +(0.20−1)2)/2 = 0.57, and of the second ((0.99−0)2 +(0−0)2 +(0.01−1)2)/2 = 0.98. The second model gets punished more for not just being wrong, but being presumptuous.

Wit reference to figure 3.2, we calculate the squared error per leaf as follows (left to right): SE1 = 20(0.33−1)2 +40(0.33−0)2 = 13.33 SE2 = 10(0.67−1)2 +5(0.67−0)2 = 3.33 SE3 = 20(0.80−1)2 +5(0.80−0)2 = 4.00. which leads to a mean squared error of MSE = 1 100 (SE1+SE2+SE3) = 0.21

An interesting question is whether we can change the predicted probabilities in each leaf to obtain a lower mean squared error. It turns out that this is not possible: predicting probabilities obtained from the class distributions in each leaf is optimal in the sense of lowest MSE.

For instance, changing the predicted probabilities in the left-most leaf to 0.40 for spam and 0.60 for ham, or 0.20 for spam and 0.80 for ham, results in a higher squared error: SE 1 = 20(0.40−1)2 +40(0.40−0)2 = 13.6 SE 1 = 20(0.20−1)2 +40(0.20−0)2 = 14.4 The reason for this becomes obvious if we rewrite the expression for two-class squared error of a leaf as follows,

using the notation n⊕ and nfor the numbers of positive and negative examples in the leaf:

$$n^{\oplus}(\hat{p}-1)^2 + n^{\ominus}\hat{p}^2 = (n^{\oplus}+n^{\ominus})\hat{p}^2 - 2n^{\oplus}\hat{p} + n^{\oplus} = (n^{\oplus}+n^{\ominus})\left[\hat{p}^2 - 2\dot{p}\hat{p} + \dot{p}\right]$$
$$= (n^{\oplus}+n^{\ominus})\left[(\hat{p}-\dot{p})^2 + \dot{p}(1-\dot{p})\right]$$

where $p^{\cdot} = n\oplus/(n\oplus + n)$ is the relative frequency of the positive class among the examples covered by the leaf, also called the empirical probability. As the term $p^{\cdot}(1-p^{\cdot})$ does not depend on the predicted probability $p\hat{}$, we see immediately that we achieve lowest squared error in the leaf if we assign $p\hat{} = p^{\cdot}$.

3.6 MULTI-CLASS PROBABILITY ESTIMATION

Consider the standard setting of multi-class classification with an instance space X and a set of classes Y = {y1, . . . , yK}. We are interested in learning a probabilistic classifier, that is, a model that estimates the conditional probabilities of classes given an instance x ∈ X:
(p1, . . . , pK) = (Py (y1 | x), . . . , Py (yK | x)) (3.4)

Since true probability degrees are rarely available for training, probabilistic classifiers are typically trained on standard classification data, that is, observations of the form (x, y) ∈ X × Y, where the class label y is assumed to be generated according to Py (· | x).

Probability estimation is known to be a quite hard problem, especially in comparison to standard classification. This comes at no surprise, noting that proper probability estimation is a sufficient but not necessary condition for proper classification: If the conditional class probabilities (1) are predicted accurately, an optimal classification can simply be made by picking the class with highest probability:

$$\hat{y} = \arg\max_{y_i \in \mathcal{Y}} \mathbf{P}(y_i \mid \boldsymbol{x}) \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.5)$$

The Bayes decision can be taken so as to minimize any loss in expectation. On the other hand, a correct classification can also be obtained based on less accurate probability estimates. In fact, the classification will remain correct as long as the estimated probability is highest for the true class. Or, stated differently, an estimation error will remain ineffective unless it changes the result of the arg max operation.

Methods like naive Bayes and decision trees are multi-class classifiers and can in principle be used to produce probability estimates in this setting. In practice, however, one often prefers to estimate probabilities in the two-class setting, especially because estimating a single probability (of the positive class) is much simpler than estimating K − 1 probabilities simultaneously. Moreover, the binary case is amenable to a broader

spectrum of classifiers, including logistic regression, which is a proven method for probability estimation. On the other hand, the reduction of multinomial to binomial probability estimation obviously involves an aggregation problem, namely the need to combine probabilities on pairs of classes into probabilities on the label set Y. This is the idea of "pairwise coupling" techniques.

SUMMARY

Classification deals with labelling the tuples base on some attribute. Binary classification refers to predicting one of two classes and multi-class classification involves predicting one of more than two classes. Multi-label classification involves predicting one or more classes for each example and imbalanced classification refers to classification tasks where the distribution of examples across the classes is not equal. Class probability Estimation. A class probability estimator – or probability estimator is a scoring classifier. Multi-class classification makes the assumption that each sample is assigned to one and only one label.

UNIT END QUESTIONS

1. Explain the concept of classification with suitable example.
2. Illustrate the assessment of classification with suitable example.
3. Write a note on binary classification
4. Briefly explain the concept of class probability Estimation
5. Explain Multiclass Classification with concept note.
6. How are classification estimates assessed? Explain with suitable example.

REFERENCES

- Peter Flach, Machine Learning The Art and Science of Algorithms that Make Sense of Data, Cambridge Press, 2012
- Baidaa M Alsafy, Zahoor Mosad, Wamidh k. Mutlag, Multiclass Classification Methods: A Review, International Journal of Advanced Engineering Technology and Innovative Science (IJAETIS), 2020.
- **Robust Model-Free Multiclass Probability Estimation (nih.gov)**
- **Probability Estimation - an overview | ScienceDirect Topics**

*****

4

REGRESSION

Unit structure

4.0 OBJECTIVES

A learning will learn:

- regression method with practical cases
- how to apply regression method
- identify the error measures with overfitting values

4.1 INTRODUCTION

Regression is a method of modelling a target value based on independent predictors. This method is mostly used for forecasting and finding out cause and effect relationship between variables. Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables. 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 variables.

As mentioned earlier Linear Regression is a supervised machine learning algorithm. It predicts the value within a continuous range of numbers.

1. Simple regression:

Simple linear regression uses traditional slope-intercept form to produce the most accurate predictions. x represents our input data and y represents our prediction.

The motive of the linear regression algorithm is to find the best values for m and c in the equation y = mx + c.

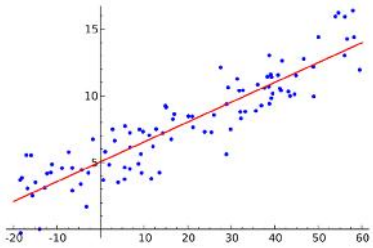


Fig. 4.1: Simple Linear Regression

Simple linear regression is a type of regression analysis where the number of independent variables is one and there is a linear relationship between the independent(x) and dependent(y) variable. The red line in the above graph is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best. The line can be modelled based on the linear equation shown below.

y = a_0 + a_1 * x # Linear Equation

The motive of the linear regression algorithm is to find the best values for a_0 and a_1. Before moving on to the algorithm, let's have a look at two important concepts you must know to better understand linear regression.

4.1.1 Cost Function:

The cost function helps us to figure out the best possible values for a_0 and a_1 which would provide the best fit line for the data points. Since we want the best values for a_0 and a_1, we convert this search problem into a minimization problem where we would like to minimize the error between the predicted value and the actual value

$$minimize \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

We choose the above function to minimize. The difference between the predicted values and ground truth measures the error difference. We square the error difference and sum over all data points and divide that value by the total number of data points. This provides the average squared error over all the data points. Therefore, this cost function is also known as the Mean Squared Error(MSE) function. Now, using this MSE

function we are going to change the values of a_0 and a_1 such that the MSE value settles at the minima. Cost function optimizes the regression coefficients or weights. It measures how 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.

4.1.2 Gradient Descent:

The next important concept needed to understand linear regression is gradient descent. Gradient descent is a method of updating a_0 and a_1 to reduce the cost function(MSE). The idea is that we start with some values for a_0 and a_1 and then we change these values iteratively to reduce the cost. Gradient descent helps us on how to change the values. Gradient descent is used to minimize the MSE by calculating the gradient of the cost function. A regression model uses gradient descent to update the coefficients of the line by reducing the cost function. It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.

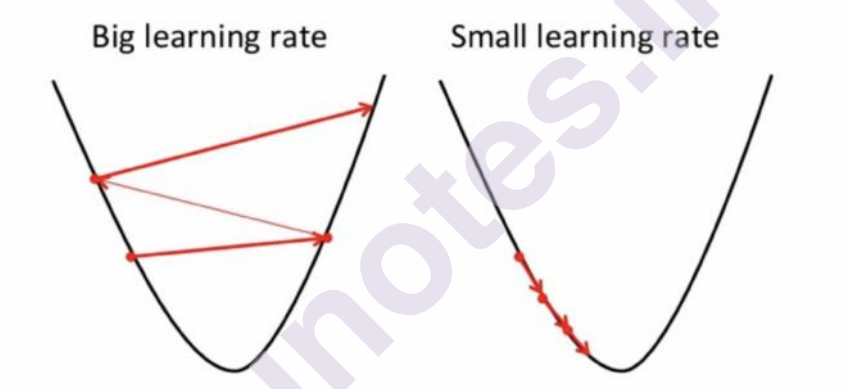


Fig 4.2: Gradient Descent

To draw an analogy, imagine a pit in the shape of U and you are standing at the topmost point in the pit and your objective is to reach the bottom of the pit. There is a catch, you can only take a discrete number of steps to reach the bottom. If you decide to take one step at a time you would eventually reach the bottom of the pit but this would take a longer time. If you choose to take longer steps each time, you would reach sooner but, there is a chance that you could overshoot the bottom of the pit and not exactly at the bottom. In the gradient descent algorithm, the number of steps you take is the learning rate. This decides on how fast the algorithm converges to the minima.

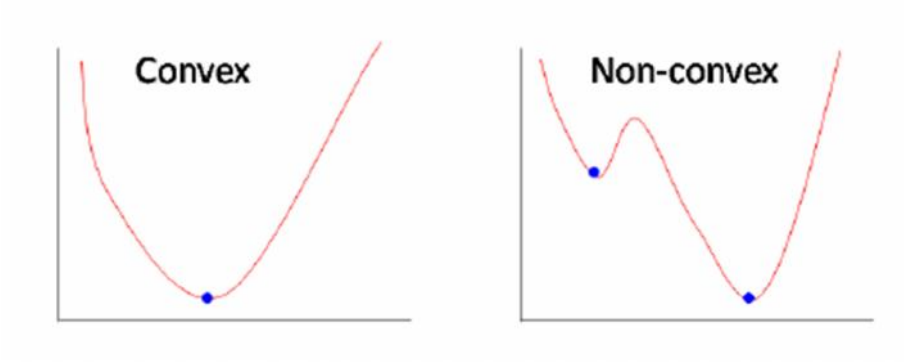


Fig 4.3: Convex and Non Convex

4.2 ASSESSING PERFORMANCE OF REGRESSION

In regression model, the most commonly known evaluation metrics include:

1. **Mean Absolute Error** (MAE), like the RMSE, the MAE measures the prediction error. Mathematically, it is the average absolute difference between observed and predicted outcomes, MAE = mean(abs(observeds - predicteds)). MAE is less sensitive to outliers compared to RMSE.
2. **Root Mean Squared Error** (RMSE), which measures the average error performed by the model in predicting the outcome for an observation. Mathematically, the RMSE is the square root of the *mean squared error (MSE)*, which is the average squared difference between the observed actual outome values and the values predicted by the model. So, MSE = mean((observeds - predicteds)^2) and RMSE = sqrt(MSE). The lower the RMSE, the better the model.
3. **R-squared** (R2), which is the proportion of variation in the outcome that is explained by the predictor variables. In multiple regression models, R2 corresponds to the squared correlation between the observed outcome values and the predicted values by the model. The Higher the R-squared, the better the model.
4. **Residual Standard Error** (RSE), also known as the *model sigma*, is a variant of the RMSE adjusted for the number of predictors in the model. The lower the RSE, the better the model. In practice, the difference between RMSE and RSE is very small, particularly for large multivariate data.

4.2.1 Mean Absolute Error (MAE):

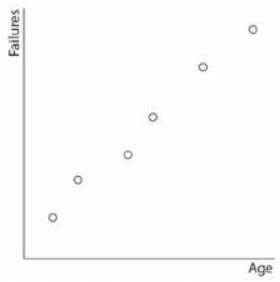
It is the average absolute difference between observed and predicted outcomes.

We defined the MAE as,

$$MAE = \frac{\Sigma|y - \hat{y}|}{N}$$

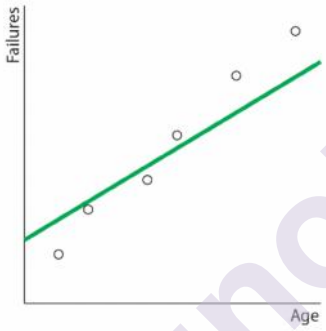
where y_v is the actual value $\hat{y}$ is the predicted value and $|y-y^{\wedge}|$ is the absolute value of the difference between the actual and predicted value. N is the number of sample points.

For Example: Take a look at the following plot, which shows the number of failures for a piece of machinery against the age of the machine:



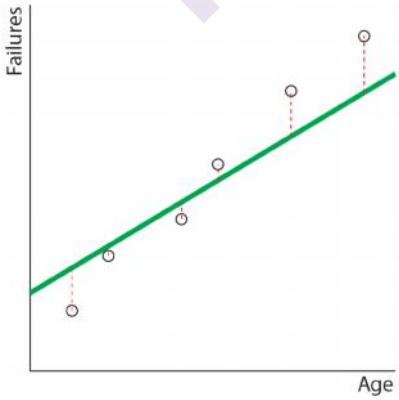
Age	Failures
10	15
20	30
40	40
50	55
70	75
90	90

In order to predict the number of failures from the age, we would want to fit a regression line such as this:



Age	Failures	Prediction
10	15	26
20	30	32
40	40	44
50	55	50
70	75	62
90	90	74

In order to understand how well this line represents the actual data, we need to measure *how good a fit it is*. We can do this by measuring the distance from the actual data points to the line:



Age	Failures	Prediction	Error
10	15	26	11
20	30	32	2
40	40	44	4
50	55	50	-5
70	75	62	-13
90	90	74	-16

You may recall that these distances are called residuals or errors. The mean size of these errors is the MAE. We can calculate it as follows:

Age	Failures	Prediction	Error	abs(Error)
10	15	26	11	11
20	30	32	2	2
40	40	44	4	4
50	55	50	-5	5
70	75	62	-13	13
90	90	74	-16	16
		Mean abs(Error)		8.5

here is how the table and formula relate:

	y	$\hat{y}$	$y-\hat{y}$	$\lvert y-\hat{y}\rvert$
Age	Failures	Prediction	Error	abs(Error)
10	15	26	11	11
20	30	32	2	2
40	40	44	4	4
50	55	50	-5	5
70	75	62	-13	13
90	90	74	-16	16
	Mean abs(Error)		$\frac{\Sigma\lvert y-\hat{y}\rvert}{N}$	8.5

The MAE has a big advantage in that **the units of the MAE are the same as the units of** yy**,** the feature we want to predict. In the example above, we have an MAE of 8.5, so it means that on average our predictions of the number of machine failures are incorrect by 8.5 machine failures.

4.2.2 Root Mean Squared Error (RMSE):

It measures the average error performed by the model in predicting the outcome for an observation. Its calculation is very similar to MAE, but instead of taking the *absolute* value to get rid of the sign on the individual errors, we *square* the error (because the square of a negative number is positive).

The formula for RMSE is:

$$RMSE = \sqrt{\frac{\Sigma(y-\hat{y})^2}{N}}$$

Age	Failures y	Prediction $\hat{y}$	Error $y-\hat{y}$	Error^2 $(y-\hat{y})^2$
10	15	26	11	121
20	30	32	2	4
40	40	44	4	16
50	55	50	-5	25
70	75	62	-13	169
90	90	74	-16	256

Mean of Error^2	$\frac{\Sigma(y-\hat{y})^2}{N}$	98.5
Square root of Mean of Error^2	$\sqrt{\frac{\Sigma(y-\hat{y})^2}{N}}$	**9.9**

As with MAE, we can think of **RMSE as being measured in the y units.** So the above error can be read as an error of 9.9 machine failures on average per observation.

4.2.3 R-Squared:

It tells us the degree to which the model explains the variance in the data. In other words how much better it is than just predicting the mean.

- A value of 1 indicates a perfect fit.
- A value of 0 indicates a model no better than the mean.
- A value less than 0 indicates a model worse than just predicting the mean.

4.2.4 Residual Standard Error:

The **residual standard error** is √MSE. The MSE is an unbiased estimator of σ^2, where σ^2=Var(y|x).

For example: **Anova table of *SLR/Simple Linear Regression* (DF is different for multiple regression):**

Source	DF	Sum Sq	Mean Sq	F value
Regression	1	SSR	$MSR = \frac{SSR}{1}$	$\frac{MSR}{MSE}$
Residual	$n-2$	SSE	$MSE = \frac{SSE}{n-2}$	
Total	$n-1$	SST		

where n is the sample size of x_i, $SST = SSE + SSR$, $SST = S_{YY} = \sum_{i=1}^{n} (y_i - \bar{y})^2$, $SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$, $SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$.

The SSR is the part of variance of y_i which can be explained by $\hat{y}_i$, the greater the better.

Also for **SLR**, $se(\beta_1) = \sqrt{MSE}/\sqrt{S_{xx}}$, where S_{XX} is defined similarly as S_{YY}.

4.3 OVERFITTING

Overfitting a model is a condition where a statistical model begins to describe the random error in the data rather than the relationships between variables. This problem occurs when the model is too complex.

In regression analysis, overfitting can produce misleading R-squared values, regression coefficients, and p-values. In this post, I explain how overfitting models is a problem and how you can identify and avoid it. Overfit regression models have too many terms for the number of observations. When this occurs, the regression coefficients represent the noise rather than the genuine relationships in the population.

That's problematic by itself. However, there is another problem. Each sample has its own unique quirks. Consequently, a regression model that becomes tailor-made to fit the random quirks of one sample is unlikely to fit the random quirks of another sample. Thus, overfitting a regression model reduces its generalizability outside the original dataset.

4.3.1 Graphical Illustration of Overfitting Regression Models:

The image below illustrates an overfit model. The green line represents the true relationship between the variables. The random error inherent in the data causes the data points to fall randomly around the green fit line. The red line represents an overfit model. This model is too complex, and it attempts to explain the random error present in the data.

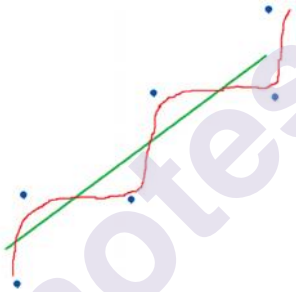


Fig 4.4: Overfitting

The example above is very clear. However, it's not always that obvious. Below, the fitted line plot shows an overfit model. In the graph, it appears that the model explains a good proportion of the dependent variable variance.

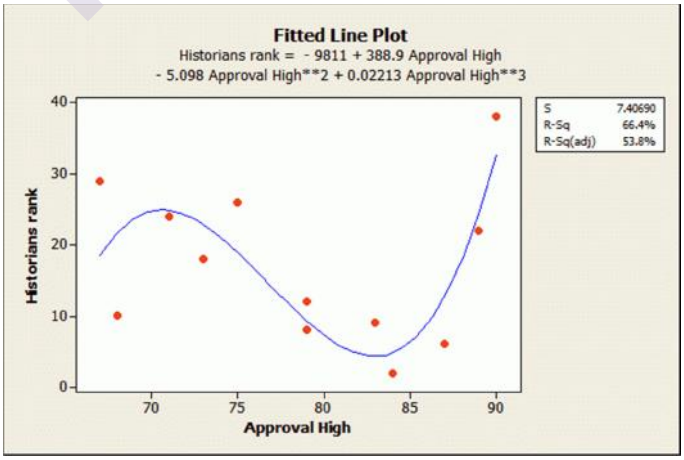


Fig 4.5: Overfitting Line

4.3.2 Catalysts for Overfitting:

This concept is fairly intuitive. Suppose we have a total sample size of 20 and we need to estimate one population mean using a 1-sample t-test. We'll probably obtain a good estimate. However, if we want to use a 2-sample t-test to estimate the means of two populations, it's not as good because we have only ten observations to estimate each mean. If we want to estimate three or more means using one-way ANOVA, it becomes pretty bad.

As the number of observations per estimate decreases (20, 10, 6.7, etc.), the estimates become more erratic. Furthermore, a new sample is unlikely to replicate the inconsistent estimates produced by the smaller sample sizes.

In short, the quality of the estimates deteriorates as you draw more conclusions from a sample. This idea is directly related to the degrees of freedom in the analysis. To learn more about this concept, read my post: Degrees of Freedom in Statistics.

4.3.3 Applying These Concepts to Overfitting Regression Models:

Overfitting a regression model is similar to the example above. The problems occur when you try to estimate too many parameters from the sample. Each term in the model forces the regression analysis to estimate a parameter using a fixed sample size. Therefore, the size of your sample restricts the number of terms that you can safely add to the model before you obtain erratic estimates.

Similar to the example with the means, you need a sufficient number of observations for each term in the regression model to help ensure trustworthy results. Statisticians have conducted simulation studies* which indicate you should have at least 10-15 observations for each term in a linear model. The number of terms in a model is the sum of all the independent variables, their interactions, and polynomial terms to model curvature.

For instance, if the regression model has two independent variables and their interaction term, you have three terms and need 30-45 observations. Although, if the model has multicollinearity or if the effect size is small, you might need more observations.

To obtain reliable results, you need a sample size that is large enough to handle the model complexity that your study requires. If your study calls for a complex model, you must collect a relatively large sample size. If the sample is too small, you can't dependably fit a model that approaches the true model for your independent variable. In that case, the results can be misleading.

4.3.4 How to Detect Overfit Models:

Linear regression, there is an excellent accelerated cross-validation method called predicted R-squared. This method doesn't require you to collect a separate sample or partition your data, and you can obtain the cross-validated results as you fit the model. Statistical software calculates predicted R-squared using the following automated procedure:

- It removes a data point from the dataset.
- Calculates the regression equation.
- Evaluates how well the model predicts the missing observation.
- And, repeats this for all data points in the dataset.

Predicted R-squared has several cool features. First, you can just include it in the output as you fit the model without any extra steps on your part. Second, it's easy to interpret. You simply compare predicted R-squared to the regular R-squared and see if there is a big difference.

If there is a large discrepancy between the two values, your model doesn't predict new observations as well as it fits the original dataset. The results are not generalizable, and there's a good chance you're overfitting the model.

For the fitted line plot above, the model produces a predicted R-squared (not shown) of 0%, which reveals the overfitting. For more information, read my post about how to interpret predicted R-squared, which also covers the model in the fitted line plot in more detail.`

4.3.5 How to Avoid Overfitting Models:

To avoid overfitting a regression model, you should draw a random sample that is large enough to handle all of the terms that you expect to include in your model. This process requires that you investigate similar studies before you collect data. The goal is to identify relevant variables and terms that you are likely to include in your own model. After you get a sense of the typical complexity of models in your study area, you'll be able to estimate a good sample size.

4.4 CASE STUDY OF POLYNOMIAL REGRESSION

Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial. The Polynomial Regression equation is given below:

$$y= b_0+b_1x_1+ b_2x_1^2+ b_2x_1^3+..... b_nx_1^n$$

- It is also called the special case of Multiple Linear Regression in ML. Because we add some polynomial terms to the Multiple Linear regression equation to convert it into Polynomial Regression.
- It is a linear model with some modification in order to increase the accuracy.

- The dataset used in Polynomial regression for training is of non-linear nature.
- It makes use of a linear regression model to fit the complicated and non-linear functions and datasets.
- **Hence, "*In Polynomial regression, the original features are converted into Polynomial features of required degree (2,3,..,n) and then modeled using a linear model.*"**

4.4.1 Need for Polynomial Regression:

- The need of Polynomial Regression in ML can be understood in the below points:
- If we apply a linear model on a **linear dataset**, then it provides us a good result as we have seen in Simple Linear Regression, but if we apply the same model without any modification on a **non-linear dataset**, then it will produce a drastic output. Due to which loss function will increase, the error rate will be high, and accuracy will be decreased.
- So for such cases, **where data points are arranged in a non-linear fashion, we need the Polynomial Regression model**. We can understand it in a better way using the below comparison diagram of the linear dataset and non-linear dataset.

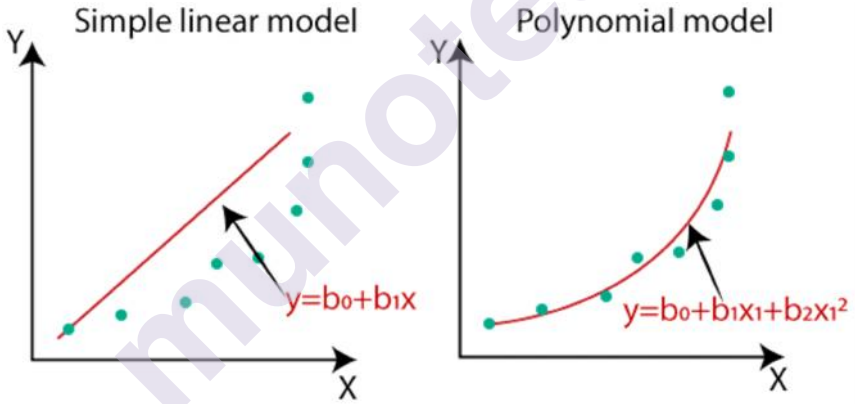


Fig 4.6: linear regression and polynomial model

- In the above figure, we have taken a dataset which is arranged non-linearly. So if we try to cover it with a linear model, then we can clearly see that it hardly covers any data point. On the other hand, a curve is suitable to cover most of the data points, which is of the Polynomial model.
- Hence, *if the datasets are arranged in a non-linear fashion, then we should use the Polynomial Regression model instead of Simple Linear Regression.*

4.4.2 Equation of the Polynomial Regression Model:

Simple Linear Regression equation:

$y = b_0+b_1x$ .....(a)

Multiple Linear Regression equation:
$y= b_0+b_1x+ b_2x_2+ b_3x_3+....+ b_nx_n$ .....(b)

Polynomial Regression equation:
$y= b_0+b_1x + b_2x^2+ b_3x^3+....+ b_nx^n$ .....(c)

When we compare the above three equations, we can clearly see that all three equations are Polynomial equations but differ by the degree of variables. The Simple and Multiple Linear equations are also Polynomial equations with a single degree, and the Polynomial regression equation is Linear equation with the nth degree. So if we add a degree to our linear equations, then it will be converted into Polynomial Linear equations.

Here we will implement the Polynomial Regression using Python. We will understand it by comparing Polynomial Regression model with the Simple Linear Regression model. So first, let's understand the problem for which we are going to build the model.

Problem Description: There is a Human Resource company, which is going to hire a new candidate. The candidate has told his previous salary 160K per annum, and the HR have to check whether he is telling the truth or bluff. So to identify this, they only have a dataset of his previous company in which the salaries of the top 10 positions are mentioned with their levels. By checking the dataset available, we have found that there is a **non-linear relationship between the Position levels and the salaries**. Our goal is to build a **Bluffing detector regression** model, so HR can hire an honest candidate. Below are the steps to build such a model.

Position	Level(X-variable)	Salary(Y-Variable)
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

Steps for Polynomial Regression:

The main steps involved in Polynomial Regression are given below:

- Data Pre-processing

- Build a Linear Regression model and fit it to the dataset
- Build a Polynomial Regression model and fit it to the dataset
- Visualize the result for Linear Regression and Polynomial Regression model.
- Predicting the output.

Data Pre-processing Step:

The data pre-processing step will remain the same as in previous regression models, except for some changes. In the Polynomial Regression model, we will not use feature scaling, and also we will not split our dataset into training and test set. It has two reasons:

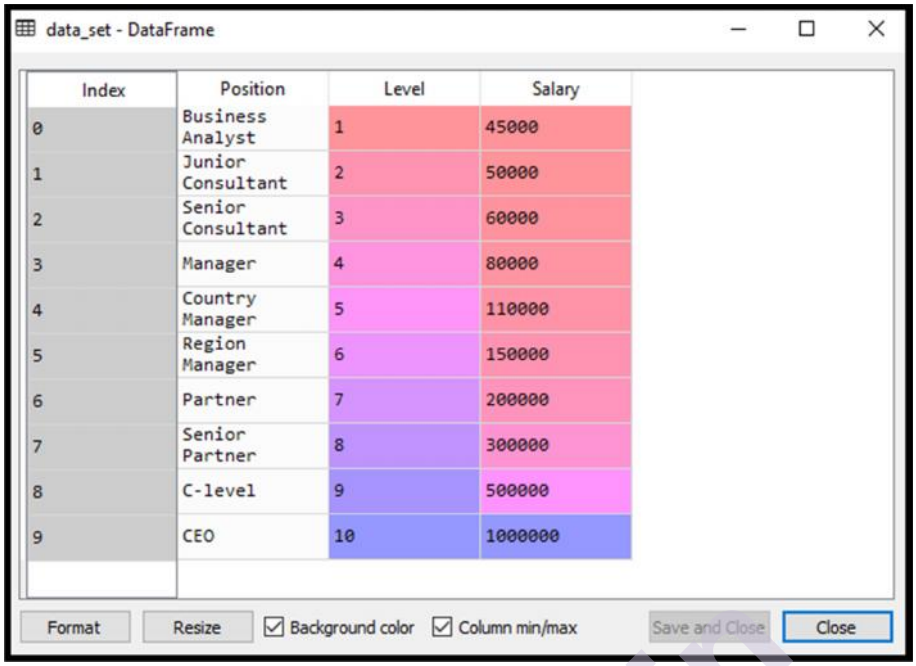
- The dataset contains very less information which is not suitable to divide it into a test and training set, else our model will not be able to find the correlations between the salaries and levels.
- In this model, we want very accurate predictions for salary, so the model should have enough information.

The code for pre-processing step is given below:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('Position_Salaries.csv')
  #Extracting Independent and dependent Variable
x= data_set.iloc[:, 1:2].values
y= data_set.iloc[:, 2].values
```

- In the above lines of code, we have imported the important Python libraries to import dataset and operate on it.
- Next, we have imported the dataset '**Position_Salaries.csv**', which contains three columns (Position, Levels, and Salary), but we will consider only two columns (Salary and Levels).
- After that, we have extracted the dependent(Y) and independent variable(X) from the dataset. For x-variable, we have taken parameters as [:,1:2], because we want 1 index(levels), and included :2 to make it as a matrix.



Index	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

Now, we will build and fit the Linear regression model to the dataset. In building polynomial regression, we will take the Linear regression model as reference and compare both the results. The code is given below:

```
1. #Fitting the Linear Regression to the dataset
2. from sklearn.linear_model import LinearRegression
3. lin_regs= LinearRegression()
4. lin_regs.fit(x,y)
```

In the above code, we have created the Simple Linear model using **lin_regs** object of **LinearRegression** class and fitted it to the dataset variables (x and y).

Output:

```
Out[5]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Building the Polynomial regression model:

Now we will build the Polynomial Regression model, but it will be a little different from the Simple Linear model. Because here we will use **PolynomialFeatures** class of **preprocessing** library. We are using this class to add some extra features to our dataset.

```
1. #Fitting the Polynomial regression to the dataset
2. from sklearn.preprocessing import PolynomialFeatures
3. poly_regs= PolynomialFeatures(degree= 2)
4. x_poly= poly_regs.fit_transform(x)
5. lin_reg_2 =LinearRegression()
6. lin_reg_2.fit(x_poly, y)
```

In the above lines of code, we have used **poly_regs.fit_transform(x)**, because first we are converting our feature matrix into polynomial feature matrix, and then fitting it to the Polynomial regression model. The parameter value(degree= 2) depends on our choice. We can choose it according to our Polynomial features.

After executing the code, we will get another matrix **x_poly**, which can be seen under the variable explorer option:

	0	1	2
0	1	1	1
1	1	2	4
2	1	3	9
3	1	4	16
4	1	5	25
5	1	6	36
6	1	7	49
7	1	8	64
8	1	9	81
9	1	10	100

Output:

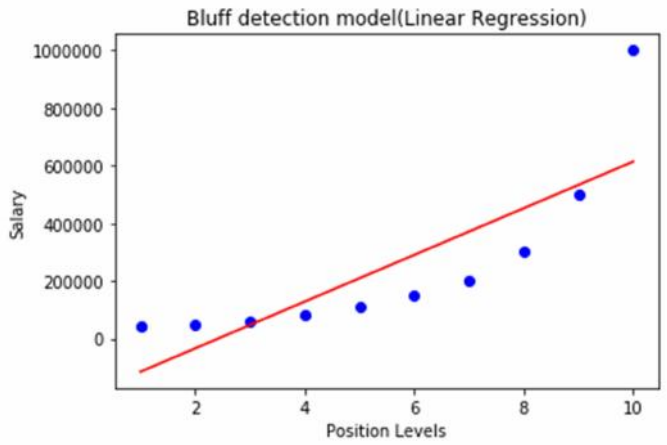
```
Out[11]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Visualizing the result for Linear regression:

Now we will visualize the result for Linear regression model as we did in Simple Linear Regression. Below is the code for it:

```
1. #Visulaizing the result for Linear Regression model
2. mtp.scatter(x,y,color="blue")
3. mtp.plot(x,lin_regs.predict(x), color="red")
4. mtp.title("Bluff detection model(Linear Regression)")
5. mtp.xlabel("Position Levels")
6. mtp.ylabel("Salary")
7. mtp.show()
```

Output:



In the above output image, we can clearly see that the regression line is so far from the datasets. Predictions are in a red straight line, and blue points are actual values. If we consider this output to predict the value of CEO, it will give a salary of approx. 600000$, which is far away from the real value.

Visualizing the result for Polynomial Regression:

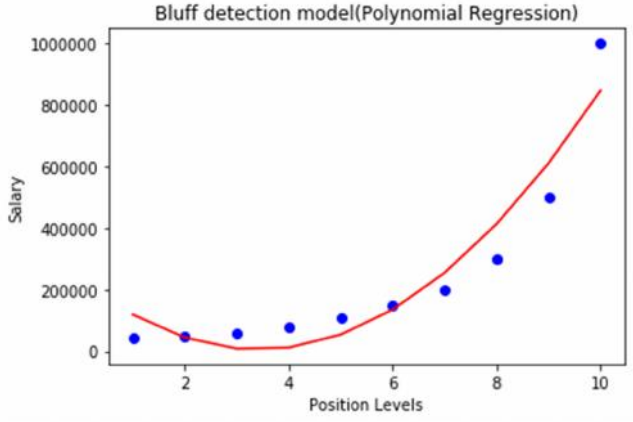
Here we will visualize the result of Polynomial regression model, code for which is little different from the above model.

Code for this is given below:

```
#Visulaizing the result for Polynomial Regression
mtp.scatter(x,y,color="blue")
mtp.plot(x, lin_reg_2.predict(poly_regs.fit_transform(x)), color="red")
mtp.title("Bluff detection model(Polynomial Regression)")
mtp.xlabel("Position Levels")
mtp.ylabel("Salary")
mtp.show()
```

In the above code, we have taken lin_reg_2.predict(poly_regs.fit_transform(x), instead of x_poly, because we want a Linear regressor object to predict the polynomial features matrix.

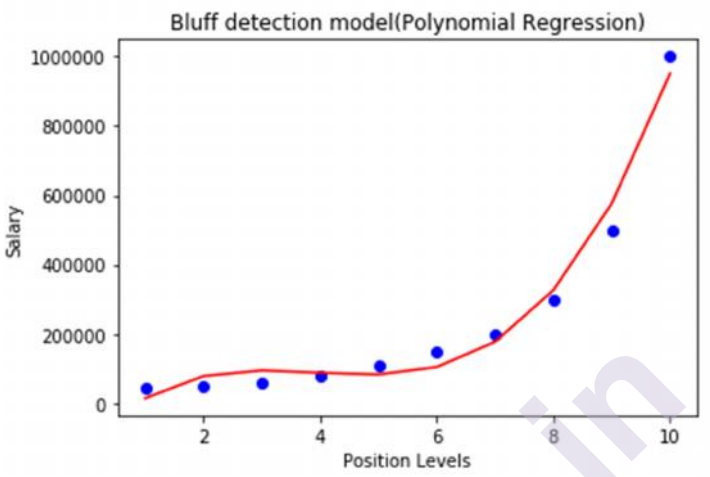
Output:



As we can see in the above output image, the predictions are close to the real values. The above plot will vary as we will change the degree.

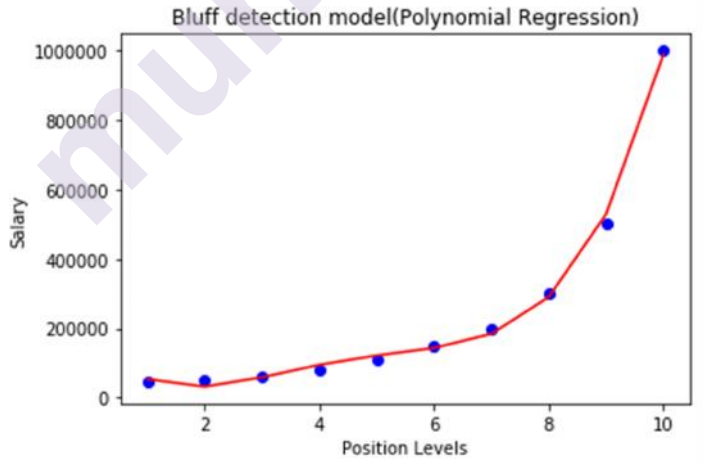
For degree= 3:

If we change the degree=3, then we will give a more accurate plot, as shown in the below image.



SO as we can see here in the above output image, the predicted salary for level 6.5 is near to 170K$-190k$, which seems that future employee is saying the truth about his salary.

Degree= 4: Let's again change the degree to 4, and now will get the most accurate plot. Hence we can get more accurate results by increasing the degree of Polynomial.



Predicting the final result with the Linear Regression model:

Now, we will predict the final output using the Linear regression model to see whether an employee is saying truth or bluff. So, for this, we will use the **predict()** method and will pass the value 6.5. Below is the code for it:

```
1.  lin_pred = lin_regs.predict([[6.5]])
2.  print(lin_pred)
```

Output:

```
[330378.78787879]
```

Predicting the final result with the Polynomial Regression model:

Now, we will predict the final output using the Polynomial Regression model to compare with Linear model. Below is the code for it:

```
1. poly_pred = lin_reg_2.predict(poly_regs.fit_transform([[6.5]]))
2. print(poly_pred)
```

Output:

```
[158862.45265153]
```

SUMMARY

Regression analysis is a statistical technique for studying linear relationships. It begins by supposing a general form for the relationship, known as the regression model. The ultimate goal of the regression algorithm is to plot a best-fit line or a curve between the data. The three main metrics that are used for evaluating the trained regression model are variance, bias and error. If the variance is high, it leads to overfitting and when the bias is high, it leads to underfitting. Based on the number of input features and output labels, regression is classified as linear (one input and one output), multiple (many inputs and one output) and multivariate (many outputs). Linear regression allows us to plot a linear equation, i.e., a straight line. We need to tune the coefficient and bias of the linear equation over the training data for accurate predictions. The tuning of coefficient and bias is achieved through gradient descent or a cost function — least squares method. Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available.

UNIT END QUESTIONS

1. What is regression? Explain types of regression.
2. Give the illustration of regression performance.
3. Explain the methods used for regression analysis.
4. Write a note on R square method.
5. Write a note on Mean absolute error.
6. Explain Root mean square method with suitable example.
7. Discuss Polynomial Regression in detail.

References

- Peter Flach, Machine Learning The Art and Science of Algorithms that Make Sense of Data, Cambridge Press, 2012
- https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a
- Baidaa M Alsafy, Zahoor Mosad, Wamidh k. Mutlag, Multiclass Classification Methods: A Review, International Journal of Advanced Engineering Technology and Innovative Science (IJAETIS), 2020.
- Dragos D. Margineantu, Class Probability Estimation and Cost-Sensitive Classification Decisions, Inc Springer-Verlag Berlin Heidelberg 2002
- https://www.educative.io/edpresso/what-is-linear-regression

*****

5

THEORY OF GENERALIZATION

Unit Structure

5.0 OBJECTIVES

A learning will learn:

- hyptothesis implementation
- to implement the regularization theory
- understand the VC dimensions in Machine learning.

5.1 EFFECTIVE NUMBER OF HYPOTHESIS

A hypothesis is an explanation for something. It is a provisional idea, an educated guess that requires some evaluation. A good hypothesis is testable; it can be either true or false. In science, a hypothesis must be falsifiable, meaning that there exists a test whose outcome could mean that the hypothesis is not true. The hypothesis must also be framed before the outcome of the test is known.

Consider the regression estimation problem where X = Y = R and the data are the following points:

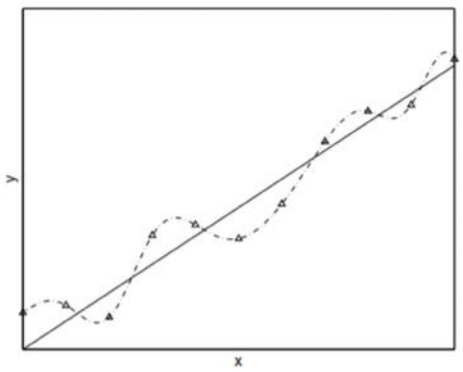


Fig 5.1: regression estimation

where the dash-dot line represents are fairly complex model and fits the data set perfectly, and the straight line does not completely "explain" the data but seems to be a "simpler" model, if we argue that the residuals are possibly measurement errors.

Statistical hypothesis tests are techniques used to calculate a critical value called an "*effect*." The critical value can then be interpreted in order to determine how likely it is to observe the effect if a relationship does not exist. If the likelihood is very small, then it suggests that the effect is probably real. If the likelihood is large, then we may have observed a statistical fluctuation, and the effect is probably not real. For example, we may be interested in evaluating the relationship between the means of two samples, e.g. whether the samples were drawn from the same distribution or not, whether there is a difference between them.

One hypothesis is that there is no difference between the population means, based on the data samples. This is a hypothesis of no effect and is called the null hypothesis and we can use the statistical hypothesis test to either reject this hypothesis, or fail to reject (retain) it. We don't say "accept" because the outcome is probabilistic and could still be wrong, just with a very low probability.

5.1.1 Underfitting versus overfitting:

In the machine learning literature the more complex model is said to show signs of overfitting, while the simpler model underfitting. Often several heuristic are developed in order to avoid overfitting, for example, when designing neural networks one may:

1. limit the number of hidden nodes (equivalent to reducing the number of support vectors),
2. stop training early to avoid a perfect explanation of the training set, and
3. apply weight decay to limit the size of the weights, and thus of the function class implemented by the network (equivalent to the regularization used in ridge regression).

5.1.2 Minimizing risk (expected loss):

The risk or expected loss (L) is defined as:

$$\mathrm{err}_{\mathcal{D}}(h) = \mathsf{E}[\tfrac{1}{\ell}\mathrm{err}_{S_t}(h)] = \int_{\mathcal{X}\times\mathcal{Y}} L(\boldsymbol{x}, y, h(\boldsymbol{x}))d\mathcal{D}(\boldsymbol{x}, y)$$

where S_t is a test set. This problem is intractable since we do not know D(x, y) which is the probability distribution on X ×Y which governs the data generation and underlying functional dependency.

The only thing we do have it our training set S sampled from the distribution D and we can use this to approximate the above integral by the finite sum:

$$\frac{1}{\ell}\text{err}_S(h) = \frac{1}{\ell}\sum_{i=1}^{\ell} L(\boldsymbol{x}_i, y_i, h(\boldsymbol{x}_i))$$

If we allow h to be taken from a very large class of function $H = \mathcal{Y}^{\mathcal{X}}$, we can always find a h that leads to a rather small value of $\frac{1}{\ell}\text{err}_S(h)$, but yet be distant from minimizer of $\text{err}_{\mathcal{D}}(h)$. For example, a look-up-table for the training set would give excellent results for that set, but contain no information about any other points.

5.2 BOUNDING THE GROWTH FUNCTION

We have considered the case when H is finite or countably infinite. In practice, however, the function class H could be uncountable. Under this situation, the previous method does not work. The key idea is to group functions based on the sample. Given a sample Dn = $\{(x_1, y_1),...,(x_n, y_n)\}$, and define S = $\{x_1, . . . , x_n\}$. Consider the set HS = $H_{x1,...,xn}$ = $\{(h(x_1),...,h(x_n) : h \in H\}$. The size of this set is the total number of possible ways that S = $\{x_1, . . . , x_n\}$ can be classified. For binary classification the cardinality of this set is always finite, no matter how large H is.

The growth function is the maximum number of ways into which n points can be classified by the function class:
$G_H(n) = \sup_{x1,...,xn} |H_S|$.

Growth function can be thought as a measure of the "size" for the class of functions H. Several facts about the growth function:

- When H is finite, we always have $G_H(n) \leq |H| = m$.
- Since $h(x) \in \{0, 1\}$, we have $G_H(n) \leq 2n$. If $G_H(n) = 2n$, then there is a set of n points such that the class of functions H can generate any possible classification result on these points.

5.3 VC (VAPNIK-CHERVONENKIS) DIMENSIONS

The VC (Vapnik-Chervonenkis) dimension is a single parameter that characterizes the growth function:

The VC dimension of a hypothesis set H, denoted by $d_{VC}(H)$, is the largest value of m, for which $\Pi_H(m) = 2^m$. If $\Pi_H(m) = 2^m$, then $d_{VC}(H) = \infty$.

To illustrate this definition, we will now take the examples for the growth function to learn their VC-dimension.To find a lower bound we have to simply find a set S that can be shattered by H. To give an upper bound, we need to prove that no set S of d + 1 points exists, that can be shattered by H, which is usually more difficult.

For Example, **Positive rays:** We have shown that the growth function for positive rays is$\Pi_H(m)$ = m+1. Only for m=0 ,1 we have $\Pi_H(m) = 2^m$, therefore $d_{VC}(H)=1$

Positive intervals: The growth function for positive intervals is, $\Pi_H(m) = \frac{1}{2}$ $m^2 + \frac{1}{2}m + .$
We have $\Pi_H(m) = 2^m$ for m = 0,1,2 which yields $d_{VC}(H)=2$.

Convex sets: We have seen that by arrange convex sets in the right way, sets of every size can be shattered.Therefore$\Pi_H(m) =$ 2m for all m $d_{VC}(H)= \infty$.

Perceptrons: The next example will be a bit more elaborate. We will show that for the perceptron in $R^d$the VC-dimension is always d + 1. We won't explicitly calculate the growth function $\Pi_H(m)$ for all m.Therefore, for determining the VC dimension $d_{VC}(H) = d + 1$, we have to show that:

a) The VC dimension is at least d+1: To prove this, we have to find d + 1 points in the input space $X = R^d$ that the perceptron can shatter. We first consider the set of hyperplanes in R^2 . We have already seen that any three non-collinear points in R^2 can be shattered by Perceptron. $d_{VC}(H)$ = (hyperplanes in R^2) = 3.If we now show the general case of hyperplanes in R^d. We pick a set of d + 1 points inR^d, setting x_0 to be the origin and defining x_i , i∈ {1, ..., x}as the points whose i-th coordinate is 1 and all others are 0.Let y_0, ..., y_n∈ {−1, +1} be an arbitrary set of labels for x_0, x_1, ...x_d.Let wbe the vector whose ith coordinate is y_i .The perceptron defined by the hyperplane of equation $w \cdot x + \frac{y0}{2} = 0$ is shatters x_0, ..., x_d because for any i∈ {0, ..., d}:

$$\text{sgn}(w \cdot x_i + \frac{y0}{2}) = \text{sgn}(y_i + \frac{y0}{2}).$$

Now we need to show that b) the VC-dimension is at most d + 1. This is a bit trickier, because we have to show that there is no subset of more than d + 1 points that can be shattered by the perceptron. To prove this, we will show that on any d + 2 points, there is a dichotomy that can not be realized by the perceptron classifier.

Choose the points x_1, ..., x_{d+2} at random. There are more points than dimension, therefore we must have

$$x_j = \sum_{i \neq j} a_i x_i,$$

i.e. one point is a linear combination of the rest of the points. This will apply to any set of d + 2 points you choose. Also, some of the a_i's must be nonzero, because the first coordinate of the x_i's is always one (see definition of the perceptron, first coordinate is one to include the bias term of the hyperplane into the form w · x).Now we show a dichotomy that can't be implemented: Consider the following dichotomy.Let y_1, ..., y_{d+2} the labels of x1, ..., x_{d+2}. Give x_i with nonzero coefficient ai get the label +1, give any label to the x_i with $a_i = 0$ and set $y_i = -1$ as the label of x_i . Let $w \in R^d + 1$ be the weight vector to any hyperplane h. Now we have,

$$x_j = \sum_{i \neq j} a_i x_i \Rightarrow w^T x_j = \sum_{i \neq j} a_i w^T x_i$$

If $y_i = \text{sgn}(w^T x_i) = \text{sgn}(a_i)$, then $a_i w^T x_i > 0$ for all $0 \leq i \leq d$. Then $\text{sgn}(\sum_{i \neq j} a\ w^T x_i > 0$. However, we set $y_j = \text{sgn}(w^T x_j) = \text{sgn}(\sum_{i \neq j} a\ w^T x_i) < 0$ that gives a contradiction. The dichotomy can't be implemented on any set of d+2 points by the perceptron classifier.

Combining both parts of the proof, we get: d_{VC}(hyperplanes in R^d) = d + 1.Consider again the perceptron in the two-dimensional space. As shown above, its VCdimension is three. The fact that no four points can be shattered by H limits the number of the dichotomies that can be realized significantly. We exploit that fact to get a bound

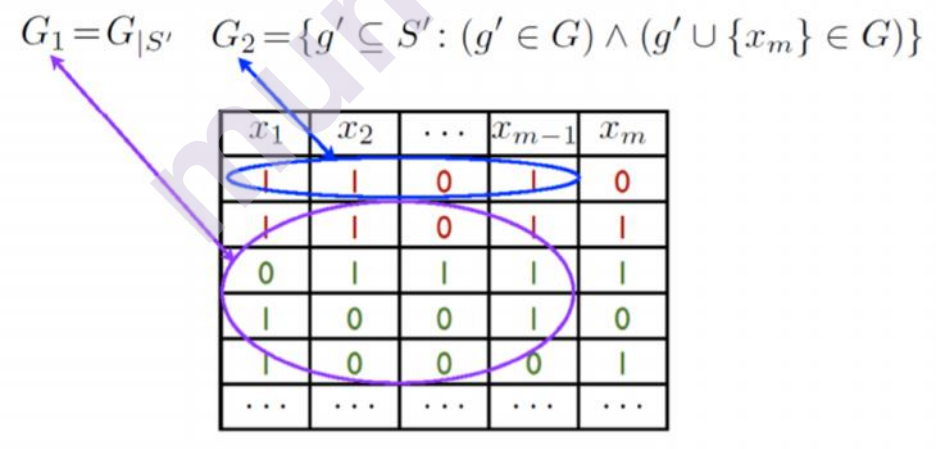


Fig 5.2: Construction of G1 and G2

for $\Pi_H(m)$ for all m ∈ N. We will prove, that, if the VC-dimension for a set of hypotheses is finite, then there is a polynomial that bounds ΠH(m) for all values of m. If such a polynomial exists, and ΠH(m) can replace |H| in above equation then the generalization error will go to zero as $m \to \infty$. The next result uses the VC-dimension to define a bound for the growth function.

5.4 REGULARIZATION THEORY

Many strategies used in machine learning are explicitly designed to reduce the test error, possibly at the expense of increased training error. In another way we can say that any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error is regularization. Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.

There are many regularization strategies. Some put extra constraints on a machine learning model, such as adding restrictions on the parameter values. Some add extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values. These strategies are collectively known as **regularization**.

In fact, developing more effective regularization strategies has been one of the major research efforts in the machine learning field. Sometimes these constraints and penalties are designed to encode specific kinds of prior knowledge. Other times, these constraints and penalties are designed to express a generic preference for a simpler model class in order to promote generalization. Sometimes penalties and constraints are necessary to make an underdetermined problem determined.

There are several form of **regularizations** by which we can prevent overfitting in our network or machine learning model.

Parameter Norm Penalties:
Many regularization approaches are based on limiting the capacity of models, such as neural networks, linear regression, or logistic regression, by adding a parameter norm penalty $\Omega(\boldsymbol{\theta})$ to the objective function J. We denote the regularized objective function by J.

$$J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = J(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\boldsymbol{\theta}) \text{ — } \{1\}$$

where $\alpha \in [0, \infty)$ is a hyperparameter that weights the relative contribution of the norm penalty term, Ω, relative to the standard objective function J. Setting α to zero results in no regularization. Larger values of α correspond to more regularization.

We note that for neural networks, we typically choose to use a parameter norm penalty Ω that penalizes only the weights of the a ne transformation at each layer and leaves the biases unregularized. **The biases typically require less data to fit accurately than the weights**. Each weight specifies how two variables interact. Fitting the weight well requires observing both variables in a variety of conditions. Each bias controls only a single variable. This means that we do not induce too much variance by leaving the biases unregularized. Also, **regularizing the bias parameters can introduce a significant amount of underfitting**. We therefore use the

vector **w** to indicate all the weights that should be a ected by a norm penalty, while the vector **θ** denotes all of the parameters, including both **w** and the unregularized parameters.

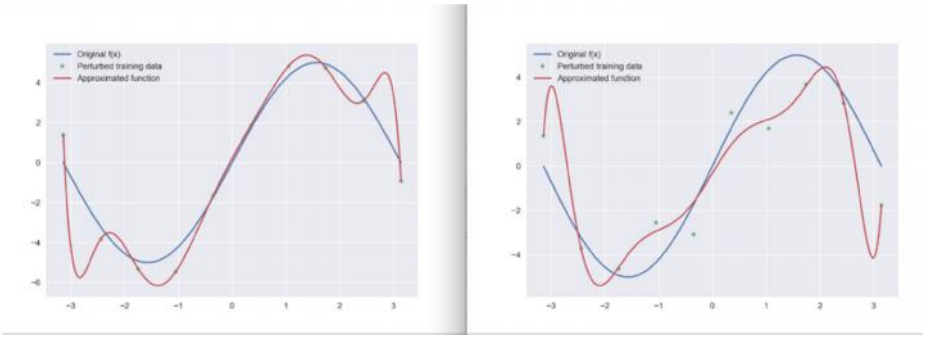


Fig 5.3: Overfitting and Regularization

L² regularization: It is one of the commonly used regularization form. The L² parameter norm penalty commonly known as weight decay. L² regularization drives the weights closer to origin by adding a regularization term $\Omega(\boldsymbol{\theta}) = 1/2\|\mathbf{w}\|^2$ to the objective function. Such a model has following total objective function:

$J(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \alpha/2(\mathbf{w}`\mathbf{w}) + J(\mathbf{w}; \mathbf{X}, \mathbf{y})$ ( ` means transpose)

The L² regularization has the intuitive interpretation of heavily penalizing peaky weight vectors and preferring diffuse weight vectors. Due to multiplicative interactions between weights and inputs this has the appealing property of encouraging the network to use all of its inputs a little rather that some of its inputs a lot.

Lastly, also notice that during gradient descent parameter update, using the L² regularization ultimately means that every weight is decayed linearly: **W** += -lambda * **W** towards zero. Let's see what this means, We can see that the addition of the weight decay term has modified the learning rule to multiplicatively shrink the weight vector by a constant factor on each step, just before performing the usual gradient update. This describes what happens in a single step. But what happens over the entire course of training? The L² regularization causes the learning algorithm to "perceive" the input **X** as having higher variance, which makes it shrink the weights on features whose covariance with the output target is low compared to this added variance.

L¹ regularization: The L¹ regularization has the intriguing and fascinating property that it leads the weight vectors to become sparse during optimization (i.e. very close to exactly zero). In other words, neurons with L¹ regularization end up using only a sparse subset of their most important inputs as most weight goes very close to zero and become nearly invariant to the "noisy" inputs. In comparison, final weight vectors from L² regularization are usually diffuse, small numbers. **The sparsity property induced by L¹ regularization has been used extensively as a feature**

selection mechanism. The L^1 penalty causes a subset of the weights to become zero, suggesting that the corresponding features may safely be discarded. In practice, if you are not concerned with explicit feature selection, L^2 regularization can be expected to give superior performance over L1.

Formally, L^2 regularization on the model parameter **w** is defined as

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|,$$

SUMMARY

We have derivated bounds for finite hypothesis sets. But in machine learning, the hypothesis sets are usually infinite. We show that the hypotheses in a hypothesis set H can be "similar" to each other and therefore their "bad events" with poor generalization can overlap. Therefore, we define the growth function, that formalizes the number of "effective" hypotheses in a hypothesis set. The VC (Vapnik-Chervonenkis) dimension is a single parameter that characterizes the growth function. A regression model which uses L1 Regularization technique is called LASSO(Least Absolute Shrinkage and Selection Operator) regression. A regression model that uses L2 regularization technique is called Ridge regression. Lasso Regression adds "absolute value of magnitude" of coefficient as penalty term to the loss function(L).

UNIT END QUESTIONS

1. What is hypothesis? Explain different types of hypothesis.
2. Explain underfitting and overfitting with suitable example.
3. Explain the growth bounding function with suitable derivation.
4. Give the illustration of VC (Vapnik-Chervonenkis) Dimensions.
5. What is regularization? Explain its theory.
6. Explain L1 and L2 regularization with suitable example.

REFERENCES

- Peter Flach, Machine Learning The Art and Science of Algorithms that Make Sense of Data, Cambridge Press, 2012.
- https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a

*****

UNIT III

6

LINEAR MODELS

Unit Structure

6.0 Objectives

The Objective for this chapter is to present the background and mathematical function needed to construct a statistical model that describes a particular scientific or engineering process. These types of models can be used for prediction of process outputs, for calibration, or for process optimization.

6.1 INRODUCTION LEAST SQUARE METHOD

The most widely used modeling method is Linear least squares regression. It is what most people mean when they say they have used "regression", "linear regression" or "least squares" to fit a model to their data. Not only is linear least squares regression the most widely used modeling method, but it has been adapted to a broad range of situations that are outside its direct scope. It plays a strong underlying role in many other modeling methods, including the other methods discussed in this section

6.1.1 Definition:
The process of finding the best-fitting curve for a set of data points by reducing the sum of the squares of the offsets of the points from the curve is called the **least square method**.

The method of least squares defines the solution for the minimization of the sum of squares of errors in equation. to find the variation in observed data we need to find the formula for sum of squares of errors. This method is applied in data fitting. The result of this method is used to reduce the sum of squared errors which are differences between the observed or experimental value and corresponding fitted value given in the model.

The regression analysis is the process of finding the relation between two variables, the trend of outcomes is estimated quantitatively. The method of curve fitting is an approach to regression analysis. This method of fitting equations which approximates the curves to given raw data is the least squares.

If we add up all of the errors, the sum will be zero. So how do we measure overall error? We use a little trick: we square the errors and find a line that minimizes this sum of the squared errors.

$$\sum {e_t}^2 = \sum (Y_i - \overline{Y}_i)^2$$

Following are the basic categories of least-squares problems:

- Ordinary or linear least squares
- Nonlinear least squares

These depend upon linearity or nonlinearity of the errors. The linear problems are often seen in regression analysis in statistics. On the other hand, the non-linear problems are generally used in the iterative method of refinement in which the model is approximated to the linear one with each iteration.

Figure 1: Least square method graph

6.1.3 Formula:

The least-square method states that the curve that best fits a given set of observations, is said to be a curve having a minimum sum of the squared residuals (or deviations or errors) from the given data points. Let us assume that the given points of data are (x1,y1), (x2,y2), (x3,y3), …, (xn,yn) and fitting curve f(x) with d represents error or deviation from each given point.

Now, we can write:

$$d1 = y1 - f(x1)$$
$$d2 = y2 - f(x2)$$
$$d3 = y3 - f(x3)$$
$$\ldots..$$
$$dn = yn - f(xn)$$

The least-squares explain that the curve that best fits is represented by the property that the sum of squares of all the deviations from given values must be minimum. i.e:

$$S = \sum_{i=1}^{n} d_i^2$$

$$S = \sum_{i=1}^{n} [y_i - f_{x_i}]^2$$

$$S = d_1^2 + d_2^2 + d_3^2 + \cdots + d_n^2$$

Sum = Minimum Quantity

6.1.4 Advantages of Least Square method:

1. This method is completely free from personal bias of the analyst as it is very objective in nature.
2. This method provides us with a rate of growth per period.

6.1.5 Disadvantages of Least Square Method:

1. Sensitivity to outliers: One or two outliers can sometimes seriously skew the results of a least-squares analysis.
2. Tendency to overfit data

6.2 MULTIVARIATE LINEAR REGRESSION (MLR)

The general multivariate regression model is a compact way of writing several multiple linear regression models simultaneously. In that sense it is not a separate statistical linear model. The various multiple linear regression models may be compactly written as follows

6.2.1 Normal Equation

Y=XB+U

Matrix form writes MLR model for all *nm* points simultaneously

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E}$$

$$\begin{pmatrix} y_{11} & \cdots & y_{1m} \\ y_{21} & \cdots & y_{2m} \\ y_{31} & \cdots & y_{3m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nm} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ 1 & x_{31} & x_{32} & \cdots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \begin{pmatrix} b_{01} & \cdots & b_{0m} \\ b_{11} & \cdots & b_{1m} \\ b_{21} & \cdots & b_{2m} \\ \vdots & \ddots & \vdots \\ b_{p1} & \cdots & b_{pm} \end{pmatrix} + \begin{pmatrix} e_{11} & \cdots & e_{1m} \\ e_{21} & \cdots & e_{2m} \\ e_{31} & \cdots & e_{3m} \\ \vdots & \ddots & \vdots \\ e_{n1} & \cdots & e_{nm} \end{pmatrix}$$

Where,

Y is a matrix with series of multivariate measurements

X is a matrix of observations on independent variables

B is a matrix containing parameters that are usually to be estimated and
U is a matrix containing errors i.e. noise.

In multivariate linear regression, the focus is on selecting the best possible independent variables that contribute well to the dependent variable.

1. create a correlation matrix for all the independent variables and the dependent variable from the observed data.
2. The correlation value gives us an idea about which variable is significant and by what factor.
3. Low value of correlation between two independent variables shows low overlap between the respective variables and high value points towards high overlap.
4. If two variables highly overlap each other then their contribution in minimizing the loss function is the same therefore making the contribution of one of the layers redundant.

This method can get complicated when there are large no. of independent features that have significant contribution in deciding our dependent variable. Multivariate regression is a technique that estimates a single regression model with more than one outcome variable.

6.2.2 Example:
Alice and Bob are planning to buy a new home. Nancy has mentioned a rate for the home they like, but they want a method to verify it. All they have got is the data set of size M house price and 3 feature counts (no.of bedrooms, size of home in sq mtrs and age of the home). Bob has decided to use his statistical data processing experience from his previous job and do a multivariate linear regression. If **B,S,A** are the parameter values of the house they like and **P** is the price mentioned by Nancy, help Bob decide if they are being cheated or not. (If the price mentioned by Nancy - expected price ≥ 2000 dollars - then they are being cheated)

Input format

Line 1: M value
Next M lines contain 4 values separated by spaces
Line 2 to m+1: Bi Si Ai Pi
Line m+2: B S A (features of their future home)
Line m+3: P (price mentioned by Nancy)
Output format

Pexp C
Pexp is expected price of their home and C is a binary value (0: being cheated, 1: not cheated

SAMPLE INPUT	SAMPLE OUTPUT
2 2 100 10 25000 3 300 2 70000 2 100 10 35000	22375.889180 0

The general linear model incorporates a number of different statistical models: ANOVA, ANCOVA, MANOVA, MANCOVA, ordinary linear regression, t-test and F-test. The general linear model is a generalization of multiple linear regression to the case of more than one dependent variable. If Y, B, and U were column vectors, the matrix equation above would represent multiple linear regression.

6.2.3 Steps for Multivariate Linear Regression:

Steps involved for Multivariate regression analysis are 1) feature selection, 2) normalizing the features, 3) selecting the loss function 4) hypothesis, setting hypothesis parameters, 5) minimizing the loss function, 6) testing the hypothesis, and 7) generating the regression model.

6.2.3.1 Normalizing Features:

We need to scale the features as it maintains general distribution and ratios in data. This will lead to an efficient analysis. The value of each feature can also be changed.

6.2.3.2 Select Loss function and Hypothesis:

The loss function predicts whenever there is an error. Meaning, when the hypothesis prediction deviates from actual values. Here, the hypothesis is the predicted value from the feature/variable.

6.2.3.3 Set Hypothesis Parameters:

The hypothesis parameter needs to be set in such a way that it reduces the loss function and predicts well.

6.2.3.4 Minimize the Loss Function:

The loss function needs to be minimized by using a loss minimization algorithm on the dataset, which will help in adjusting hypothesis parameters. After the loss is minimized, it can be used for further action. Gradient descent is one of the algorithms commonly used for loss minimization.

6.2.3.5 Test the hypothesis function:

The hypothesis function needs to be checked on as well, as it is predicting values. Once this is done, it has to be tested on test data.

6.2.3.6 Multivariate Linear Regression model: Scalar Model

It has the following form

$$y_{ik} = b_{0k} + \sum_{j=1}^{p} b_{jk} x_{ij} + e_{ik}$$

For i € {1,….,n} and k € {1,…..,m}
Where,

$y_{ik} \in \mathbb{R}$ is the kth real valued response for ith observation

$b_{0k} \in \mathbb{R}$ is the regression intercept for the kth response

$b_{jk} \in \mathbb{R}$ is the jth predictor's regression slope for kth response

$x_{ij} \in \mathbb{R}$ is the jth predictor for ith response

$(e_{i1}, \ldots, e_{im}) \overset{\text{iid}}{\sim} \mathrm{N}(\mathbf{0}_m, \mathbf{\Sigma})$ is a multivariate Gaussian error vector.

6.2.4 Advantages:

1. MLR is it helps us to understand the relationships among variables present in the dataset.
2. MLR is a widely used machine learning algorithm.

6.2.5 Disadvantages:

1. This Technique are a bit complex and require a high-levels of mathematical calculation.
2. MLR model's output is not easy to interpret
3. MLR model does not have much scope for smaller datasets.

6.3 REGULARIZATION

6.3.1 Definition:

The type of regression where the coefficient estimates are constrained to zero is called Regularization. The magnitude (size) of coefficients, and error term, are penalized.

Formula:

$$Y \approx \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta pXp$$

Here Y represents the learned relation
β represents the coefficient estimates for different variables or predictors(X).

The Model fitting procedure involves a loss function, known as residual sum of squares(RSS). The coefficients are chosen, such that they minimize this loss function.

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Now, this will adjust the coefficients based on your training data. If there is noise in the training data, then the estimated coefficients will not give efficient results. This is where regularization comes in and regularizes these learned estimates towards zero.

6.3.2 Types of regularized regression:

1. **Ridge regression** is a way to create a sparing model when the number of predictor variables in a set are more than the number of observations.
2. **Least Absolute Shrinkage and Selection Operator (LASSO) regression** is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean.

6.3.3 Ridge regression:

It is a technique that is implemented by adding bias to a multilinear regression model to expect a much more accurate regression with tested data.

The general equation of a best-fit line for multilinear regression is

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots \beta kxk$$

where y is the output variable and x1,x2…xk are predictor variables.
The penalty term for ridge regression is λ(slope) ², where lambda denotes the degree of deflection from the original curve by restricting the coefficients of predictor variables but never makes them zero.

Therefore the equation for ridge regression is

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots \beta kxk + \lambda(\text{slope})^2$$

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Above function shows ridge regression, where the RSS is modified by adding the shrinkage quantity. Now, the coefficients are estimated by minimizing this function. Here, λ is the tuning parameter that decides how much we want to penalize the flexibility of our model. The increase in flexibility of a model is represented by an increase in its coefficients, and

if we want to minimize the above function, then these coefficients need to be small. This is how the Ridge regression technique prevents coefficients from rising too high.

6.3.4 Lasso Regression:

Lasso regression is much similar to ridge regression but only differs in the penalty term. The penalty for lasso regression is λ|slope|.

Lasso regression can even eliminate the variables by making their coefficients to zero thus removing the variables that have high covariance with other predictor variables.

The equation for lasso regression is

$y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots \beta kxk + \lambda|slope|$

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

Lasso is another variation, in which the above function is minimized. It's clear that this variation differs from ridge regression only in penalizing the high coefficients. It uses |βj|(modulus)instead of squares of β, as its penalty. In statistics, this is known as the L1 norm.

6.3.5 Comparison between ridge regression and lasso regression:

According to the above formulation, the ridge regression is expressed by $\beta 1^2 + \beta 2^2 \leq s$. This implies that ridge regression coefficients have the smallest RSS for all points that lie within the circle given by $\beta 1^2 + \beta 2^2 \leq s$. Similarly, for lasso, the equation becomes,$|\beta 1| + |\beta 2| \leq s$. This implies that lasso coefficients have the smallest RSS for all points that lie within the diamond given by $|\beta 1| + |\beta 2| \leq s$.

Comparison of lasso and ridge with the linear regression model, we get:

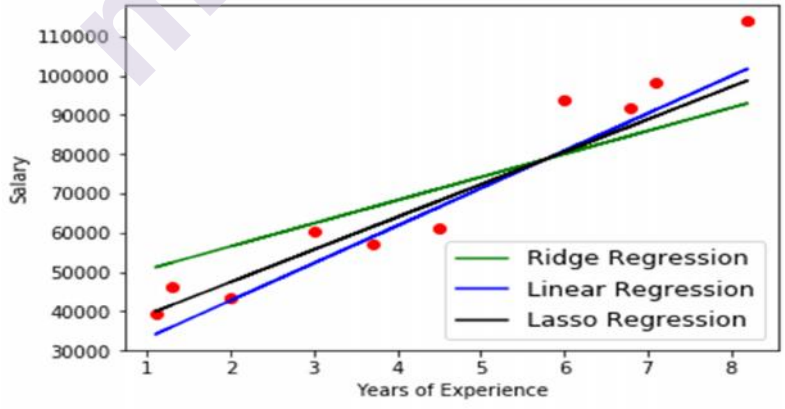


Figure2: Comparison of lasso and ridge with the linear regression model

Note: All the three plots must pass through a single point that is ($\bar{x}$, $\bar{y}$), where $\bar{x}$ is the mean of predictor variables and $\bar{y}$ is the mean of the output variable.

6.4 LEAST SQUARE REGRESSION FOR CLASSIFICATION

Regression and classification are basic in learning. In regression, the output variable takes continuous values, while in classification, the output variable takes class labels.

6.4.1 Linear regression and least squares problem:

The linear regression is similar to linear least squares problem and can be used for classification problems appearing in machine learning algorithms. We will revise the solution of the linear least squares problem in terms of linear regression.

The simplest linear model for regression is
$f(x,\omega)=\omega 0 \cdot 1+\omega 1 x 1+...+\omega M x M$.

Here, $\omega=\{\omega i\},i=0,...,M$ are weights with bias parameter $\omega 0$, $\{xi\},i=1,...,M$ are training examples. Target values (known data) are $\{ti\},i=1,...,N$ which correspond to $\{xi\},i=1,...,M$. Here, M is the number of weights and N is the number of data points.

6.4.2 Introduction:

Each set consists of sample data points repressing two classes. One of the sets represents a linearly-separable classification problem, and the other set is for a non-linearly separable problem. To use the Least Squares Regression to solve a classification problem, a simple trick is used. The data points of the first and second classes are extended by adding a new extra dimension. This produces an augmented cloud of points in n+1 dimensional space, where n is the size of the original data space. In that extra dimension, the data points belonging to the first and second classes take values of -1 and +1 respectively.

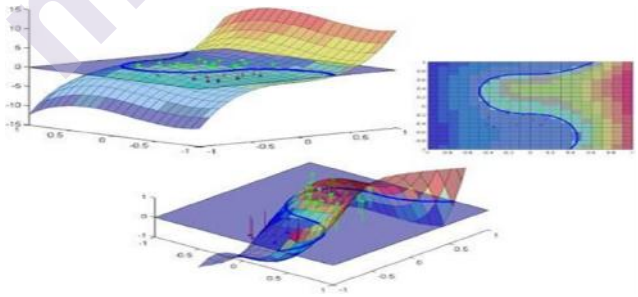


Figure 3: Least square methods to classify linear data from different points of view

Figure 3 shows the decision boundary of classifying linear data from different point of views, and figure 2 shows the same for the wave-alike data, where misclassified samples are circled in red. In such 2D data points case, the decision boundary is the intersection of the fitted

polynomial and the horizontal plane passing by z=0 (z is the extra dimension here).

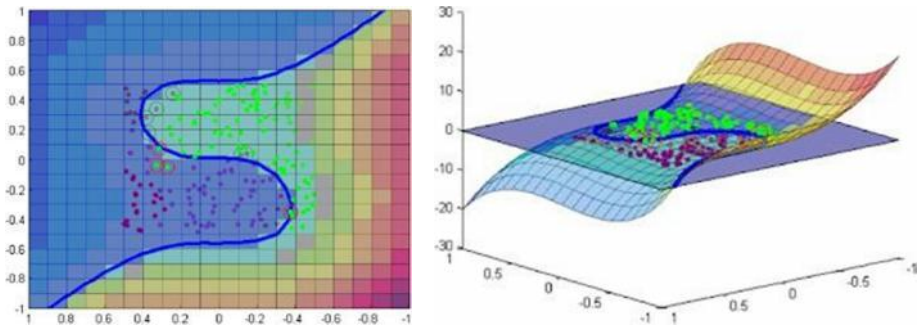


Figure 4: Least square models to classify wave-alike data from different points of view, misclassified samples are circled in red.

For classification accuracy, we use the Minimum Correct Classification Rate (MCCR). MCCR is defined as the minimum of CCR1 and CCR2. CCRn is the ratio of the correctly classified test points in class n divided by the total number of test points in class n.

6.4.3 Non-regularized Least Squares Problem:

In non-regularized linear regression or least squares problem the goal is to minimize the sum of squares

$$E(\omega)=12N\sum n=1(tn-f(x,\omega))2=12N\sum n=1(tn-\omega T\varphi(xn))2:=12\|t-\omega T\varphi(x)\|22$$

to find

$$\min\omega E(\omega)=\min\omega 12\|t-\omega T\varphi(x)\|22.$$

with the residual $r(\omega)=t-\omega T\varphi(x)$. The test functions $\varphi(x)$ form the design matrix A and the regression problem (or the least squares problem) is written as:

$$\min\omega 12\|r(\omega)\|22=\min\omega 12\|A\omega-t\|22,$$

where A is of the size $N\times M$ with $N>M$, t is the target vector of the size N, and ω is vector of weights of the size M.

6.4.4 Regularized Least Squares Problem:

Let now the matrix A will have entries $aij=\phi j(xi),i=1,...,N;j=1,...,M$. Recall, that functions $\phi j(x),j=0,...,M$ are called basis functions which should be chosen and are known. Then the regularized least squares problem takes the form

$$\min\omega 12\|r(\omega)\|22+\gamma 2\|\omega\|22=\min\omega 12\|A\omega-t\|22+\gamma 2\|\omega\|22.$$

To minimize the regularized squared errors we have to derive the normal equations. Similarly as the Fréchet derivative for the non-regularized

regression problem, we look for the ω where the gradient of 12||Aω−t||22+γ2‖ω‖22=12(Aω−t)T(Aω−t)+γ2ωTω vanishes. In other words, we consider

$$12\lim\|e\|\to 0(A(\omega+e)-t)T(A(\omega+e)-t)-(A\omega-t)T(A\omega-t)\|e\|2+\lim\|e\|\to 0\gamma 2(\omega+e)T(\omega+e)-\gamma 2\omega T\omega\|e\|2$$

We finally get

$$0=\lim\|e\|\to 0eT(ATA\omega-ATt)\|e\|2+\gamma eT\omega\|e\|2$$

The expression above means that the factor ATAω−ATt+γω must also be zero, or

$$(ATA+\gamma I)\omega=Att$$

6.5 PERCEPTRON

6.5.1 Introduction:

The perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The perceptron algorithm was invented in 1958 at the Cornell Aeronautical Laboratory by Frank Rosenblatt.

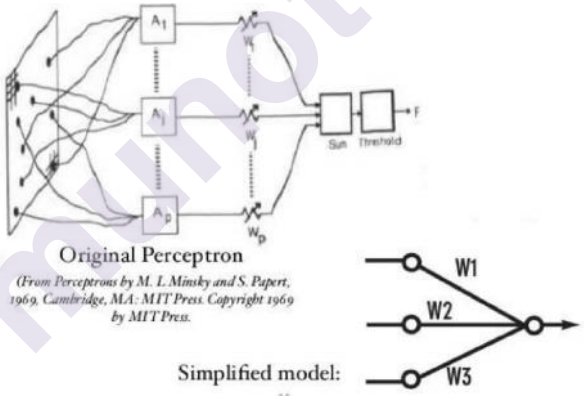


Figure5: Perceptron model

6.5.2 Types of Perceptron:

There are two types of Perceptrons: Single layer and Multilayer.

1. Single layer - Single layer perceptrons can learn only linearly separable patterns
2. Multilayer - Multilayer perceptrons or feedforward neural networks with two or more layers have the greater processing power

6.5.3 Single layer Perceptron:

It is the first and basic model of the artificial neural networks. It is also called the feed-forward neural network. Single layer perceptrons can learn

only linearly separable patterns The working of the single-layer perceptron (SLP) is based on the threshold transfer between the nodes. This is the simplest form of ANN and it is generally used in the linearly based cases for the machine learning problems.

6.5.3.1 Working of Single Layer Perceptron:

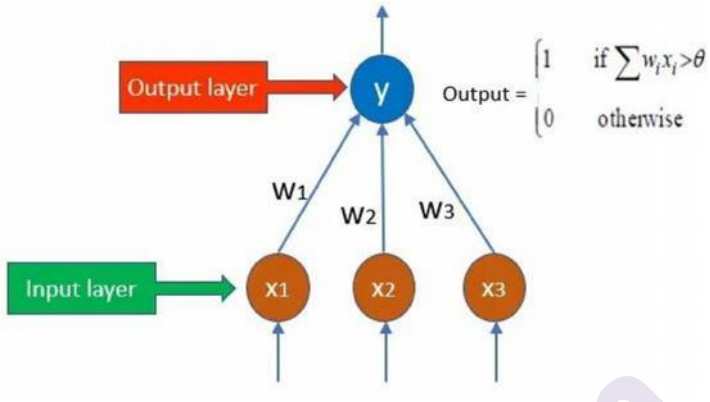


Figure 6: Single layer Perceptron

- In a single layer perceptron, the weights to each input node are assigned randomly since there is no a priori knowledge associated with the nodes.
- Now SLP sums all the weights which are inputted and if the sums are is above the threshold then the network is activated.
- If the calculated value is matched with the desired value, then the model is successful
- If it is not, then since there is no back-propagation technique involved in this the error needs to be calculated using the below formula and the weights need to be adjusted again.

6.5.3.1 Advantages:

1. Single Layer Perceptron is quite easy to set up and train.
2. The neural network model can be explicitly linked to statistical models
3. The SLP outputs a function which is a sigmoid and that sigmoid function can easily be linked to posterior probabilities.
4. We can interpret and input the output as well since the outputs are the weighted sum of inputs.

6.5.3.2 Disadvantages:

1. This neural network can represent only a limited set of functions.
2. The decision boundaries that are the threshold boundaries are only allowed to be hyperplanes.
3. This model only works for the linearly separable data.

6.5.4 Multi layer perceptron:

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation).

An MLP consists of at least three layers of nodes: 1) an input layer, 2) a hidden layer and 3) an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

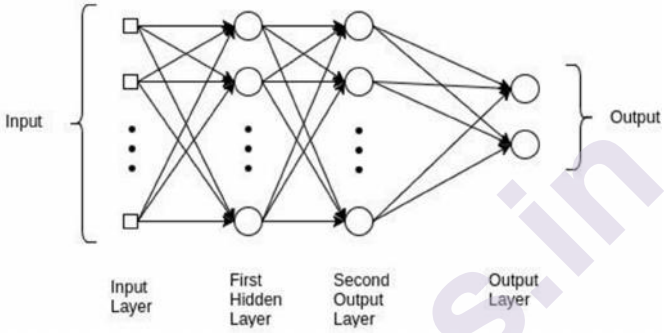


Figure 7: Multilayer Perceptron

Following are the building blocks of perceptron

1. Neurons
2. Activation
3. Networks of Neurons
4. Input or Visible Layers
5. Hidden Layers
6. Output Layer

6.5.4.1 Neurons:

The building block for neural networks is artificial neurons.

These are simple computational units that have weighted input signals and produce an output signal using an activation function.

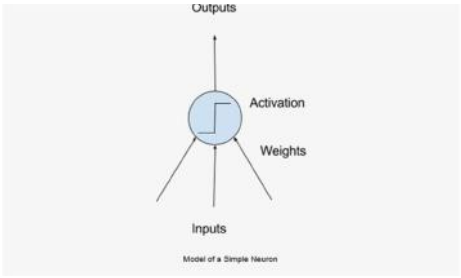


Figure 8: Artificial Neurons

You may be familiar with linear regression, in which case the weights on the inputs are very much like the coefficients used in a regression equation. Like linear regression, each neuron also has a bias which can be thought of as an input that always has the value 1.0 and it too must be weighted.

6.5.4.2 Activation:

The weighted inputs are summed and passed through an activation function, sometimes called a transfer function. Traditionally non-linear activation functions are used. This allows the network to combine the inputs in more complex ways and in turn provide a richer capability in the functions they can model. Non-linear functions like the logistic also called the sigmoid function were used that output a value between 0 and 1 with an s-shaped distribution, and the hyperbolic tangent function also called tanh that outputs the same distribution over the range -1 to +1. More recently the rectifier activation function has been shown to provide better results

6.5.4.3 Networks of Neurons:

Neurons are arranged into networks of neurons. A row of neurons is called a layer and one network can have multiple layers. The architecture of the neurons in the network is often called the network topology.

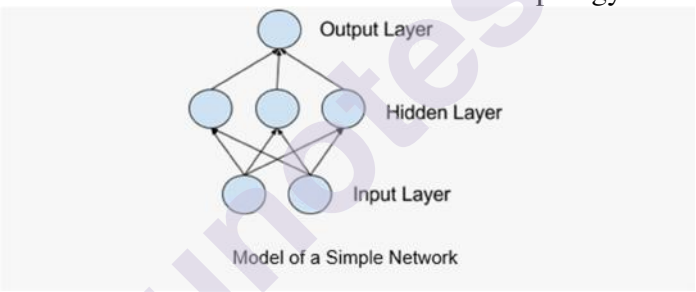


Figure 9: Networks of Neurons

6.5.4.4 Input or Visible Layers:

The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset. These are not neurons as described above, but simply pass the input value though to the next layer.

6.5.4.5 Hidden Layers:

Layers after the input layer are called hidden layers because that are not directly exposed to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value.

Given increases in computing power and efficient libraries, very deep neural networks can be constructed. Deep learning can refer to having many hidden layers in your neural network. They are deep because they would have been unimaginably slow to train historically, but may take seconds or minutes to train using modern techniques and hardware.

6.5.4.6 Output Layer:

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem.

The choice of activation function in he output layer is strongly constrained by the type of problem that you are modeling.

For example:

1. A regression problem may have a single output neuron and the neuron may have no activation function.
2. A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for the class 1.
3. A multi-class classification problem may have multiple neurons in the output layer, one for each class (e.g. three neurons for the three classes in the famous iris flowers classification problem).

6.5.4.7 Stochastic Gradient Descent:

The classical and still preferred training algorithm for neural networks is called stochastic gradient descent. This is where one row of data is exposed to the network at a time as input. The network processes the input upward activating neurons as it goes to finally produce an output value. This is called a forward pass on the network. It is the type of pass that is also used after the network is trained in order to make predictions on new data.

The process is repeated for all of the examples in your training data. One round of updating the network for the entire training dataset is called an epoch. A network may be trained for tens, hundreds or many thousands of epochs.

6.5.4.8 Weight Updates:

The weights in the network can be updated from the errors calculated for each training example and this is called online learning. It can result in fast but also chaotic changes to the network.

Alternatively, the errors can be saved up across all of the training examples and the network can be updated at the end. This is called batch learning and is often more stable.

Typically, because datasets are so large and because of computational efficiencies, the size of the batch, the number of examples the network is shown before an update is often reduced to a small number, such as tens or hundreds of examples.

6.5.4.9 Prediction:
Once a neural network has been trained it can be used to make predictions. You can make predictions on test or validation data in order to estimate the skill of the model on unseen data. You can also deploy it operationally and use it to make predictions continuously.

An MLP is a network of simple neurons called perceptrons. The basic concept of a single perceptron was introduced by Rosenblatt in 1958. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then possibly putting the output through some nonlinear activation function. Mathematically this can be written as

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

where W denotes the vector of weights, X is the vector of inputs, b is the bias and φ is the activation function. A signal-flow graph of this operation is shown in Figure given below

6.5.4.10 Advantages:

- Can be applied to complex non-linear problem
- Works well with large input data
- Provides quick prediction after training
- The same accuracy can be achieved even with smaller data

6.5.4.11 Disadvantages:

- It is not known to what extent each independent variable is affected by the dependent variable.
- Computations are difficult and time consuming
- The proper functioning of the model depends on the quality of training

SUMMARY

This chapter gives a breakdown of linear regression model. Regression analysis is one of the first modeling techniques to learn as a data scientist. It can helpful when forecasting continuous values, e.g., sales, temperature. There are quite a few formulas to learn but they are necessary to understand when we run linear regression models. As you saw above there are many ways to check the assumptions of linear regression like Least Squares method, Multivariate Linear Regression, Regularized Regression, Using Least Square regression for Classification. Also we saw Perceptron that classifies patterns and groups by finding the linear separation between different objects and patterns that are received through numeric or visual input.

UNIT END QUESTIONS

- **Least Square method:**

1. Explain least square method and its limitations.
2. Explain the types of least square method.
3. What is the difference between Linear and non- linear least square method?
4. Solve the following using least square method.

Consider the time series data given below:

x_i	8	3	2	10	11	3	6	5	6	8
y_i	4	12	1	12	9	4	9	6	1	14

Use the least square method to determine the equation of line of best fit for the data. Then plot the line.

- **Multivariate Linear Regression:**

1. Explain Multivariate linear regression with an example.
2. What are the steps for Multivariate Linear regression?

- **Regularized Regression:**

1. Explain Regularized regression.
2. What are the types of regularized regression?
3. Give comparison of Lasso and Ridge with linear regrrssion model.

- **Using Least Square Regression for classification:**

1. Explain the use of least square regression for classification.

- **Perceptron**

1. Explain perceptron algorithm.
2. Explain the types of perceptron algorithms
3. Explain the working of single layer perceptron.
4. Explain Single layer perceptron with advantages and disadvantages.
5. Explain Multilayer perceptron with advantages and disadvantages.

REFERENCE

- Bevans, R. (2020, October 26). Simple Linear Regression: An Easy Introduction & Examples. Retrieved from https://www.scribbr.com/statistics/simple-linear-regression/
- Bevans, R. (2020, October 26). Multiple Linear Regression: A Quick and Simple Guide. Retrieved from https://www.scribbr.com/statistics/multiple-linear-regression/
- International Encyclopedia of the Social Sciences. . Encyclopedia.com. 16 Oct. 2020 . (2020, November 27). Retrieved from https://www.encyclopedia.com/social-sciences/applied-and-social-sciences-magazines/ordinary-least-squares-regression

- Jcf2d, W. B. (n.d.). University of Virginia Library Research Data Services Sciences. Retrieved from https://data.library.virginia.edu/understanding-q-q-plots/
- Stephanie. (2020, September 16). Q Q Plots: Simple Definition & Example. Retrieved from https://www.statisticshowto.com/q-q-plots/
- Assumptions of Linear Regression. (2020, June 22). Retrieved from https://www.statisticssolutions.com/assumptions-of-linear-regression/

*****

7:

SUPPORT VECTOR MACHINE

Unit Structure

7.0 OBJECTIVES

Objective: Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. Through this chapter we are exhibiting SVM model.

7.1 INTRODUCTION SUPPORT VECTOR MACHINES

7.1.1 Definition:

Support Vector Machine (**SVM**) is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. Mainly 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:

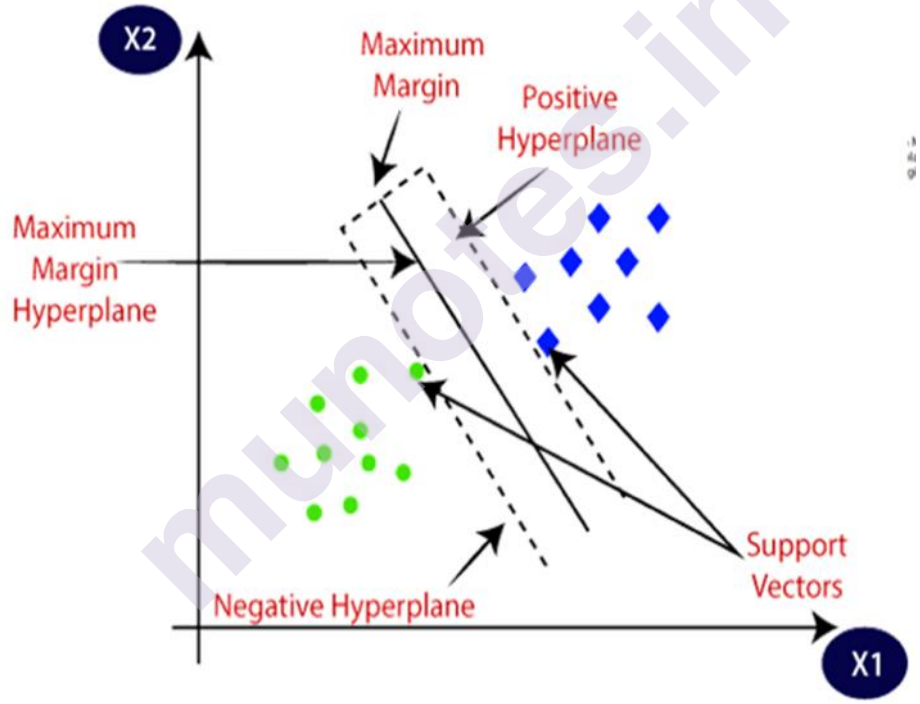


Figure 10: Decision boundary or hyperplane

7.1.2 Hyperplane and Support Vectors:

7.1.2.1 Hyperplane:

There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the **hyperplane** of SVM.

7.1.2.2 Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as **Support Vectors**. These vectors support the hyperplane, hence called a Support vector.

7.1.3 Types of SVM:

SVM can be of two types:

1. **Linear SVM:** 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.
2. **Non-linear SVM:** 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.

7.1.4 Working of SVM:

7.1.4.1 Linear SVM:

Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:

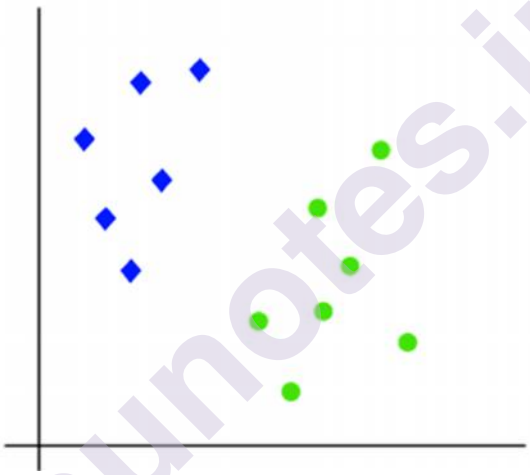


Figure 11: Linear SVM

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes

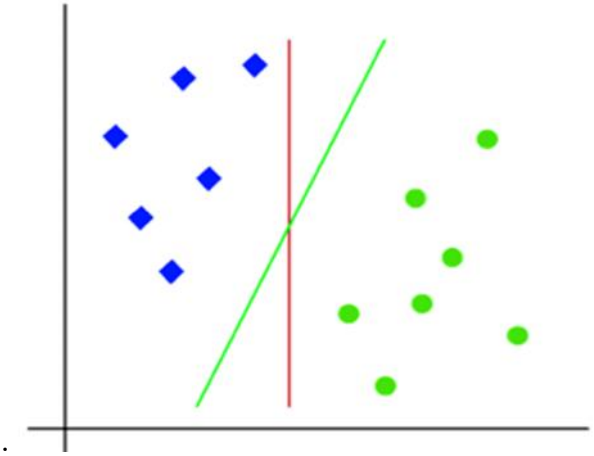


Figure 12: Separated classes

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called the margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the **optimal hyperplane**.

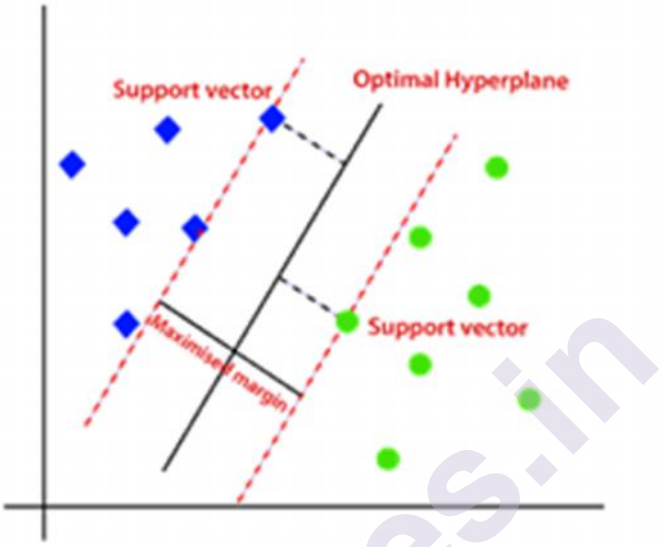


Figure 13: Optimal hyperplane

7.1.4.2 Non-linear SVM:

For non-linear data, we cannot separate by drawing a single straight line. Consider the below image:

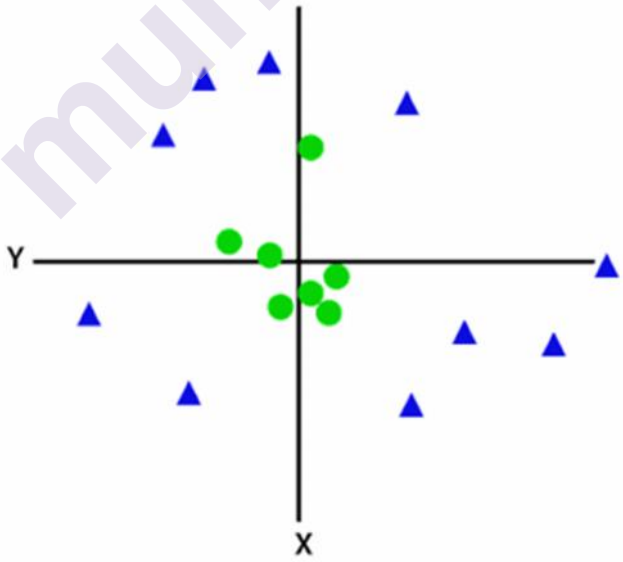


Figure 14: Non Linear SVM

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

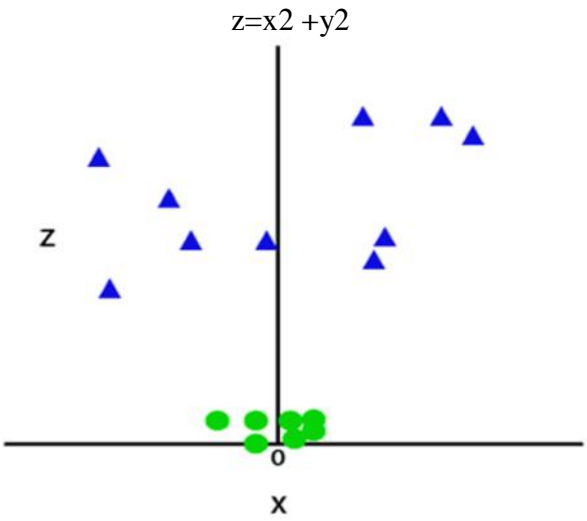


Figure 15: Third dimension z

So now, SVM will divide the datasets into classes in the following way.

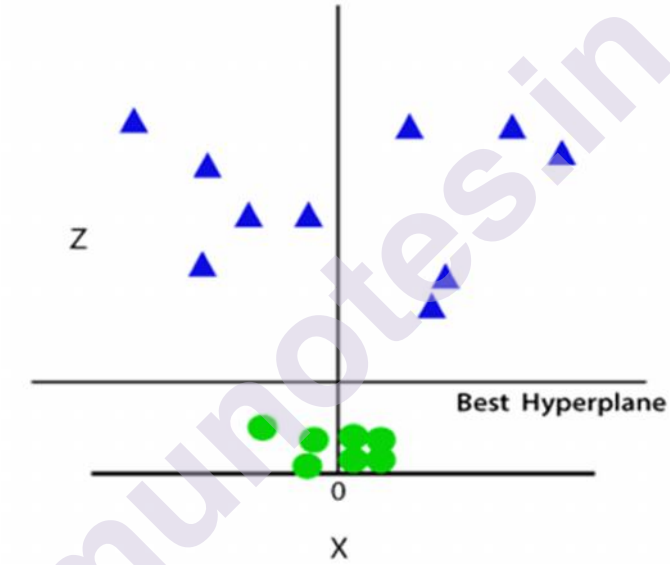


Figure 16: SVM divide dataset in classes

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

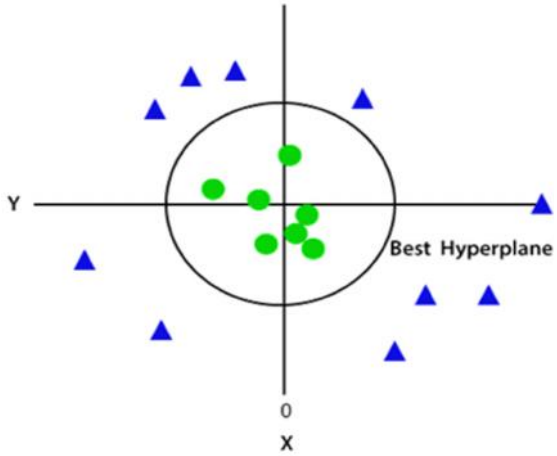


Figure 17: Best Hyperplane

Hence we get a circumference of radius 1 in case of non-linear data.

7.2 SOFT MARGIN SVM

7.2.1 What Soft Margin does is:

1. it tolerates a few dots to get misclassified
2. it tries to balance the trade-off between finding a line that maximizes the margin and minimizes the misclassification.

This idea is based on a simple premise: allow SVM to make a certain number of mistakes and keep the margin as wide as possible so that other points can still be classified correctly. This can be done simply by modifying the objective of SVM. The main idea behind the support vector classifier is to find a decision boundary with a maximum width that can classify the two classes. Maximum margin classifiers are super sensitive to outliers in the training data and that makes them pretty lame. Choosing a threshold that allows misclassifications is an example of the Bias-Variance tradeoff that plagues all the machine learning algorithms. When we allow some misclassifications (slack variables), the distance between the observations and the threshold is called a "soft margin".

This idea is based on a simple premise: allow SVM to make a certain number of mistakes and keep the margin as wide as possible so that other points can still be classified correctly. This can be done simply by modifying the objective of SVM.

7.2.2 Better soft margin:

We use cross-validation to determine how many misclassifications and observations to allow inside of the soft margin to get the best classification.

The name support vector classifier comes from the fact that the observations on the edge that helps us to draw the margin are called support vectors.

7.2.3 Degree of tolerance

How much tolerance we want to set when finding the decision boundary is an important hyper-parameter for the SVM.

7.2.4 Formulation

- Almost all real-world applications have data that is linearly inseparable.
- In some cases where the data is linearly separable, we might not want to choose a decision boundary that perfectly separates the data to avoid overfitting. For example, consider the following diagram:

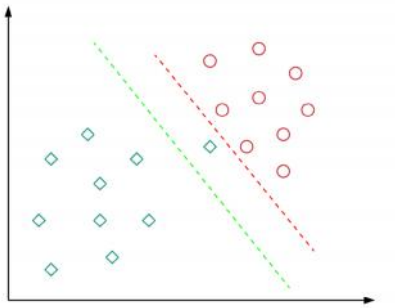


Figure 18: Better decision boundary

Here the red decision boundary perfectly separates all the training points. However, is it really a good idea to have a decision boundary with such less margin The green decision boundary has a wider margin that would allow it to generalize well on unseen data. In that sense, soft margin formulation would also help in avoiding the overfitting problem.

Let us see how we can modify our objective to achieve the desired behavior. In this new setting, we would aim to minimize the following objective:

$$L = \frac{1}{2}\|w\|^2 + C(\#\ of\ mistakes) \quad \text{…Equation 1}$$

Here, C is a hyperparameter that decides the trade-off between maximizing the margin and minimizing the mistakes. When C is small, classification mistakes are given less importance and focus is more on maximizing the margin, whereas when C is large, the focus is more on avoiding misclassification at the expense of keeping the margin small.

Let's see how this could be incorporated with the help of the following diagram.

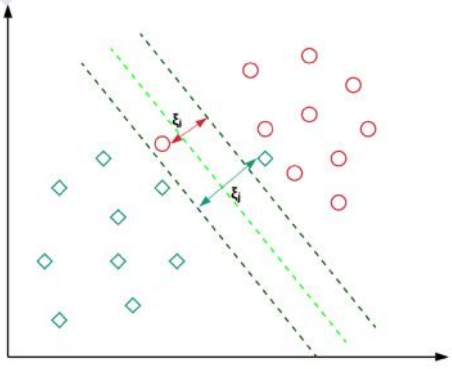


Figure 19: Avoiding Misclassification

Figure 19: The penalty incurred by data points for being on the wrong side of the decision boundary

The idea is: for every data point x_i, we introduce a slack variable ξ_i. The value of ξ_i is the distance of x_i from the corresponding class's margin if x_i is on the wrong side of the margin, otherwise zero. Thus the points that are far away from the margin on the wrong side would get more penalty.

With this idea, each data point x_i needs to satisfy the following constraint:

$$y_i(\vec{w}.\vec{x}_i + b) \geq 1 - \xi_i$$ ……..Equation 2

Here, the left-hand side of the inequality could be thought of as the confidence of classification.

7.3 LINEAR CLASSIFIERS

Solving classification tasks are based on linear models. What this means is that they aim at dividing the feature space into a collection of regions labeled according to the values the target can take, where the decision boundaries between those regions are linear: they are lines in 2D, planes in 3D, and hyperplanes with more features.

Some of linear classifier techniques include:

1. Regularized Discriminant Analysis
2. Quadratic Discriminant Analysis
3. Linear Discriminant Analysis
4. Logistic Regression

7.3.1 Regularized Discriminant Analysis:

Like linear models for regression can be regularized to improve accuracy, so can linear classifiers. One can introduce a shrinking parameter α that shrinks the separate covariance matrices of QDA towards a common LDA matrix:

$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1 - \alpha)\hat{\Sigma}$$

The shrinkage parameter can take values from 0 (LDA) to 1 (QDA) and any value in between is a compromise between the two approaches. The best value of α can be chosen based on cross-validation. To do this in Python, we need to pass the shrinkage argument to the LDA function, as well as specify the computation algorithm to be least squares, as other computation methods do not support shrinkage.

7.3.2 Linear Discriminant Analysis:

The method to be discussed is the Linear Discriminant Analysis (LDA). It assumes that the joint density of all features, conditional on the target's

class, is a multivariate Gaussian. This means that the density P of the features X, given the target y is in class k, are assumed to be given by

$$P(X|y=k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t \Sigma_k^{-1}(X-\mu_k)\right)$$

where d is the number of features, μ is a mean vector, and Σ_k the covariance matrix of the Gaussian density for class k.

The decision boundary between two classes, say k and l, is the hyperplane on which the probability of belonging to either class is the same. This implies that, on this hyperplane, the difference between the two densities should be zero.

7.3.3 Quadratic Discriminant Analysis:

This performs a quadratic discriminant analysis (QDA). QDA is closely related to linear discriminant analysis (LDA), where it is assumed that the measurements are normally distributed. Unlike LDA however, in QDA there is no assumption that the covariance of each of the classes is identical. To estimate the parameters required in quadratic discrimination more computation and data is required than in the case of linear discrimination. If there is not a great difference in the group covariance matrices, then the latter will perform as well as quadratic discrimination. Quadratic Discrimination is the general form of Bayesian discrimination.

Discriminant analysis is used to determine which variables discriminate between two or more naturally occurring groups. For example, an educational researcher may want to investigate which variables discriminate between high school graduates who decide (1) to go to college, (2) NOT to go to college. For that purpose the researcher could collect data on numerous variables prior to students' graduation. After graduation, most students will naturally fall into one of the two categories. Discriminant Analysis could then be used to determine which variable(s) are the best predictors of students' subsequent educational choice. Computationally, discriminant function analysis is very similar to analysis of variance (ANOVA). For example, suppose the same student graduation scenario. We could have measured students' stated intention to continue on to college one year prior to graduation. If the means for the two groups (those who actually went to college and those who did not) are different, then we can say that the intention to attend college as stated one year prior to graduation allows us to discriminate between those who are and are not college bound (and this information may be used by career counselors to provide the appropriate guidance to the respective students). The basic idea underlying discriminant analysis is to determine whether groups differ with regard to the mean of a variable, and then to use that variable to predict group membership (e.g. of new cases).

Discriminant Analysis may be used for two objectives: either we want to assess the adequacy of classification, given the group memberships of the objects under study; or we wish to assign objects to one of a number of (known) groups of objects. Discriminant Analysis may thus have a descriptive or a predictive objective. In both cases, some group assignments must be known before carrying out the Discriminant Analysis. Such group assignments, or labeling, may be arrived at in any way. Hence Discriminant Analysis can be employed as a useful complement to Cluster Analysis (in order to judge the results of the latter) or Principal Components Analysis.

7.3.4 Logistic Regression:

Another approach to linear classification is the logistic regression model, which, despite its name, is a classification rather than a regression method.

Logistic regression models the probabilities of an observation belonging to each of the K classes via linear functions, ensuring these probabilities sum up to one and stay in the (0, 1) range. The model is specified in terms of K-1 log-odds ratios, with an arbitrary class chosen as reference class (in this example it is the last class, K). Consequently, the difference between log-probabilities of belonging to a given class and to the reference class is modeled linearly as

$$\log \frac{\Pr(G=1|X=x)}{\Pr(G=K|X=x)} = \beta_{10} + \beta_1^T x$$

$$\log \frac{\Pr(G=2|X=x)}{\Pr(G=K|X=x)} = \beta_{20} + \beta_2^T x$$

$$\vdots$$

$$\log \frac{\Pr(G=K-1|X=x)}{\Pr(G=K|X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x$$

where G stands for the true, observed class. From here, the probabilities of an observation belonging to each of the classes can be calculated as

$$\Pr(G=k|X=x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)}, \; k = 1, \ldots, K-1,$$

$$\Pr(G=K|X=x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)},$$

which clearly shows that all class probabilities sum up to one.

Logistic regression models are typically estimated by maximum likelihood. Just like linear models for regression can be regularized to improve accuracy, so can logistic regression.

7.4 KERNEL FUNCTIONS IN NON-LINEAR CLASSIFICATION

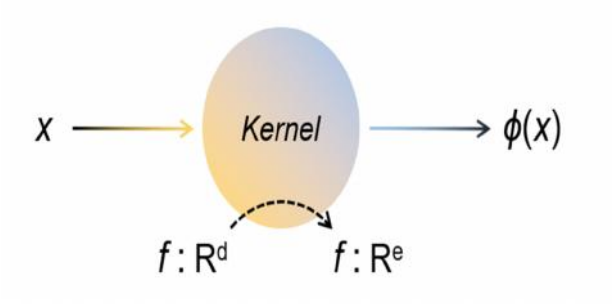


Figure 20: Kernal Function

Once the data points are non-linear separable in their original feature space, the linear classifier may fail to determine where the decision boundary is. However, mapping the original feature space ($x \in \mathbb{R}^d$) into the higher dimensional feature space ($\phi(x) \in \mathbb{R}^e$, e>d) can help to resurrect the linear classifier to do the job correctly.

$$f(x;\theta,\theta_0) \neq sign(\theta \cdot x + \theta_0)$$

$$\downarrow$$

$$f(\phi(x);\theta,\theta_0) = sign(\theta \cdot \phi(x) + \theta_0)$$

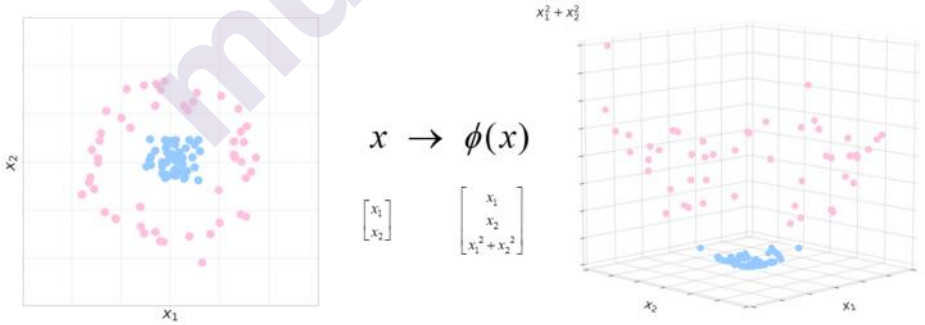


Figure 21: Mapping data points with 2-D feature vectors into 3-D feature vectors

Figure illustrates the concepts of classifying data points through feature mapping. Originally, the data points with the feature vectors $x = [x_1, x_2]$ in the 2-D space have the concentrically circular distribution. It is impossible to use a linear classifier to distinguish the decision boundary.

Nonetheless, by incorporating a certain mapping function $\phi(x)$, the feature vectors can be transformed into 3-D feature space. The new data points

with 3-D feature vectors $\phi(x) = [x_1, x_2, (x_1^2+x_2^2)]$ can now be using the linear classifier to determine the decision boundary hyperplane. This is the power of feature mapping that can allow us to deal with the more complex data distribution pattern with more expressive ability. However, the drawbacks of using $\phi(x)$ directly are that It is sometimes hard to explicitly construct a $\phi(x)$ directly. Increase computational power quickly with the increased feature dimensions. But the kernel functions can provide an efficient way to solve this.

7.4.1 Kernel Functions:

The idea of kernel functions is to take the inner products between two feature vectors, and evaluate inner products is not computationally costly. We can then exploit only the result of the inner products in our algorithms. For example, if we want to have the $\phi(x)$ as follows,

$$\phi(x) = \left[x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1 \right]$$

The kernel function is take the inner products between two feature vectors as follows,

$$K(x,x') = \phi(x) \cdot \phi(x') = (x \cdot x' + 1)^2$$

As a result, the form of the kernel functions will be easier for us to construct than directly using the mapping functions in the higher feature dimensions.

7.4.2 Kernel Composition Rules:

There are several kernel compositions rules that can be used to construct more complex kernel functions.

1. if $c \in \mathbb{R}$

   then $K(x,x') = c$ is a kernel function.

2. if $f : \mathbb{R}^d \to \mathbb{R}$, and $K(x,x')$ is a kernel function,

   then $\tilde{K}(x,x') = f(x)K(x,x')f(x')$ is a kernel function.

3. if $K_1(x,x')$ and $K_2(x,x')$ are kernel functions,

   then $\tilde{K}(x,x') = K_1(x,x') + K_2(x,x')$ is a kernel function.

4. if $K_1(x,x')$ and $K_2(x,x')$ are kernel functions,

   then $\tilde{K}(x,x') = K_1(x,x') \times K_2(x,x')$ is a kernel function.

7.4.3 Radial Basis Kernel:

The kernel functions can even empower the feature vectors to be infinite dimensional. One of the common kernel functions is the radial basis kernel. The definition is as follows.

$$K(x,x') = e^{-\gamma \|x-x'\|^2}$$

Because the exponential can be expanded to the infinite power series, the radial basis kernel gives much more expressiveness to the feature mapping. The following is the proof of the radial basis kernel that is a kernel function.

$$e^{-\gamma \|x-x'\|^2}$$

$$\Rightarrow e^{-\gamma (\|x\|^2 - 2xx' + \|x'\|^2)}$$

$$\Rightarrow e^{-\gamma \|x\|^2} \times e^{2\gamma xx'} \times e^{-\gamma \|x'\|^2}$$

$$\Rightarrow f(x)K(x,x')f(x')$$

$$\Rightarrow \tilde{K}(x,x')$$

$$\begin{cases} f(x) = e^{-\gamma \|x\|^2} \\ K(x,x') = e^{2\gamma xx'} = 2\gamma \sum_{n=0}^{\infty} \frac{(xx')^n}{n!} \text{ is a kernel function} \end{cases}$$

Kernel Perceptron Algorithm:

Recalling the perceptron algorithm here, the perceptron algorithm updates $\theta = \theta + y^{(j)} x^{(j)}$ once a data point is misclassified. In the other word, the θ can be expressed alternatively as follows.

$$\theta = \sum_{j=0}^{m} \alpha_j y^{(j)} x^{(j)}$$

where α_j is the number of mistakes the perceptron made on the j-th data point. If it is in the mapping feature space, the θ can be expressed as follows.

$$\theta = \sum_{j=0}^{m} \alpha_j y^{(j)} \phi(x^{(j)})$$

$$\Rightarrow \theta \cdot \phi(x^{(i)}) = \sum_{j=0}^{m} \alpha_j y^{(j)} K(x^{(j)}, x^{(i)})$$

SUMMARY

We have two choices, we can either use the sci-kit learn library to import the SVM model and use it directly or we can write our model from scratch. Instead, using a library from **sklearn.SVM** module which includes Support Vector Machine algorithms will be much easier in implementation as well as to tune the parameters. You can try for hand-on with the SVM algorithms including classification and regression problems.

SVM Use Cases

Some use-cases of SVM are as below.

- Face Detection
- Bioinformatics
- Classification of Images
- Remote Homology Detection
- Handwriting Detection
- Generalized Predictive Control
- Text and Hypertext Categorization

UNIT END QUESTION

- **Support vector Machine**

1. Explain support vector machines with example.
2. What are the types of Support vector machines?
3. What are Hyperplane and Support vectors in the SVM algorithms?
4. Explain the working of SVM.
5. Why SVM is an example of a large margin classifier?
6. What is a kernel in SVM? Why do we use kernels in SVM?
7. Explain the key terminologies of Support Vector Machine.
8. Define support vector machine (SVM) and further explain the maximum margin linear separators concept.

- **Soft Margin SVM**

1. What is soft margin SVM?
2. Explain the working of Soft margin SVM.
3. Explain the formulation of soft margin SVM?

- **Obtaining probabilities from linear classifiers:**

1. How to obtain probabilities from linear classifiers using logistic regression?

- **Kernal methods for non-linearity**

1. Explain Kernel methods for non-linearity

2. What are the limitations of the kernel method?
3. Explain optimization problem for SVM with non-linear kernal

REFERENCE

- "**A First Course in Machine Learning** by Simon Rogers and Mark Girolami
- Machine Learning Algorithms: A reference guide to popular algorithms for data science and machine learning July 2017 Author: Giuseppe Bonaccorso Publisher: Packt Publishing ISBN:978-1-78588-962-2.
- Deep Learning Ian Goodfellow, Yoshua Bengio, and Aaron Courville
- Hands-On Machine Learning with Scikit-Learn and TensorFlow
- Concepts, Tools, and Techniques to Build Intelligent Systems (2nd edition)

*****

UNIT IV

8

DISTANCE BASED MODELS

Unit Structure

8.0 OBJECTIVES

This chapter would make you understand the following concepts:

- Meaning of distance based model
- Distance Computation methods
- Concepts of different clustering algorithms
- Application of clustering algorithms to solve problems

8.1 INTRODUCTION TO ALGEBRIC MODEL

In this section, we consider models that define similarity by considering the geometry of the instance space. In Algebric models, features could be described as points in two 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). In algebraic 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**.

8.1.1 Distance based models:

In the first section we have seen the concept of Algebric models. Second class or type of the algebric models is known as the Distance-based models. Geometry of data is used to design the Distance-based models. Working of the distance-based models is based on the concept of distance. With respect to machine learning, the concept of distance is not based on just 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 if we are travelling by plane from one city to other then the plane will cover less distance physically as compared to travelling by train. The reason for this is the unrestricted route for a plane. In the same manner for 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.

8.1.2 Distance Calculation Methods:

The following distance metrics are commonly used to calculate the distance.

1. Euclidean distance:
For geometrical problems Euclidean distance is used as the standard metric. It is simply the ordinary distance between two points. Euclidean distance is mainly extensively used in clustering problems. In K-means algorithms by default Euclidean distance is used as distance measure. The Euclidean distance is calculated by taking the root of square differences between the coordinates of a pair of objects(x_1,y_1) and (x_2,y_2) as shown in equation given below

$$\text{Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad .$$

2. Manhattan distance:
Manhattan distance is a distance metric that calculates the absolute differences between coordinates of pair of data objects as shown in equation given below:

$$\text{Distance} = |(x_1 - x_2)| + |(y_1 - y_2)|$$

8.2 NEIGHBOURS AND EXEMPLARS

In the Distance based models distance is applied through the concept of neighbours and exemplars. Neighbours are points in proximity with respect to the distance measure expressed through exemplars. Exemplars are either centroids that find a centre of mass according to a chosen distance metric or medoids that find the most centrally located data point. The most commonly used centroid is the arithmetic mean, which minimises squared Euclidean distance to all other points.

- The **centroid** represents the geometric centre of a plane figure, i.e., the arithmetic mean position of all the points in the figure from the centroid point. This definition extends to any object in *n*-dimensional space: its centroid is the mean position of all the points.
- **Medoids** are similar in concept to means or centroids. Medoids are most commonly used on data when a mean or centroid cannot be defined. They are used in contexts where the centroid is not representative of the dataset, such as in image data.

8.3 NEAREST NEIGHBOURS CLASSIFICATION

Examples of distance-based models include the nearest-neighbour models, which use the training data as exemplars – for example, in classification.

8.3.1 What is Nearest Neighbour?:

Nearest neighbouris a method in machine learning method that aims at labelling previously unseen query objects while distinguishing two or more destination classes. As any classifier, in general, it requires some training data with given labels and, thus, is an instance of supervised learning.

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

- **Lazy learning algorithm**: KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm**: KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

8.3.2 Working of KNN Algorithm:

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the

training set. We can understand its working with the help of following steps

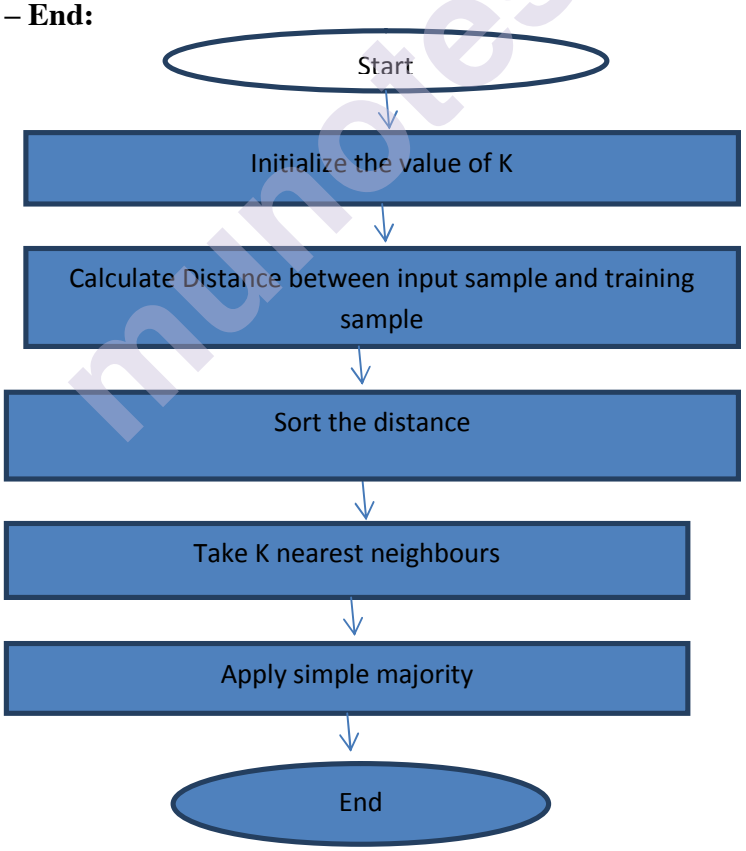
Step 1 For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 For each point in the test data do the following –

- **3.1** − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean or Manhattan distance. The most commonly used method to calculate distance is Euclidean.
- **3.2** − Now, based on the distance value, sort them in ascending order.
- **3.3** − Next, it will choose the top K rows from the sorted array.
- **3.4** − Now, it will assign a class to the test point based on most frequent class of these rows.

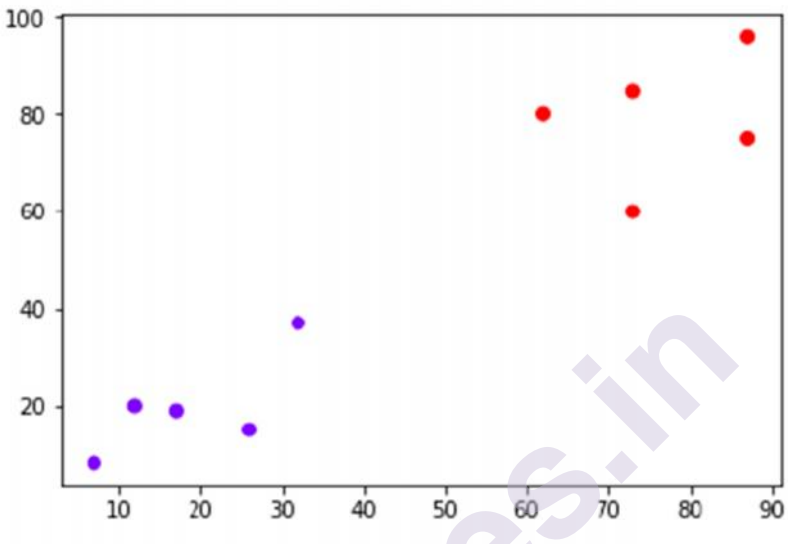
Step 4 – End:



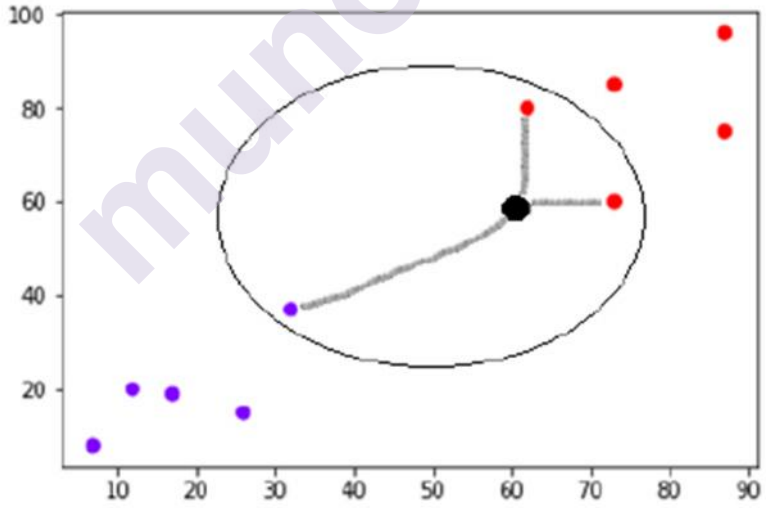
8.3.3 Examples of K-NN Algorithm:

Example 1:
The following is an example to understand the concept of K and working of KNN algorithm

Suppose we have a dataset which can be plotted as follows:



Now, we need to classify new data point (60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next figure



We can see in the above figure the three nearest neighbours of the data point with black dot. Among those three, two of them lie in Red class hence the black dot will also be assigned in red class.

Example 2:
We have collected data from the sample survey. This data represents the two attributes as rating of acting of actors in that movie and other is rating

of story line of that movie. The rating scale is used from 1(excellent) to 7 (poor). Now we need to classify whether a given movie is goodor not. Now we want to check whether new movie with rating as X1 = 3 and X2 = 7 is good or not. Here are four training samples:

X1 = Rating of Acting skills of movie actors	X2 = Rating of story line of movie	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Step 1 : Initialize and Define k.

Lets say, k = 3

NOTE : Always choose k as an odd number if the number of attributes is evento avoid a tie in the class prediction

Step2 : Compute the distance between input sample and training sample

- Co-ordinate of the input sample is (3,7).
- Instead of calculating the Euclidean distance, we calculate the Squared Euclidean distance.

X1 = Rating of Acting skills of movie actors	X2 = Rating of story line of movie	Squared Euclidean distance
7	7	$(7-3)^2 + (7-7)^2 =$ **16**
7	4	$(7-3)^2 + (4-7)^2 =$ **25**
3	4	$(3-3)^2 + (4-7)^2 =$ **09**

Step 3: Sort the distance and determine the nearest neighbours based of the K^{th} minimum distance :

X1 = Rating of Acting skills of movie actors	X2 = Rating of story line of movie	Squared Euclidean distance	Rank minimum distance	Is it included in 3-Nearest Neighbour?
7	7	16	**3**	**Yes**
7	4	25	**4**	**No**
3	4	09	**1**	**Yes**
1	4	13	**2**	**Yes**

Step 4 : Take 3-Nearest Neighbours:

Gather the category Y of the nearest neighbours.

X1 = Rating of Acting skills of movie actors	X2 = Rating of story line of movie	Squared Euclidean distance	Rank minimum distance	Is it included in 3-Nearest Neighbour?	Y = Category of the nearest neighbour
7	7	16	3	**Yes**	**Bad**
7	4	25	4	**No**	-
3	4	09	1	**Yes**	**Good**
1	4	13	2	**Yes**	**Good**

Step 5: Apply simple majority

- Use simple majority of the category of the nearest neighbours as the prediction value of the query instance.
- We have 2 "good" and 1 "bad". Thus we conclude that the new moviewith X1 = 3 and X2 = 7 is included in the "good" category.

8.4 K- MEANS CLUSTERING

To solve the wellknown clustering problem K-means is used, which is one of the simplest unsupervised learning algorithms. Given data set is classified assuming some prior number of clusters through a simple and easy procedure. In k- means clustering for each cluster one centroid is defined. Total there are k centroids. The centroids should be defined in a tricky way because result differs based on the location of centroids. To get the better results we need to place the centroids far away from each other as much as possible. Next, each point from the given data set is stored in a group with closest centroid. This process is repeated for all the points. The first step is finished when all points are grouped. In the next step new k centroids are calculated again from the result of the earlier step. After finding these new k centroids, a new grouping is done for the data points and closest new centroids. This process is done iteratively. The process is repeated unless and until no data point moves from one group to another. The aim of this algorithm is to minimize an objective function such as sum of a squared error function. The objective function is defined as follows:

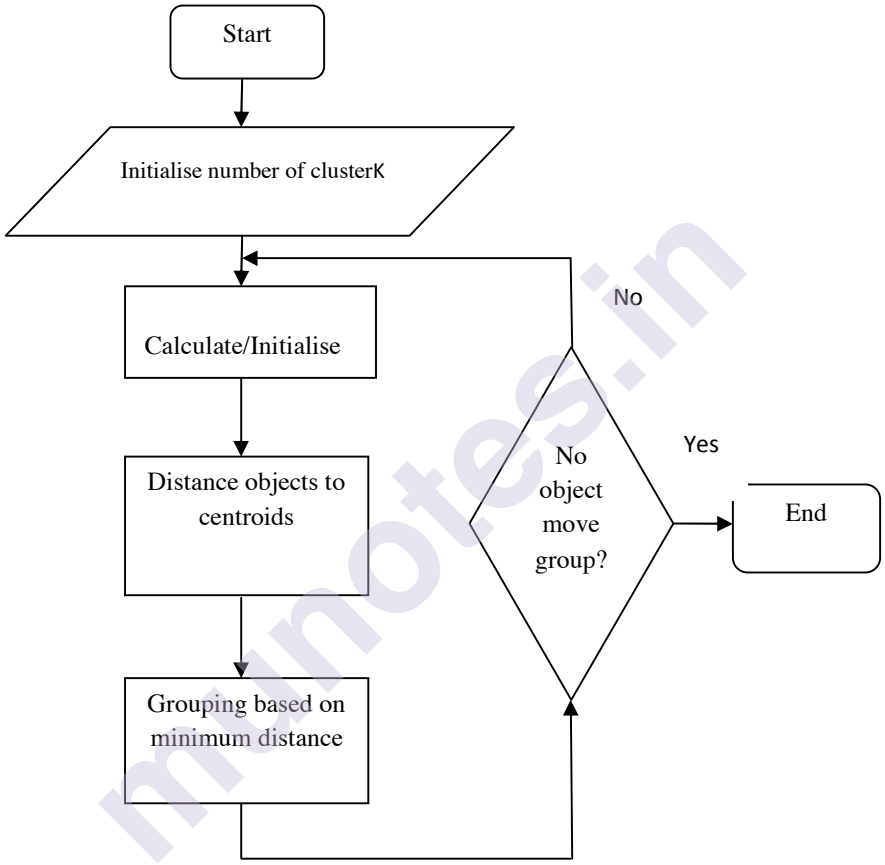
$$J = \sum_{j=1}^{k} \sum_{i=1}^{x} \| x_i^j - C_j \|^2$$

Here $\| x_i^j - C_j \|^2$shows the selected distance measure between a data point $x_i^j$and the cluster centre C_j. It is a representation of the distance of the n data points from their respective cluster centers.

8.4.1 Working of K-means Algorithm:

The algorithm is comprises of the following steps:

1. Identify the K centroids for the given data points that we want to cluster.
2. Store each data point in the group that has the nearest centroid.
3. When all data points have been stored, redefine the K centroids.
4. Repeat Steps 2 and 3 until the no data points move from one group to another. The result of this process is the clusters from which the metric to be minimized can be calculated.



8.4.3 Examples ofK-means Algorithm:

Example 1:

Given {2, 4, 10, 12, 3, 20, 30, 11, 25}. Assume number of clusters i.e. K = 2

Solution:

Randomly assign means: $m_1 = 3, m_2 = 4$

The numbers which are close to mean $m_1 = 3$ are grouped into cluster k_1 and others in k_2.

Again calculate new mean for new cluster group.

$K_1 = (2, 3\}, k_2 = \{4, 10, 12, 20, 30, 11, 25\}$ $m_1 = 2.5, m_2 = 16$

$K_1 = (2, 3, 4\}, k_2 = \{10, 12, 20, 30, 11, 25\}$ $m_1 = 3, m_2 = 18$

$K_1 = (2, 3, 4, 10\}, k_2 = \{12, 20, 30, 11, 25\}$ $m_1 = 4.75, m_2 = 19.6$

$K_1 = (2, 3, 4, 10, 11, 12\}, k_2 = \{20, 30, 25\}$ $m_1 = 7, m_2 = 25$

Final clusters

$K_1 = (2, 3, 4, 10, 11, 12\}, k_2 = \{20, 30, 25\}$

Example 2:
Given {10, 4, 2, 12, 3, 20, 30, 11, 25, 31} Assume number of clusters i.e. K = 2

Solution:
Randomly assign alternative values to each cluster
$K_1 = (10, 2, 3, 30, 25\}, k_2 = \{4, 12, 20, 11, 31\}$ $m_1 = 14, m_2 = 15.6$

Re assign
$K_1 = (2, 3, 4, 10, 11, 12\}, k_2 = \{20, 25, 30, 31\}$ $m_1 = 7, m_2 = 26.5$

Re assign
$K_1 = (2, 3, 4, 10, 11, 12\}, k_2 = \{20, 25, 30, 31\}$ $m_1 = 7, m_2 = 26.5$

Final clusters
$K_1 = (2, 3, 4, 10, 11, 12\}, k_2 = \{20, 25, 30, 31\}$

Example 3:
Let's assume that we have 4 types of items and each item has 2 attributes or features. We need to group these items in to k = 2 groups of items based on the two features.

Object	Attribute 1(x) Number of parts	Attribute 2(y) Colour code
Item 1	1	1
Item 2	2	1
Item 3	4	3
Item 4	5	4

Solution :

Initial value of centroid:
Suppose we use item 1 and 2 as the first centroids, $c_1 = (1, 1)$ and $c_2 = (2, 1)$ The distance of item 1 = (1, 1) to $c_1 = (1, 1)$ and with $c_2 = (2, 1)$ is calculated as,

$$D = \sqrt{(1-1)^2 + (1-1)^2} = 0$$

$$D = \sqrt{(1-2)^2 + (1-1)^2} = 1$$

The distance of item 2 = (2, 1) to c_1 = (1, 1) and with c_2 = (2, 1) is calculated as,

$$D = \sqrt{(2-1)^2 + (1-1)^2} = 1$$

$$D = \sqrt{(2-2)^2 + (1-1)^2} = 0$$

The distance of item 3 = (4, 3) to c_1 = (1, 1) and with c_2 = (2, 1) is calculated as,

$$D = \sqrt{(4-1)^2 + (3-1)^2} = 3.61$$

$$D = \sqrt{(4-2)^2 + (3-1)^2} = 2.83$$

The distance of item 4 = (5, 4) to c_1 = (1, 1) and with c_2 = (2, 1) is calculated as,

$$D = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$D = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Objects-centroids distance:

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \begin{matrix} c_1 = (1,1) & \text{group 1} \\ c_2 = (2,1) & \text{group 2} \end{matrix}$$

To find the cluster of each item we consider the minimum Euclidian distance between group1 and group 2.

From the above object centroid distance matrix we can see,

- Item 1 has minimum distance for group1, so we cluster item 1 in group 1.
- Item 2 has minimum distance for group 2, so we cluster item 2 in group 2.
- Item 3 has minimum distance for group 2, so we cluster item 3 in group 2.
- Item 4 has minimum distance for group 2, so we cluster item 4 in group 2.

Object Clustering:

$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Iteration 1 : Determine centroids:

C_1 has only one member thus c_1 = (1, 1) remains same.

$$C_2 = (2 + 4 + 5/3, 1 + 3 + 4/3) = (11/3, 8/3)$$

The distance of item 1 = (1, 1) to c_1 = (1, 1) and with c_2 = (11/3, 8/3) is calculated as,

$$D = \sqrt{(1-1)^2 + (1-1)^2} = 0$$

$$D = \sqrt{(1-11/3)^2 + (1-8/3)^2} = 3.41$$

The distance of item 2 = (2, 1) to c_1 = (1, 1) and with c_2 = (11/3, 8/3) is calculated as,

$$D = \sqrt{(2-1)^2 + (1-1)^2} = 1$$

$$D = \sqrt{(2-11/3)^2 + (1-8/3)^2} = 2.36$$

The distance of item 3 = (4, 3) to c_1 = (1, 1) and with c_2 = (2, 1) is calculated as,

$$D = \sqrt{(4-1)^2 + (3-1)^2} = 3.61$$

$$D = \sqrt{(4-11/3)^2 + (3-8/3)^2} = 0.47$$

The distance of item 4 = (5, 4) to c_1 = (1, 1) and with c_2 = (2, 1) is calculated as,

$$D = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$D = \sqrt{(5-11/3)^2 + (4-8/3)^2} = 1.89$$

Objects-centroids distance

$$D^2 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.41 & 2.36 & 0.47 & 1.89 \end{bmatrix} \begin{matrix} c_1 = (1, 1) & \text{group 1} \\ c_2 = \left(\frac{11}{3}, \frac{8}{3}\right) & \text{group 2} \end{matrix}$$

From the above object centroid distance matrix we can see,

- Item 1 has minimum distance for group1, so we cluster item 1 in group 1.
- Item 2 has minimum distance for group 1, so we cluster item 2 in group 1.
- Item 3 has minimum distance for group 2, so we cluster item 3 in group 2.
- Item 4 has minimum distance for group 2, so we cluster item 4 in group 2.

Object Clustering:

$$G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Iteration 2 : Determine centroids:

$$C_1 = (1 + 2/2, 1 + 1/2) = (3/2, 1)$$

$$C_2 = (4 + 5/2, 3 + 4/2) = (9/2, 7/2)$$

The distance of item 1 = (1, 1) to c_1 = (3/2, 1) and with c_2 = (9/2, 7/2) is calculated as,

$$D = \sqrt{(1 - 3/2)^2 + (1 - 1)^2} = 0.5$$

$$D = \sqrt{(1 - 9/2)^2 + (1 - 7/2)^2} = 4.3$$

The distance of item 2 = (2, 1) to c_1 = (3/2, 1) and with c_2 = (9/2, 7/2) is calculated as,

$$D = \sqrt{(2 - 3/2)^2 + (1 - 1)^2} = 0.5$$

$$D = \sqrt{(2 - 9/2)^2 + (1 - 7/2)^2} = 3.54$$

The distance of item 3 = (4, 3) to c_1 = (3/2, 1) and with c_2 = (9/2, 7/2) is calculated as,

$$D = \sqrt{(4 - 3/2)^2 + (3 - 1)^2} = 3.20$$

$$D = \sqrt{(4 - 9/2)^2 + (3 - 7/2)^2} = 0.71$$

The distance of item 4 = (5, 4) to c_1 = (3/2, 1) and with c_2 = (9/2, 7/2) is calculated as,

$$D = \sqrt{(5 - 3/2)^2 + (4 - 1)^2} = 4.61$$

$$D = \sqrt{(5 - 9/2)^2 + (4 - 7/2)^2} = 0.71$$

Objects-centroids distance:

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.3 & 3.54 & 0.71 & 0.71 \end{bmatrix} \begin{matrix} c_1 = \left(\frac{3}{2}, 1\right) \text{ group 1} \\ c_2 = \left(\frac{9}{2}, \frac{7}{2}\right) \text{ group 2} \end{matrix}$$

From the above object centroid distance matrix we can see,

- Item 1 has minimum distance for group1, so we cluster item 1 in group 1.
- Item 2 has minimum distance for group 1, so we cluster item 2 in group 1.
- Item 3 has minimum distance for group 2, so we cluster item 3 in group 2.
- Item 4 has minimum distance for group 2, so we cluster item 4 in group 2.

Object Clustering:

$$G^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$G^2 = G^1$, Objects does not move from group any more. So, the final clusters are as follows:

- Item 1 and 2 are clustered in group 1
- Item 3 and 4 are clustered in group 2

Example 4 :
Suppose we have eight data points and each data point has 2 features. Cluster the data points into 3 clusters using k-means algorithm.

Data points	Attribute 1(x)	Attribute 2(y)
1	2	10
2	2	5
3	8	4
4	5	8
5	7	5
6	6	4
7	1	2
8	4	9

Solution:

Initial value of centroid:
Suppose we use data points 1, 4 and 7 as the first centroids, $c_1 = (2, 10)$, $c_2 = (5, 8)$ and $c_3 = (1, 2)$

The distance of data point 1 = (2, 10) to $c_1 = (2, 10)$, $c_2 = (5, 8)$ and with $c_3 = (1, 2)$ is,

$$D = \sqrt{(2-2)^2 + (10-10)^2} = 0$$

$$D = \sqrt{(2-5)^2 + (10-8)^2} = 3.61$$

$$D = \sqrt{(2-1)^2 + (10-2)^2} = 8.06$$

The distance of data point 1 = (2, 5) to $c_1 = (2, 10)$, $c_2 = (5, 8)$ and with $c_3 = (1, 2)$ is,

$$D = \sqrt{(2-2)^2 + (5-10)^2} = 5$$

$$D = \sqrt{(2-5)^2 + (5-8)^2} = 4.24$$

$$D = \sqrt{(2-1)^2 + (5-2)^2} = 3.16$$

The distance of data point 1 = (8, 4) to $c_1 = (2, 10)$, $c_2 = (5, 8)$ and with $c_3 = (1, 2)$ is,

$$D = \sqrt{(8-2)^2 + (4-10)^2} = 8.48$$

$$D = \sqrt{(8-5)^2 + (4-8)^2} = 5$$

$$D = \sqrt{(8-1)^2 + (4-2)^2} = 7.28$$

The distance of data point 1 = (5, 8) to c_1 = (2, 10), c_2 = (5, 8) and with c_3 = (1, 2) is,

$$D = \sqrt{(5-2)^2 + (8-10)^2} = 3.61$$

$$D = \sqrt{(5-5)^2 + (8-8)^2} = 0$$

$$D = \sqrt{(5-1)^2 + (8-2)^2} = 7.21$$

The distance of data point 1 = (7, 5) to c_1 = (2, 10), c_2 = (5, 8) and with c_3 = (1, 2) is,

$$D = \sqrt{(7-2)^2 + (5-10)^2} = 7.07$$

$$D = \sqrt{(7-5)^2 + (5-8)^2} = 3.61$$

$$D = \sqrt{(7-1)^2 + (5-2)^2} = 6.71$$

The distance of data point 1 = (6, 4) to c_1 = (2, 10), c_2 = (5, 8) and with c_3 = (1, 2) is,

$$D = \sqrt{(6-2)^2 + (4-10)^2} = 7.21$$

$$D = \sqrt{(6-5)^2 + (4-8)^2} = 4.12$$

$$D = \sqrt{(6-1)^2 + (4-2)^2} = 5.39$$

The distance of data point 1 = (1, 2) to c_1 = (2, 10), c_2 = (5, 8) and with c_3 = (1, 2) is,

$$D = \sqrt{(1-2)^2 + (2-10)^2} = 8.06$$

$$D = \sqrt{(1-5)^2 + (2-8)^2} = 7.21$$

$$D = \sqrt{(1-1)^2 + (2-2)^2} = 0$$

The distance of data point 1 = (4, 9) to c_1 = (2, 10), c_2 = (5, 8) and with c_3 = (1, 2) is,

$$D = \sqrt{(4-2)^2 + (9-10)^2} = 2.24$$

$$D = \sqrt{(4-5)^2 + (9-8)^2} = 1.4$$

$$D = \sqrt{(4-1)^2 + (9-2)^2} = 7.62$$

Objects-centroids distance:

$$\mathbf{D^0} = \begin{bmatrix} \mathbf{0} & \mathbf{5} & \mathbf{8.48} & \mathbf{3.61} & \mathbf{7.07} & \mathbf{7.21} & \mathbf{8.06} & \mathbf{2.24} \\ \mathbf{3.61} & \mathbf{4.24} & \mathbf{5} & \mathbf{0} & \mathbf{3.61} & \mathbf{4.12} & \mathbf{7.21} & \mathbf{1.4} \\ \mathbf{8.06} & \mathbf{3.16} & \mathbf{7.28} & \mathbf{7.21} & \mathbf{6.71} & \mathbf{5.39} & \mathbf{0} & \mathbf{7.62} \end{bmatrix}$$

c_1 = (2, 10) group 1

c_2 = (5, 8) group 2

c_3 = (1, 2) group 3

From the above object centroid distance matrix we can see,

- Data point 1 has minimum distance for group1, so we cluster data point 1in group 1.
- Data point 2 has minimum distance for group3, so we cluster data point 2in group 3.
- Data point 3 has minimum distance for group 2, so we cluster data point 3 in group 2.
- Data point 4 has minimum distance for group 2, so we cluster data point 4 in group 2.
- Data point 5 has minimum distance for group 2, so we cluster data point 5 in group 2.
- Data point 6 has minimum distance for group 2, so we cluster data point 6 in group 2.
- Data point 7 has minimum distance for group 3, so we cluster data point 7 in group 3.
- Data point 8 has minimum distance for group 2, so we cluster data point 8 in group 2.

Object Clustering:

$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Iteration 1: Determine centroids

C1has only one member thus $c_1 = (2, 10)$ remains same.

$C_2 = (8 + 5 + 7 + 6 + 4/5, 4 + 8 + 5 + 4 + 9/5) = (6, 6)$

$C_3 = (2 + 1/2, 5 + 2/2) = (1.5, 3.5)$

Objects-centroids distance:

$$D^1 = \begin{bmatrix} 0 & 5 & 8.48 & 3.61 & 7.07 & 7.21 & 8.06 & 2.24 \\ 5.66 & 4.12 & 2.83 & 2.24 & 1.41 & 2 & 6.40 & 3.16 \\ 6.52 & 1.58 & 6.25 & 5.7 & 5.7 & 4.52 & 1.58 & 6.04 \end{bmatrix}$$

$c_1 = (2, 10)$ group 1

$c_2 = (6, 6)$ group 2

$c_3 = (1.5, 3.5)$ group 3

Object Clustering:

$$G^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Iteration 2 : Determine centroids:

C_1 = $(2 + 4/2, 10 + 9/2) = (3, 9.5)$

C_2 = $(8 + 5 + 7 + 6/4, 4 + 8 + 5 + 4/4) = (6.5, 5.25)$

C_3 = $(2 + 1/2, 5 + 2/2) = (1.5, 3.5)$

$$D^2 = \begin{bmatrix} 1.12 & 2.35 & 7.43 & 2.5 & 6.02 & 6.26 & 7.76 & 1.12 \\ 6.54 & 4.51 & 1.95 & 3.13 & 0.56 & 1.35 & 6.38 & 7.68 \\ 6.52 & 1.58 & 6.52 & 5.7 & 5.7 & 4.52 & 1.58 & 6.04 \end{bmatrix}$$

$c_1 = (3, 9.5)$ group 1

$c_2 = (6.5, 5.25)$ group 2

$c_3 = (1.5, 3.5)$ group 3

Object Clustering:

$$G^2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Iteration 3 : Determine centroids:

C_1 = $(2 + 5 + 4/3, 10 + 9 + 8/3) = (3.67, 9)$

C_2 = $(8 + 7 + 6/3, 4 + 5 + 4/3) = (7, 4.33)$

C_3 = $(2 + 1/2, 5 + 2/2) = (1.5, 3.5)$

$$D^2 = \begin{bmatrix} 1.95 & 4.33 & 6.61 & 1.66 & 5.2 & 5.52 & 7.49 & 0.33 \\ 6.01 & 5.04 & 1.05 & 4.17 & 0.67 & 1.05 & 6.44 & 5.55 \\ 6.52 & 1.58 & 6.52 & 5.7 & 5.7 & 4.52 & 1.58 & 6.04 \end{bmatrix}$$

$c_1 = (3.67, 9)$ group 1

$c_2 = (7, 4.33)$ group 2

$c_3 = (1.5, 3.5)$ group 3

Object Clustering:

$$G^3 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$G^3 = G^2$, Objects does not move from group any more. So, the final clusters are as follows:

- Data points 1, 4 and 8 are clustered in group 1
- Data points 3, 5 and 6 are clustered in group 2
- Data points 2 and 7 are clustered in group 3

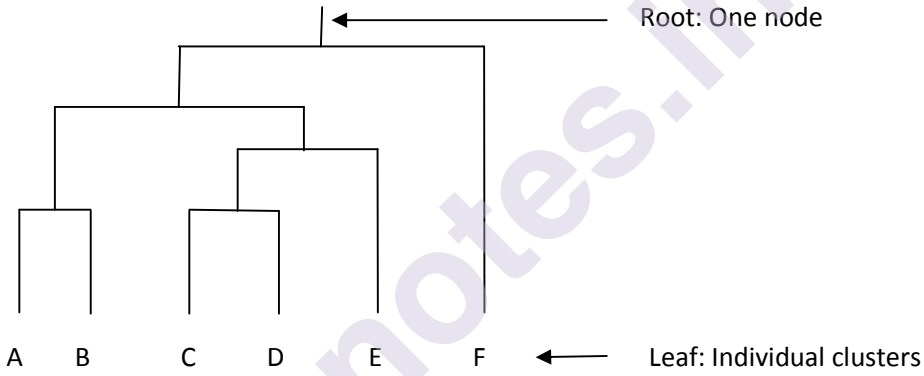
8.5 HIERARCHICAL CLUSTERING

Hierarchical clustering algorithms works in top down manner or bottom up manner. Hierarchical clustering is known as Hierarchical agglomerative clustering.

8.5.1 Agglomerative Hierarchical Clustering:

In agglomerative clustering initially each data point is considered as a single cluster. In the next step, pairs of clusters are merged or agglomerated. This step is repeated until all clusters have been merged in to a single cluster. At the end a single cluster remains that contains all the data points. Hierarchical clustering algorithms works in top-down manner or bottom-up manner. Hierarchical clustering is known as Hierarchical agglomerative clustering.

In agglomerative, clustering is represented as a dendogram as shown below where each merge is represented by a horizontal line



A clustering of the data objects is obtained by cutting the dendogram at the desired level,then each connected forms a cluster.

The basic steps of Agglomerative hierarchical clustering are as follows:

1. Compute the proximity matrix(distance matrix)
2. Assume each data point as a cluster.
3. Repeat
4. Merge the two nearest clusters.
5. Update the proximity matrix
6. Until only a single cluster remains

In Agglomerative hierarchical clustering proximity matrix is symmetric ie., the number on lower half will be same as the numbers on top half.

Different approaches to defining the distance between clusters distinguish the different algorithm's ie., Single linkage,Complete linkage and Average linkage clusters.

In single linkage, the distance between two clusters is considered to be

equal to shortest distance from any member of one cluster to any member of other cluster.

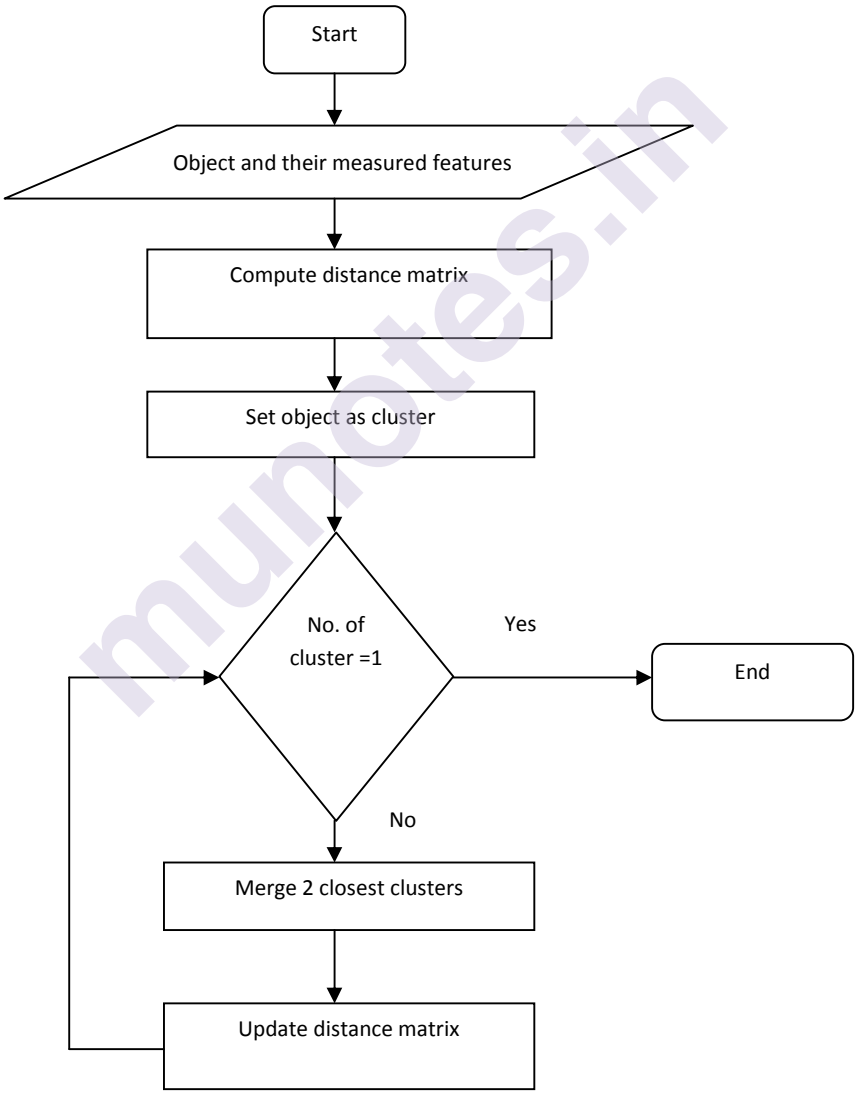
D(r,s) =Min { d(i,j), object i-> cluster r and object j-> cluster s

In complete linkage, the distance between two clusters is considered to be equal to greatest distance from any member of one cluster to any member of other cluster.

D(r,s) =Max { d(i,j), object i-> cluster r and object j-> cluster s

In average linkage, we consider the distance between any two clusters A and B is taken to be equal to average of all distances between pairs of object I in A and j in B.ie., mean distance between elements of each other.

D(r,s) =Mean { d(i,j), object i-> cluster r and object j-> cluster s



8.5.2. Examples of Hierarchical Clustering:

Example 1:

The table below shows the six data points. Use all link methods to find clusters. Use Euclidian distance measure.

	X	**y**
D_1	0.4	0.53
D_2	0.22	0.38
D_3	0.35	0.32
D_4	0.26	0.19
D_5	0.08	0.41
D_6	0.45	0.30

Solution:

First we will solve using single linkage:

The distance of data point $D_1 = (0.4, 0.53)$ to $D_2 = (0.22, 0.38)$ is,

$$D = \sqrt{(0.4 - 0.22)^2 + (0.53 - 0.38)^2} = 0.24$$

The distance of data point $D_1 = (0.4, 0.53)$ to $D_3 = (0.35, 0.32)$ is,

$$D = \sqrt{(0.4 - 0.35)^2 + (0.53 - 0.32)^2} = 0.22$$

The distance of data point $D_1 = (0.4, 0.53)$ to $D_4 = (0.26, 0.19)$ is,

$$D = \sqrt{(0.4 - 0.26)^2 + (0.53 - 0.19)^2} = 0.37$$

The distance of data point $D_1 = (0.4, 0.53)$ to $D_5 = (0.08, 0.41)$ is,

$$D = \sqrt{(0.4 - 0.08)^2 + (0.53 - 0.41)^2} = 0.34$$

The distance of data point $D_1 = (0.4, 0.53)$ to $D_6 = (0.45, 0.30)$ is,

$$D = \sqrt{(0.4 - 0.45)^2 + (0.53 - 0.30)^2} = 0.23$$

Similarly we will calculate all distances.

Distance matrix:

D_1	**0**					
D_2	**0.24**	**0**				
D_3	**0.22**	**0.15**	**0**			
D_4	**0.37**	**0.20**	**0.15**	**0**		
D_5	**0.34**	**0.14**	**0.28**	**0.29**	**0**	
D_6	**0.23**	**0.25**	**0.11**	**0.22**	**0.39**	**0**
	D_1	D_2	D_3	D_4	D_5	D_6

0.11 is smallest. D_3 and D_6 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), D_1)$ = min (distance (D_3, D_1), distance (D_6, D_1)) = min $(0.22, 0.23) = 0.22$

Distance $((D_3, D_6), D_2)$ = min (distance (D_3, D_2), distance (D_6, D_2)) = min $(0.15, 0.25) = 0.15$

Distance $((D_3, D_6), D_4)$ = min (distance (D_3, D_4), distance (D_6, D_4)) = min $(0.15, 0.22) = 0.15$

Distance $((D_3, D_6), D_5)$ = min (distance (D_3, D_5), distance (D_6, D_5)) = min $(0.28, 0.39) = 0.28$

Similarly we will calculate all distances.

Distance matrix:

D_1	**0**				
D_2	**0.24**	**0**			
(D_3, D_6)	**0.22**	**0.15**	**0**		
D_4	**0.37**	**0.20**	**0.15**	**0**	
D_5	**0.34**	**0.14**	**0.28**	**0.29**	**0**
	D_1	D_2	(D_3, D_6)	D_4	D_5

0.14 is smallest. D_2 and D_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), (D_2, D_5))$ = min (distance (D_3, D_2), distance (D_6, D_2), distance (D_3, D_5), distance (D_6, D_6))

= min $(0.15, 0.25, 0.28, 0.29) = 0.15$

Similarly, we will calculate all distances.

Distance matrix:

D_1	**0**			
(D_2, D_5)	**0.24**	**0**		
(D_3, D_6)	**0.22**	**0.15**	**0**	
D_4	**0.37**	**0.20**	**0.15**	**0**
	D_1	(D_2, D_5)	(D_3, D_6)	D_4

0.15 is smallest. (D_2, D_5) and (D_3, D_6) as well as D_4 and (D_3, D_6) have smallest distance. We can pick either one.

Distance matrix:

D_1	**0**		
(D_2, D_5, D_3, D_6)	**0.22**	**0**	
D_4	**0.37**	**0.15**	**0**
	D_1	(D_2, D_5, D_3, D_6)	D_4

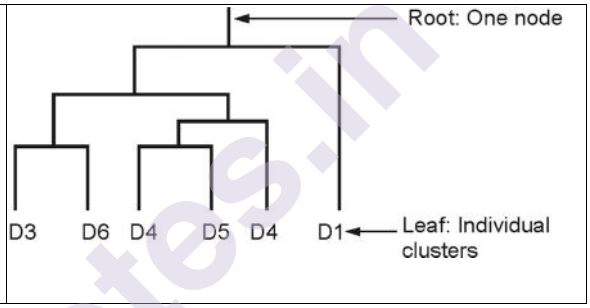
0.15 is smallest. (D_2, D_5, D_3, D_6) and D_4 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

D_1	**0**	
$(D_2, D_5, D_3, D_6, D_4)$	**0.22**	**0**
	D_1	$(D_2, D_5, D_3, D_6, D_4)$

Now a single cluster remains $(D_2, D_5, D_3, D_6, D_4, D_1)$

Next, we represent the final dendogram for single linkage as,



Now we will solve using complete linkage:

Distance matrix:

D_1	0					
D_2	0.24	0				
D_3	0.22	0.15	0			
D_4	0.37	0.20	0.15	0		
D_5	0.34	0.14	0.28	0.29	0	
D_6	0.23	0.25	0.11	0.22	0.39	0
	D_1	D_2	D_3	D_4	D_5	D_6

0.11 is smallest. D_3 and D_6 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), D_1)$ = max (distance (D_3, D_1), distance (D_6, D_1)) = max $(0.22, 0.23) = 0.23$

Similarly, we will calculate all distances.

Distance matrix:

D_1	**0**				
D_2	**0.24**	**0**			
(D_3, D_6)	**0.23**	**0.25**	**0**		
D_4	**0.37**	**0.20**	**0.22**	**0**	
D_5	**0.34**	**0.14**	**0.39**	**0.29**	**0**
	D_1	**D_2**	**(D_3, D_6)**	**D_4**	**D_5**

0.14 is smallest. D_2 and D_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

D_1	0			
(D_2, D_5)	0.34	0		
(D_3, D_6)	0.23	0.39	0	
D_4	0.37	0.29	0.22	0
	D_1	(D_2, D_5)	(D_3, D_6)	D_4

- 0.22 is smallest. Here (D_3, D_6) and D_4 have smallest distance. So, we combine these two in one cluster and recalculate distance matrix.

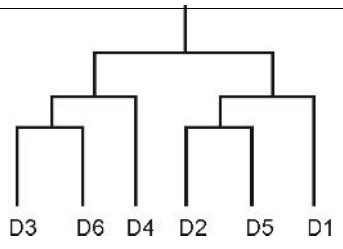
Distance matrix:

D_1	0		
(D_2, D_5)	0.34	0	
(D_3, D_6, D_4)	0.37	0.39	0
	D_1	(D_3, D_6, D_4)	(D_3, D_6, D_4)

0.34 is smallest. (D_2, D_5) and D_1 have smallest distance so, we combine these two in one cluster and recalculate distance matrix.

Distance matrix:

(D_2, D_5, D_1)	**0**	**0**
(D_3, D_6, D_4)	**0.39**	**0**
	(D_2, D_5, D_1)	(D_3, D_6, D_4)

Now a single cluster remains (D_2, D_5, D_1, D_3, D_6, D_4)	
Next, we represent the final dendogram for complete linkage as,	

Now we will solve using average linkage

Distance matrix:

D_1	0					
D_2	0.24	0				
D_3	0.22	0.15	0			
D_4	0.37	0.20	0.15	0		
D_5	0.34	0.14	0.28	0.29	0	
D_6	0.23	0.25	0.11	0.22	0.39	0
	D_1	D_2	D_3	D_4	D_5	D_6

0.11 is smallest. D_3 and D_6 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((D_3, D_6), D_1)$ = 1/2 (distance (D_3, D_1) + distance (D_6, D_1)) = 1/2 (0.22 + 0.23) = 0.23

Similarly, we will calculate all distances.

Distance matrix:

D_1	0				
D_2	0.24	0			
(D_3, D_6)	0.23	0.2	0		
D_4	0.37	0.20	0.19	0	
D_5	0.34	0.14	0.34	0.29	0
	D_1	D_2	(D_3, D_6)	D_4	D_5

0.14 is smallest. D_2 and D_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

D_1	0			
(D_2, D_5)	0.29	0		
(D_3, D_6)	0.22	0.27	0	
D_4	0.37	0.22	0.15	0
	D_1	(D_2, D_5)	(D_3, D_6)	D_4

(D_3, D_6) and D_4 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

D_1	0		
(D_2, D_5)	0.24	0	
(D_3, D_6, D_4)	0.27	0.26	0
	D_1	(D_2, D_5)	(D_3, D_6, D_4)

0.24 is smallest. (D_2, D_5) and D_1 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

(D_2, D_5, D_1)	**0**	**0**
(D_3, D_6, D_4)	**0.26**	**0**
	(D_2, D_5, D_1)	(D_3, D_6, D_4)

Now a single cluster remains (D_2, D_5, D_1, D_3, D_6, D_4)
Next, we represent the final dendogram for average linkage as,

Example 2:
Apply single linkage, complete linkage and average linkage on the following distance matrix and draw dendogram.

Solution :
First we will solve using single linkage

Distance matrix:

P_1	0				
P_2	2	0			
P_3	6	3	0		
P_4	10	9	7	0	
P_5	9	8	5	4	0
	P_1	P_1	P_3	P_4	P_5

2 is smallest. P_1 and P_2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance ((P_1, P_2), P_3) = min (distance (P, P_3), distance (P_2, P_3)) = min (6, 3) = 3

Similarly, we will calculate all distances.

Distance matrix:

(P_1, P_2)	**0**			
P_3	**3**	**0**		
P_4	**9**	**7**	**0**	
P_5	**8**	**5**	**4**	**0**
	(P_1, P_2)	P_3	P_4	P_5

3 is smallest. (P_1, P_2) and P_3 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance ((P_1, P_2, P_3), P_4)) = min (distance (P_1, P_4), distance (P_2, P_4), distance (P_3, P_4)) = min (9, 7) = 7

Similarly, we will calculate all distances.

Distance matrix:

(P_1, P_2, P_3)	**0**		
P_4	**7**	**0**	
P_5	**5**	**4**	**0**
	(P_1, P_2, P_3)	P_4	P_5

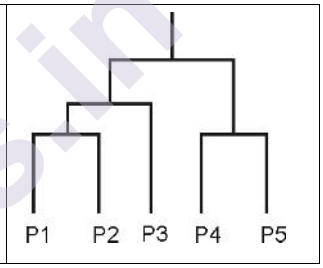
4 is smallest. P_4 and P_5 have smallest distance.

Distance matrix:

(P_1, P_2, P_3)	**0**	
(P_4, P_5)	**5**	**0**
	(P_1, P_2, P_3)	(P_4, P_5)

Now a single cluster remains $(P_1, P_2, P_3, P_4, P_5)$

Next, we represent the final dendogram for single linkage as,



Now we will solve using complete linkage
Distance matrix

P_1	**0**				
P_2	**2**	**0**			
P_3	**6**	**3**	**0**		
P_4	**10**	**9**	**7**	**0**	
P_5	**9**	**8**	**5**	**4**	**0**
	P_1	P_2	P_3	P_4	P_5

2 is smallest. P_1 and P_2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((P_1, P_2), P_3)$ = max (distance (P_1, P_3), distance (P_2, P_3)) = max $(6, 3) = 6$

Similarly, we will calculate all distances.

Distance matrix

(P_1, P_2)	**0**			
P_3	**6**	**0**		
P_4	**10**	**7**	**0**	
P_5	**9**	**5**	**4**	**0**
	(P_1, P_2)	P_3	P_4	P_5

4 is smallest. P_4 and P_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

(P_1, P_2)	**0**		
P_3	**6**	**0**	
(P_4, P_5)	**10**	**7**	**0**
	(P_1, P_2)	P_3	(P_4, P_5)

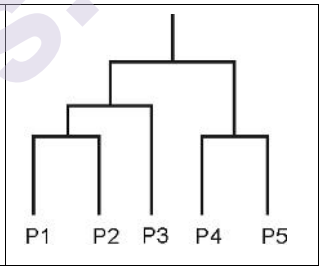
6 is smallest. (P_1, P_2) and P_3 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

(P_1, P_2, P_3)	**0**	
(P_4, P_5)	**10**	**0**
	(P_1, P_2, P_3)	(P_4, P_5)

Now a single cluster remains $(P_1, P_2, P_3, P_4, P_5)$

Next, we represent the final dendogram for complete linkage as,



Now we will solve using average linkage

Distance matrix:

P_1	0				
P_2	2	0			
P_3	6	3	0		
P_4	10	9	7	0	
P_5	9	8	5	4	0
	P_1	P_2	P_3	P_4	P_5

2 is smallest. P_1 and P_2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance $((P_1, P_2), P_3)$ = 1/2 (distance (P_1, P_3), distance (P_2, P_3)) = 1/2 (6, 3) = 4.5

Similarly, we will calculate all distances.

Distance matrix:

(P_1, P_2)	**0**			
P_3	**4.5**	**0**		
P_4	**9.5**	**7**	**0**	
P_5	**8.5**	**5**	**4**	**0**
	(P_1, P_2)	P_3	P_4	P_5

4 is smallest. P_4 and P_5 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

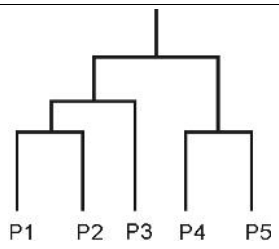
Distance matrix:

(P_1, P_2)	**0**		
P_3	**4.5**	**0**	
(P_4, P_5)	**9**	**6**	**0**
	(P_1, P_2)	P_3	(P_4, P_5)

4.5 is smallest. (P_1, P_2) and P_3 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix:

(P_1, P_2, P_3)	**0**	
(P_4, P_5)	**8**	**0**
	(P_1, P_2, P_3)	(P_4, P_5)

Now a single cluster remains (P_1, P_2, P_3, P_4, P_5) Next, we represent the final dendogram for average linkage as,	

SUMMARY

In this chapter we have seen distance based model which is based on the concept of distance. We have seen how to calculate the distance between the data using Euclidean and manhattan distance. In this chapter we have seen nearest neighbor method which is used to classify the data point in to one of the classes based on the concept of minimum distance. Here we have also seen K means algorithm in which we calculate the centroid and then distance of each data point is calculated from this centroid. Data points are clustered based on minimum distance and this process is repeated unless and until there is no change in the clutsers. We have also seen agglomerative clustering in which we calculate the distance matrix

which is used to find minimum distance. The data points having minimum distance are clustered together and distance matrix is updated. This process is repeated unless and until single cluster remains.

UNIT END EXERCISES

1. Describe the essential steps of K-means algorithm for clustering analysis.

2. Apply K-means algorithm algorithm on given data for k=3.Use c1(2),c2(16) and c3(38) as initial cluster centres.

   Data: 2,4,6,3,31,12,15,16,38,35,14,21, 23,25,30

3. Apply K-means algorithm algorithm on given data for k=3.Use c1(2),c2(16) and c3(38) as initial cluster centres.

   Data: 2,4,6,3,31,12,15,16,38,35,14,21,23,25,30

4. Apply Agglomerative clustering algorithm on given data and draw dendogram. Show three clusters with its allocated points. Use single link method.

	a	b	c	d	e	F
a	0	$\sqrt{2}$	$\sqrt{10}$	$\sqrt{17}$	$\sqrt{5}$	$\sqrt{20}$
b	$\sqrt{2}$	0	$\sqrt{8}$	3	1	$\sqrt{18}$
c	$\sqrt{10}$	$\sqrt{8}$	0	$\sqrt{5}$	$\sqrt{5}$	2
d	$\sqrt{17}$	1	$\sqrt{5}$	0	2	3
e	$\sqrt{5}$	1	$\sqrt{5}$	2	0	$\sqrt{13}$
f	$\sqrt{20}$	$\sqrt{18}$	2	3	$\sqrt{13}$	0

5. For the given set of points identify clusters using complete link and average link using Agglomerative clustering.

	A	B
P1	1	1
P2	1.5	1.5
P3	5	5
P4	3	4
P5	4	4
P6	3	3.5
	A	B

LIST OF REFERENCES

- kdnuggets.com/2019/06/main-approaches-machine-learning-models.html
- https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering
- https://www.displayr.com/what-is-hierarchical-clustering
- https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning.

*****

9

RULE BASED MODELS

Unit Structure

9.0 OBJECTIVES

1. To identify label or class of a given example
2. To identify rules for subgroup discovery
3. To identify frequent item set.

9.1 INTRODUCTION TO LOGICAL MODELS

Logical 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.

There are mainly two kinds of logical models: **Tree models** and **Rule models**.

9.1.1 Rule based Classifier:

Rule models consist of a collection of implications or IF-THEN rules. For tree-based models, the 'if-part' defines a segment and the 'then-part' defines the behaviour of the model for this segment. Rule models follow the same reasoning.

Rule-based classifier classifies records using a set of IF-THEN rules. The rules can be expressed in the following from −
(Condition) ->y

Where

- LHS of above rule is known as an antecedent or condition
- RHS of above rule is known as rule consequent
- Condition is a conjunction of attribute. Here condition consist of one or more attribute tests which are logically ANDed.
- Y represents the class label

Assume a rule R1,
R1: IF Buying-Price = high AND Maintenance_Price = high AND Safety = low THEN Car_evaluation = unacceptable

Rule R1 can also be rewritten as:
R1: (Buying-Price = high) ^ (Maintenance_Price = high)^ (Safety = low)->Car_evaluation = unacceptable

If the antecedent is true for a given record, then the consequent is given as output.

9.1.2 Example of Rule based classifier:

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
High	High	Small	High	Unacceptable
High	High	Small	Low	Unacceptable
Medium	High	Small	High	Acceptable
Low	Medium	Small	High	Acceptable
Low	Low	Big	High	Acceptable
Low	Low	Big	Low	Unacceptable
Medium	Low	Big	Low	Acceptable

High	Medium	Small	High	Unacceptable
High	Low	Big	High	Acceptable
Low	Medium	Big	High	Acceptable
High	Medium	Big	Low	Acceptable
Medium	Medium	Small	Low	Acceptable
Medium	High	Big	High	Acceptable
Low	Medium	Small	Low	Unacceptable

R1: (Buying-Price = high) ^ (Maintenance_Price = high)^ (Safety = low)->Car_evaluation = unacceptable

R2: (Buying-Price = medium) ^(Maintenance_Price = high) ^(Lug_Boot= big)->Car_evaluation = acceptable

R3: (Buying-Price = high) ^(Lug_Boot = big)->Car_evaluation = unacceptable

R4: (Maintenance_Price= medium) (Lug_Boot = big)^ (Safety = high)->Car_evaluation = acceptable

9.1.3 Application of Rule based classifier:

A record x is said to be covered by a rule, if all the attributes present in the record satisfy the antecedent of the rule.

Car	Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
1	High	High	Small	low	?
2	medium	High	big	Low	?
3	High	medium	big	high	?
4	High	medium	small	low	?

Car 1 triggers rule R1=>unacceptable

Car 2 triggers rule R2 =>acceptable

Car 3 triggers both R3 and R4

Car 4 triggers none of the rules

9.1.4 Characteristicsof Rule based classifier:

1. Mutually Exclusive rules- Rule based Classifier comprises of mutually exclusive rules if the rules are independent of each other. Each and every record is covered by at most one rule.

Solution:- Arrange rules in the order

Arrangement of Rules in the Order:

Rules are assigned a priority and based on this they are arranged and ranks are associated. When a test record is given as a input to the classifier, a label of the class with highest priority triggered rule is assigned. If the test record does not trigger any of the rule then a default class is assigned.

Rule-based ordering:- In rule based ordering individual rules are ranked based on their quality.

R1: (Buying-Price = high) ^ (Maintenance_Price = high)^ (Safety = low)->Car_evaluation = unacceptable

R2: (Buying-Price = medium) ^ (Maintenance_Price = high) ^ (Lug_Boot= big)->Car_evaluation = acceptable

R3: (Buying-Price = high) ^(Lug_Boot = big)->Car_evaluation = unacceptable

R4: (Maintenance_Price= medium) (Lug_Boot = big)^ (Safety = high)->Car_evaluation = acceptable

Car	Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation ?
3	High	Medium	big	high	?

Car 3 triggers rule R3 first => unacceptable

Class-based ordering:- In class based ordering rules which belongs to the same class are grouped together.

R2: (Buying-Price = medium) ^(Maintenance_Price = high) ^(Lug_Boot= big)->Car_evaluation = acceptable

R4: (Maintenance_Price= medium) (Lug_Boot = big)^ (Safety = high)->Car_evaluation = acceptable

R1: (Buying-Price = high) ^ (Maintenance_Price = high)^ (Safety = low)->Car_evaluation = unacceptable

R3: (Buying-Price = high) ^(Lug_Boot = big)->Car_evaluation = unacceptable

Car 3 triggers rule R4 =>acceptable

2. Exhaustive rules: It said to has a complete coverage for the rule based Classifier if it accounts for each doable attribute values combination. Every instance is roofed with a minimum of one rule.

Solution: - Use a default class

9.1.5 Rule Building using Coverage Algorithm:
Coverage Algorithm can be used to extract IF-THEN rules form the training data. We do not require to generate a decision tree first. In this algorithm, each rule for a given class covers many of the tuples of that class.

As per the general strategy the rules are learned one at a time. For each time rules are learned, a tuple covered by the rule is removed and the process continues for the rest of the tuples. This is because the path to each leaf in a decision tree corresponds to a rule.

Note − The Decision tree induction can be considered as learning a set of rules simultaneously.

The Following is the sequential learning Algorithm where rules are learned for one class at a time. When learning a rule from a class Ci, we want the rule to cover all the tuples from class C only and no tuple form any other class.

Algorithm: Coverage

Input:
D, a data set class-labeled tuples,
Att_vals, the set of all attributes and their possible values.

Output: A Set of IF-THEN rules.
Method:
Rule_set={ }; // initial set of rules learned is empty

for each class c do

repeat
Rule = Learn_One_Rule(D, Att_valls, c);
remove tuples covered by Rule form D;
until termination condition;

Rule_set=Rule_set+Rule; // add a new rule to rule-set
end for
returnRule_Set;

9.2 RULE LEARNING FOR SUBGROUP DISCOVERY

9.2.1 Subgroup discovery:

- Imagine you want to market the new version of a successful product. You have a database of people who have been sent information about the previous version, containing all kinds of demographic, economic and social information about those people, as well as whether or not they purchased the product.

- If you were to build a classifier or ranker to find the most likely customers for your product, it is unlikely to outperform the majority class classifier (typically, relatively few people will have bought the product).

- However, what you are really interested in is finding reasonably sized subsets of people with a proportion of customers that is significantly higher than in the overall population. You can then target those people in your marketing campaign, ignoring the rest of your database.

- Subgroups are subsets of the instance space – or alternatively, mappings $\hat{g} : X \rightarrow \{true,false\}$ that are learned from a set of labelled examples $(x_i, label(x_i))$, where label : $X \rightarrow C$ is the true labelling function.

- A good subgroup is one whose class distribution is significantly different from the overall population. This is by definition true for pure subgroups, but these are not the only interesting ones.

- Consider small dolphins data set with positive examples

  p1 : Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many

  p2 : Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many

  p3 : Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

  p4 : Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many

  p5 : Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

  and negatives

  n1 : Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many

  n2 : Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many

  n3 : Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many

  n4 : Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many

  n5 : Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

- For instance, one could argue that the complement of a subgroup is as interesting as the subgroup itself: in our dolphin example, the concept Gills = yes, which covers four negatives and no positives, could be considered as interesting as its complement Gills = no, which covers one negative and all positives.

- This means that we need to move away from impurity-based evaluation measures.

9.2.2 Working of Rule learning for subgroup discovery:

Classification identifies a rule such that it forms a pure class. From a training set rules are used to identify pure subset of examples means only positive or negative instances.This is called subgroups. Based on rules we have to find subset of instance space. Along with finding subgroups we need to find interesting patterns.Means if we apply one rule we get one output and applying second rule we get other output. If subgroup is interesting pattern then its complementary is also an interesting pattern.

Eg Gills=yes it identifies 4 negative and 0 positive. If we are assuming that it is an interesting pattern then its complementary Gills=No. Here we will assume that it will identify 5 positive, but it may not be true. It may identify 5 positive and 1 negative. means Complementary of rule is not complementary of subgroup**.**

9.2.3 Measures in subgroup discovery:

If we draw a graph considering positives along y axis and negatives along x axis. suppose there are different subgroups are formed. Any subgroup that is present on diagonalhave equal proportion of positives to overall population.

1. **Precision:** Any subgroup that is present on diagonal have same propotion of positives to overall positives. Examples which are present on above diagonal has more number of positives and examples that are below diagonal have less number of positives.
   Precision = $|p \quad - p \quad |$

2. **Avearge recall:** The subgroups that are present on diagonal have avearge recall of 0.5
   Average recall = $|a \qquad r \qquad - 0.5|$

   If above value is positive means the exmaples are present on above diagonal else below diagonal.

3. **Weighted relative accuracy:** Here accuracy is assigned to each subgroup and one subgroup is compared to other one.
4. Weighted relative accuracy = pos*neg (true positive rate – false positive rate)

9.2.4 Weighted Coverage Algorithm for subgroup discovery:

The main difference in rule based classification and subgroup discover is overlapping rules.In classification rules are not overlapped. We have seen coverage algorithm in section 9.1. Whenever new rule is generated, we identify the instances that are satisfied by the rule then we remove those examples. In subgroup discovery the rules are overlapped. Whenever new rule is generated then examples are not directly removed. For this purpose we are usingweighted coverage algorithm. Herefor each example we are assigning weight as 1. Whenever rule is generated and example is covered by rule then weight of that example is halved. When it is 0 then we remove that example from set.

Algorithm: Weighted coverage
Input: Data set D
Output: Rule set R
R= null
While some examples in D, have weight =1 do
    r= LearnRule(D)// Coverage algorithm of classification
append r to the end R
decrease weights of examample covered by r
return R

9.3 INTRODUCTION TO ASSOCIATION RULE MINING

- Associations are things that usually occur together. For example, in market basket analysis we are interested in items frequently bought together. An example of an association rule is if milk then bread, stating that customers who buy milk tend to also buybread.

- In a bakery shop most clients will buy cake. This means that there will be many frequent item sets involving cake, such as {candle,cake}.

- This might suggest the construction of an association rule **if candlethen cake** – however, this is predictable given that {cake} is already a frequent item set (and clearly at least as frequent as {candle,cake}).

- Of more interest would be the converse rule **if cake then candle** which expresses that a considerable proportion of the people buying cake also buy a candle.

9.3.1 Apriori Algorithm:

The Apriori algorithm uses frequent item sets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rule, it determines how strongly or how weakly two objects are connected. This algorithm uses a **breadth-first search** and **Hash Tree** to calculate the itemset associations efficiently. It is the iterative process for finding the frequent item sets from the large dataset.

This algorithm was given by the **R. Agrawal** and **Srikant** in the year **1994**. It is mainly used for *market basket analysis* and helps to find those products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

9.3.1.1 What is Frequent Itemset?:

Frequent itemsets are those items whose support is greater than the threshold value or user-specified minimum support. It means if A & B are the frequent itemsets together, then individually A and B should also be the frequent itemset.

Suppose there are the two transactions: A= {1,2,3,4,5}, and B= {2,3,7}, in these two transactions, 2 and 3 are the frequent itemsets.

9.3.1.2 Steps for Apriori Algorithm:

Below are the steps for the apriori algorithm:

Step-1: Determine the support of itemsets in the transactional database, and select the minimum support and confidence.

Step-2: Take all supports in the transaction with higher support value than the minimum or selected support value.

Step-3: Find all the rules of these subsets that have higher confidence value than the threshold or minimum confidence.

Step-4: Sort the rules as the decreasing order of lift.

9.3.1.3 Apriori Algorithm Working:

We will understand the apriori algorithm using an example and mathematical calculation:

Example: Suppose we have the following dataset that has various transactions, and from this dataset, we need to find the frequent itemsets and generate the association rules using the Apriori algorithm given minimum support 2 and minimum confidence 50%

TID	ITEM SETS
T1	A,B
T2	B,D
T3	B,C
T4	A,B,D
T5	A,C
T6	B,C
T7	A,C
T8	A,B,C,E
T9	A,B,C

Solution:

Step-1: Calculating C1 and L1:

- In the first step, we will create a table that contains support count (The frequency of each itemset individually in the dataset) of each itemset in the given dataset. This table is called the **Candidate set or C1.**

Itemset	Support_count
A	6
B	7
C	5
D	2
E	1

Now, we will take out all the itemsets that have the greater support count that the Minimum Support (2). It will give us the table for the **frequent itemset L1.**

Since all the itemsets have greater or equal support count than the minimum support, except the E, so E itemset will be removed.

Itemset	Support_count
A	6
B	7
C	5
D	2

Step-2: Candidate Generation C2, and L2:
In this step, we will generate C2 with the help of L1. In C2, we will create the pair of the itemsets of L1 in the form of subsets.

After creating the subsets, we will again find the support count from the main transaction table of datasets, i.e., how many times these pairs have occurred together in the given dataset. So, we will get the below table for C2:

Itemset	Support_count
{A,B}	4
{A,C}	4
{A,D}	1
{B,C}	4
{B,D}	2
{C,D}	0

Again, we need to compare the C2 Support count with the minimum support count, and after comparing, the itemset with less support count will be eliminated from the table C2. It will give us the below table for L2

Itemset	Support_count
{A,B}	4
{A,C}	4
{B,C}	4
{B,D}	2

Step-3: Candidate generation C3, and L3:
For C3, we will repeat the same two processes, but now we will form the C3 table with subsets of three itemsets together, and will calculate the support count from the dataset. It will give the below table:

Itemset	Support_count
{A,B,C}	2
{B,C,D}	1
{A,C,D}	0
{A,B,D}	0

Now we will create the L3 table. As we can see from the above C3 table, there is only one combination of itemset that has support count equal to the minimum support count. So, the L3 will have only one combination,

i.e., **{A, B, C}.**

Step-4: Finding the association rules for the subsets:

To generate the association rules, first, we will create a new table with the possible rules from the occurred combination {A, B.C}. For all the rules, we will calculate the Confidence using formula **sup(A ^B)/A.** After calculating the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).

Consider the below table:

Rules	Support	Confidence
A ^B → C	2	Sup{(A ^B) ^C}/sup(A ^B)= 2/4=0.5=50%
B^C → A	2	Sup{(B^C) ^A}/sup(B ^C)= 2/4=0.5=50%
A^C → B	2	Sup{(A ^C) ^B}/sup(A ^C)= 2/4=0.5=50%
C→ A ^B	2	Sup{(C^(A ^B)}/sup(C)= 2/5=0.4=40%
A→ B^C	2	Sup{(A^(B ^C)}/sup(A)= 2/6=0.33=33.33%
B→ B^C	2	Sup{(B^(B ^C)}/sup(B)= 2/7=0.28=28%

As the given threshold or minimum confidence is 50%, so the first three rules **A ^B → C, B^C → A, and A^C → B** can be considered as the strong association rules for the given problem.

9.3.2 Association Rule Mining:

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.

The association rule learning is one of the very important concepts of machine learning, and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.** Here market basket analysis is a technique used by the various big retailer to discover the associations between items. We can understand it by taking an example of a

supermarket, as in a supermarket, all products that are purchased together are put together.

For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby. For Association rule learning Apriori algorithm can be used.

9.3.2.1 How does Association Rule Mining work?
Association rule learning works on the concept of If and Else Statement, such as if A then B.

Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known *as single cardinality*. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:

- **Support**
- **Confidence**
- **Lift**

Let's understand each of them:

Support:
Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Support}(X) = \text{Freq}(X) / T$$

Confidence:
Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence} = \text{Freq}(X,Y) / \text{Freq}(X)$$

Lift:
It is the strength of any rule, which can be defined as below formula:

$$\text{Lift} = \text{Support}(X,Y) / (\text{Support}(X) * \text{Support}(Y))$$

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

- If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.

- **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
- **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

9.3.2.2 Association Rule Mining using Apriori Algorithm:
This algorithm uses frequent datasets to generate association rules. It is designed to work on the databases that contain transactions. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset efficiently.

It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

9.3.2.3 Applications of Association Rule Mining:
It has various applications in machine learning and data mining. Below are some popular applications of association rule learning:

- **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
- It is also used for the **Catalog Design** and **Loss-leader Analysis** and many more other applications.

SUMMARY

In this chapter we have seen rule based model. In rule based model rules are defined in the form of if-then. Rules are used to identify the label or class of a given example. In classification when a new rule is generated then the example that covers this rule is eliminated from training set. In subgroup discovery we identify a pure class means all examples are corresponding to eithrt positive label or negative label. In subgroup discovery when new rule is generated the example which covers this rule is not directly eliminated. Weight of this example is halved and this process is repeated till it becomes zero. When a weight becomes zero then that example is eliminated from training set. This algorithm is called as weighted coverage algorithm. In this chapter we have also seen association rule mining. In this we find the frequent item set using apriori algorithm. Then using these frequent item set rules are defined.

UNIT END EXCERCISES

1. Find frequent item set and association rules for minimum support count 2 and minimum confidence is 60%

TID	Items
T1	i1,i2,i5
T2	i2,i4
T3	i2,i3
T4	i1,i2,i4
T5	i1,i3
T6	i2,i3
T7	i1,i3
T8	i1,i2,i3,i5
T9	i1,i2,i3

LIST OF REFERENCES

- https://www.geeksforgeeks.org/rule-based-classifier-machine-learning/
- Using rule learning for subgroup discovery,BrankoKavšek (2004) *Using rule learning for subgroup discovery*.
- https://www.upgrad.com/blog/association-rule-mining-an-overview-and-its-applications/
- https://www.educba.com/association-rules-in-data-mining/
- https://medium.com/analytics-vidhya/association-rule-mining-7f06401f0601\

*****

10

TREE BASED MODELS

Unit Structure

10.0 OBJECTIVES

- Gain conceptual picture of decision trees, regression trees and clustering trees.
- Learn to predict the class using decision tree.
- Learn to predict value of continuous variable by using regression tree.
- Learn to identify the ranking.

10.1 INTRODUCTION TO TREE MODEL

- A tree model is a hierarchical structure of conditions, where leafs contain tree outcome.

- They represent recursive divide-and-conquer strategies.
- Tree models are among the most popular models in machine learning, because they are easy to understand and interpret:
- Tree models can be seen as a particular type of rule model where the if-parts of the rules are organised in a tree structure.
- Both Tree models and Rule models use the same approach to supervised learning.
- The approach can be summarised in two strategies: we could first find the body of the rule (the concept) that covers a sufficiently homogeneous set of examples and then find a label to represent the body.
- Alternately, we could approach it from the other direction, i.e., first select a class we want to learn and then find rules that cover examples of the class.

10.2 DECISION TREES

- Decision trees are very strong and most suitable tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural network, decision trees represent rules.
- Rules are represented using linguistic variables so that user interpretability may be achieved. By comparing the records with the rules one can easily find a particular category to which the record belongs to.
- In some applications, the accuracy of a classification or prediction is the only thing that matters in such situations we do not necessarily care how or why the model works. In other situations, the ability to explain the reason for a decision is crucial, in marketing one has described the customer segments to marketing professionals, so that they can use this knowledge to start a victorious marketing campaign.
- This domain expert must acknowledge and approve this discovered knowledge and for this we need good descriptions. There are a variety of algorithms for building decision trees that share the desirable quality of interpretability (ID3).

10.2.1 Where Decision Tree is applicable? :

Decision tree method is mainly used for the tasks that possess the following properties.

- The tasks or the problems in which the records are represented by attribute-value pairs.
  - Records are represented by a fixed set of attribute and their value Example: For ‘temperature’ attribute the value is ‘hot’.

- When there are small numbers of disjoint possible values for each attribute, then decision tree learning becomes very simple.

  Example: Temperature attribute takes three values as hot, mild and cold.

- Basic decision tree algorithm may be extended to allow real valued attributes as well.

  Example: we can define floating point temperature.

- An application where the target function takes discrete output values.
  - In Decision tree methods an easiest situation exists, if there are only two possible classes.

    Example: Yes or No

  - When there are more than two possible output classes then decision tree methods can also be easily extended.
  - A more significant extension allows learning target functions with real valued outputs, although the application of decision trees in this area is not frequent.
- The tasks or the problems where the basic requirement is the disjunctive descriptors.
  - Decision trees naturally represent disjunctive expressions.
- In certain cases where the training data may contain errors.
  - Decision tree learning methods are tolerant to errors that can be a classification error of training records or attribute-value representation error.
- The training data may be incomplete as there are missing attribute values
  - Although some training records have unknown values, decision tree methods can be used.

10.2.2 Decision Tree Representation:

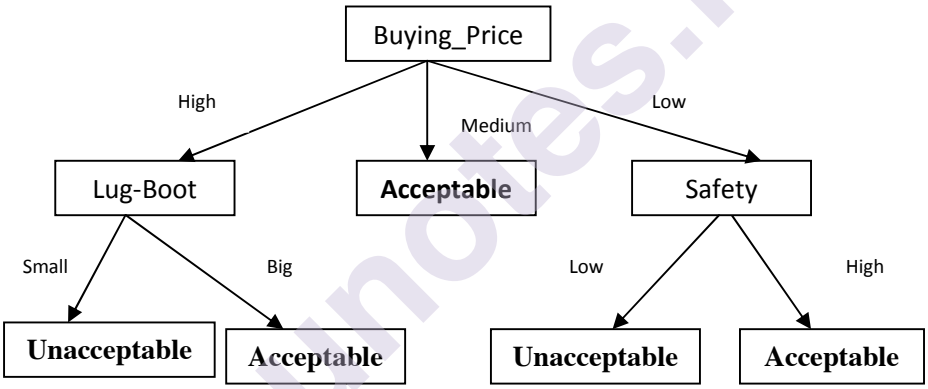
Decision tree is a classifier which is represented in the form of a tree structure where each node is either a leaf node or a decision node.

- Leaf node represents the value of the target or response attribute (class) of examples.
- Decision node represents some test to be carried out on a single attribute-value, with one branch and sub tree for each possible outcome of the test.

Decision tree generates regression or classification models in the form of a tree structure. Decision tree divides a dataset into smaller subsets with increase in depth of tree. The final decision tree is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Buying_Price) has two or more branches (e.g., High, Medium and Low). Leaf node (e.g., Evaluation)

shows a classification or decision. The topmost decision node in a tree which represents the best predictor is called **root node**. Decision trees can be used to represent categorical as well as numerical data.

1. **Root Node:** It represents entire set of records or dataset and this is again divided into two or more similar sets.
2. **Splitting:** Splitting procedure is used to divide a node into two or more sub-nodes depending on the criteria.
3. **Decision Node:** A decision node is a sub-node which is divided into more sub-nodes.
4. **Leaf/ Terminal Node:** Leaf node is a node which is not further divided or a node with no children.
5. **Parent and Child Node:** Parent node is a node, which is split into sub-nodes and sub-nodes are called as child of parent node.
6. **Branch / Sub-Tree:** A branch or sub-tree is a sub part of decision tree.
7. **Pruning:** Pruning method is used to reduce the size of decision trees by removing nodes.



10.2.3 Attribute Selection Measure:

1) Gini Index:

- All attributes are assumed to be continuous valued.
- It is assumed that there exist several possible split values for each attribute.
- Gini index method can be modified for categorical attributes.
- Gini is used in Classification and Regression Tree (CART).

If a data set T contains example from n classes, gini index, gini (T) is defined as,

Gini (T) =1- $\sum_{j=1}^{n}(Pj)^2$ (Eq.1)

In the above equation Pjrepresents the relative frequency of class j in T.
After splitting T into two subsets T1 and T2 with sizes N1 and N2,gini index of split data is,

Gini $_{split}$ (T) $= \frac{N1}{N} g \quad (T1) + \frac{N2}{N} g \quad (T2)$ (Eq.2)

The attribute with smallest gini $_{split}$ (T) is selected to split the node.

2) <u>Information Gain (ID3):</u>

In this method all attributes are assumed to be categorical. The method can be modified for continuous valued attributes. Here we select the attribute with highest information gain.

Assume there are 2 classes P and N. Let the set of records S contain p records of class P and n records of N.

The amount of information required to decide if a random record in S belongs to P or N is defined as,

$$I(p, n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right) \quad \text{(Eq.3)}$$

Assume that using attribute A, a set S will be partitioned in to sets $\{s_1, s_2, ----s_k\}$

If S_i has p_i records of P and n_i records of N, the entropy or the expected information required to classify objects in all subtrees S_i is,

$$E(A) = \sum_{i-1}^{v} \frac{pi+ni}{p+n} I(p_i, n_i) \quad \text{(Eq.4)}$$

Entropy (E):- Expected amount of information (in bits) needed to assign a class to a randomly drawn object in S under the optimal shortest length code.

Gain (A):- Measures reduction in entropy achieved because of split. Choose split that achieves most reduction (maximum Gain).

$$Gain(A) = I(p, n) - E(A) \quad \text{(Eq.5)}$$

10.2.4. Avoid Overfitting in classification (Tree pruning):

- The generated tree may overfit the training data.
  - If there are too many branches then some may reflect anomalies due to noise or outliers.
  - Overfitting result in poor accuracy for unseen samples.

- There are two approaches to avoid overfitting, prune the tree so that it is not too specific.
  - Prepruning (prune while building tree):-

    Stop tree construction early do not divide a node if this would result in the goodness measure falling below threshold.
  - Postpruning (prune after building tree):-

Fully constructed tree get a sequence of progressively pruned trees.

10.2.5. Strengths of Decision Tree Method:-

- Able to generate understandable rules
- Performs classification without requiring much computation.
- Able to handle both continuous and categorical variables.
- Decision tree clearly indicates which fields are most important for prediction or classification.

10.2.6. Weakness of Decision Tree Method:-

- Not suitable for prediction of continuous attribute
- Perform poorly with many class and small data
- Computationally expensive to train

10.2.7 Constructing Decision Trees:

The ID3 algorithm starts with the original set S as the root node. On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the entropy (or information gain) of that attribute. Algorithm next select the attribute which has the smallest entropy (or largest information gain) value. The set S is then divided by the chosen attribute (e.g. Income is less than 20 K , Income is between 20 K and 40 K, Income is greater than 40 K) to produce subsets of the data. The algorithm is recursively called for each subset, considering the attributes which are not selected before.

The stopping criteria for recursion can be one of these situations:

- When all records in the subset belongs to the same class (+ or -), then the node is converted into a leaf node and labeled with the class of the records.
- When we have selected all the attributes , but the records still do not belong to the same class (some are + and some are -), then the node is converted into a leaf node and labeled with the most frequent class of the records in the subset
- When there are no records in the subset, this is due to the non coverage of a specific attribute value for the record in the parent set, for example if there was no record with income = 40 K. Then a leaf node is generated, and labeled with the most frequent class of the record in the parent set.

Decision tree is generated with each non-terminal node representing the selected attribute on which the data was split, and terminal nodes representing the class label of the final subset of this branch.

Summary

Entropy of each and every attribute is calculated using the data set

1. Divide the set S into subsets using the attribute for which the resulting entropy (after splitting) is minimum (or, equivalently, information gain is maximum)
2. Make a decision tree node containing that attribute Recurse on subsets using remaining attributes.

10.2.8 Example of Classification Tree Using ID3:

Example 1:

Suppose we want ID3 to evaluate car database as whether the car is acceptable or not. The target classification is "Should we accept car?" which can be acceptable or unacceptable.

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
High	High	Small	High	Unacceptable
High	High	Small	Low	Unacceptable
Medium	High	Small	High	Acceptable
Low	Medium	Small	High	Acceptable
Low	Low	Big	High	Acceptable
Low	Low	Big	Low	Unacceptable
Medium	Low	Big	Low	Acceptable
High	Medium	Small	High	Unacceptable
High	Low	Big	High	Acceptable
Low	Medium	Big	High	Acceptable
High	Medium	Big	Low	Acceptable
Medium	Medium	Small	Low	Acceptable
Medium	High	Big	High	Acceptable
Low	Medium	Small	Low	Unacceptable

Class P: Evaluation = "Acceptable"
Class N: Evaluation = "Unacceptable"
Total records = 14
No. of records with Acceptable =9 and Unacceptable =5

$$I(p, n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

$$I(9, 5) = -\left(\frac{9}{1}\right)\log_2\left(\frac{9}{1}\right) - \left(\frac{5}{1}\right)\log_2\left(\frac{5}{1}\right)$$

$$= 0.940$$

Step 1->

1. Compute entropy for Buying_Price

   For Buying_Price = High

   Pi=2 and n_i =3

$$I(p_i, n_i) = I(2, 3) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.971$$

Similarly we will calculate I (p_i, n_i) for Medium and Low.

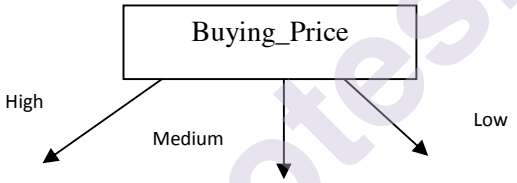
Buying_Price	**p_i**	**n_i**	**I(p_i, n_i)**
High	2	3	0.971
Medium	4	0	0
Low	3	2	0.971

$$E(A) = \sum_{i=1}^{v} \frac{pi+ni}{p+n} I(p_i, n_i)$$

$$E(\text{Buying\_Price}) = \left(\frac{5}{1}\right)I(2,3) + \left(\frac{4}{1}\right)I(4,0) + \left(\frac{5}{1}\right)I(3,2) = 0.694$$

Gain(S, Buying_Price) = I (p, n)-E (Buying_Price) =0.940-0.694 =0.246
Similarly, Gain (S, Maintenance_ Price) = 0.029, Gain (S, Lug_Boot) = 0.151, Gain (S, Safety) = 0.048

Since Buying_Price is the highest we select Buying_Price as the root node.



Step2->

As attribute Buying_Price at root, we have to decide on remaining tree attribute for High branch.

Buying_Price	**Maintenance_ Price**	**Lug_Boot**	**Safety**	**Evaluation?**
High	High	Small	High	Unacceptable
High	High	Small	Low	Unacceptable
High	Medium	Small	High	Unacceptable
High	Low	Big	High	Acceptable
High	Medium	Big	Low	Acceptable

No. Of records with Acceptable =2 and Unacceptable =3

$$I(p,n) = -\left(\frac{p}{p+n}\right)\log_2\left(\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\right)\log_2\left(\frac{n}{p+n}\right)$$

$$I(2,3) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right)$$

$$= 0.971$$

1. Compute entropy for Maintenance_ Price

Maintenance_ Price	p_i	n_i	$I(P_i, n_i)$
High	0	2	0
Medium	1	1	1
Low	1	0	0

E(Maintenance_ Price)= $\left(\frac{2}{5}\right)$I(0,2)+$\left(\frac{2}{5}\right)$ I(1,1)+$\left(\frac{1}{5}\right)$ I(1,0)= 0.4

Gain(S_{High}, Maintenance_ Price) = I (p, n)-E(Maintenance_ Price) =0.971-0.4 =0.571

2. Compute entropy for Lug_Boot

P_i=0 and n_i =3

I (P_i, n_i) = I(0,3)= 0

Lug_Boot	p_i	n_i	$I(P_i, n_i)$
Small	0	3	0
Big	2	0	0

E (Lug_Boot) = $\left(\frac{3}{5}\right)$I (0,3)+$\left(\frac{2}{5}\right)$ I(2,0)= 0

Gain (S_{High}, Lug_Boot) = I (p, n)- E(Lug_Boot) = 0.971- 0 = 0.971

3. Compute entropy for Safety

p_i = 1 and n_i = 2

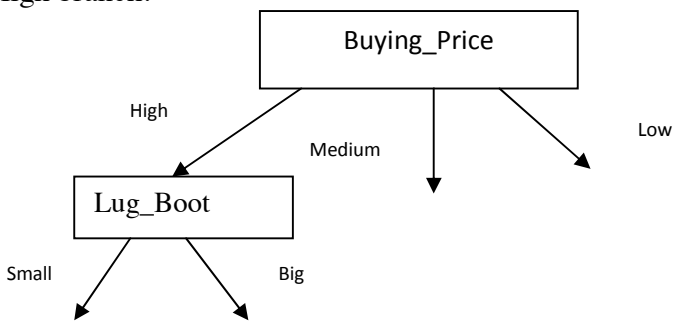
I (p_i, n_i) = I (1,2) = 0.918

Safety	p_i	n_i	$I(P_i, n_i)$
High	1	2	0.918
Low	1	1	1

E (Safety) = $\left(\frac{3}{5}\right)$I (1,2)+$\left(\frac{2}{5}\right)$ I(1,1)= 0.951

Gain(S_{High}, Safety) =I (p, n)-E (Safety) =0.971-0.951 =0.02

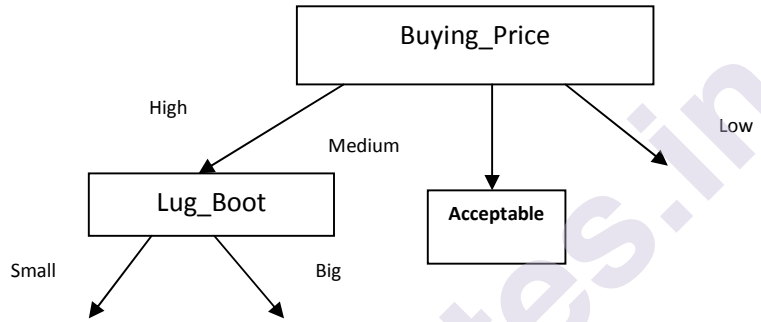
Since Lug_Boot is the highest we select Lug_Boot as a next node below High branch.



Step 3->
Consider now only Maintenance_ Price and Safety for Buying_Price = Medium

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
Medium	High	Small	High	Acceptable
Medium	Low	Big	Low	Acceptable
Medium	Medium	Small	Low	Acceptable
Medium	High	Big	High	Acceptable

Since for any combination of values of Maintenance_ Price and Safety, Evaluation? value is Acceptable, so we can directly write down the answer as Acceptable.



Step 4->
Consider now only Maintenance_ Price and Safety for Buying_Price = Low

Buying_Price	Maintenance_ Price	Lug_Boot	Saftey	Evaluation?
Low	Medium	Small	High	Acceptable
Low	Low	Big	High	Acceptable
Low	Low	Big	Low	Unacceptable
Low	Medium	Big	High	Acceptable
Low	Medium	Small	Low	Unacceptable

P_i=3 and n_i =2

$I(P_i, n_i) = I(3,2) = 0.970$

1. Compute entropy for Safety

Safety	p_i	n_i	$I(P_i, n_i)$
High	3	0	0
Low	0	2	0

$E(\text{Safety}) = \left(\frac{3}{5}\right) I(3,0) + \left(\frac{2}{5}\right) I(0,2) = 0$

Gain (S_{Low}, Safety) =I(p,n)-E(Safety) =0.970-0 =0.970

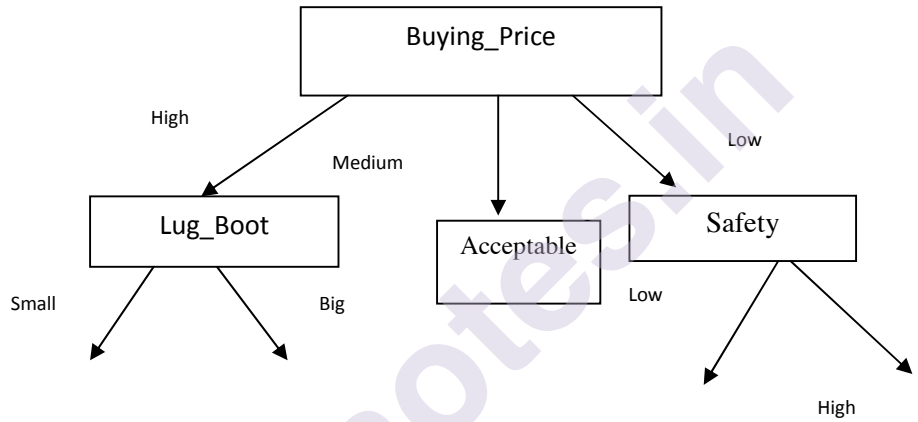
2. Compute entropy for Maintenance_ Price

Maintenance_ Price	**p_i**	**n_i**	**$I(P_i, n_i)$**
High	0	0	0
Medium	2	1	0.918
Low	1	1	1

E(Maintenance_ Price)= $\left(\frac{0}{5}\right)$I(0,0)+$\left(\frac{3}{5}\right)$ I(2,1)+ $\left(\frac{2}{5}\right)$ I(1,1)= 0.951

Gain (S_{Low}, Maintenance_ Price) =I (p, n)-E (Maintenance_ Price) =0.970-0.951 =0.019

Since, Safety is the highest we select Safety below Low branch.



Now we will check the value of 'Evaluation?' from the database, for all branches,

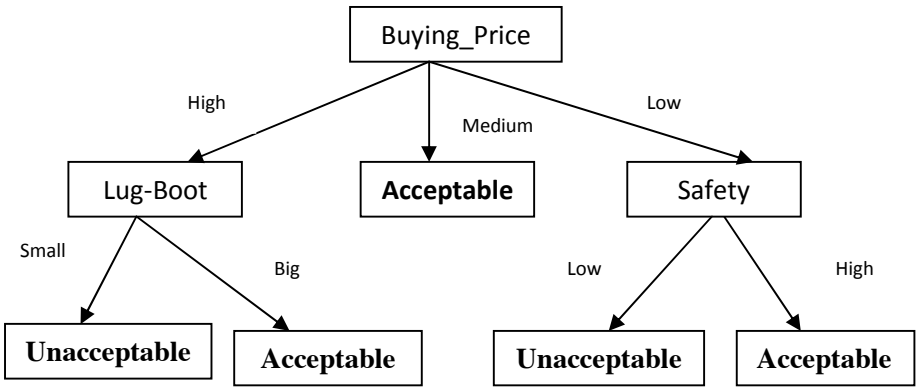
Buying_Price=High and Lug_Boot =Small -> Evaluation? = Unacceptable

Buying_Price=High and Lug_Boot =Big -> Evaluation? =Acceptable

Buying_Price=Low and Safety =Low -> Evaluation? = Unacceptable

Buying_Price= Low and Safety = High -> Evaluation? = Acceptable

Final Decision Tree is



10.2.9 Example of Decision Tree Using Gini Index:

Example1:

Create a decision tree using Gini Index to classify following dataset.

Sr.No.	Income	Age	Own Car
1	Very High	Young	Yes
2	High	Medium	Yes
3	Low	Young	No
4	High	Medium	Yes
5	Very High	Medium	Yes
6	Medium	Young	Yes
7	High	Old	Yes
8	Medium	Medium	No
9	Low	Medium	No
10	Low	Old	No
11	High	Young	Yes
12	Medium	Old	No

In this example there are two classes Yes and No.

No. Of records for Yes = 7

No. Of records for No = 5

Total No. Of records = 12

Now we will calculate Gini of the complete database as,

Gini (T) = $1 - \left(\left(\frac{7}{1}\right)^2 + \left(\frac{5}{1}\right)^2\right) = 0.48$

Next we will calculate Split for all attributes, i.e. Income and Age.

Income->

Split = $\frac{2}{1}g\ (V\quad h) + \frac{4}{1}g\ (H\quad h) + \frac{3}{1}g\ (L\ \ ) + \frac{3}{1}g\ (M\quad)$

$= \frac{2}{1}[\ 1 - \left(\left(\frac{2}{2}\right)^2 + \left(\frac{0}{2}\right)^2\right)] + \frac{4}{1}[\ 1 - \left(\left(\frac{4}{4}\right)^2 + \left(\frac{0}{4}\right)^2\right)] + \frac{3}{1}\Big[\ 1 - \left(\left(\frac{0}{3}\right)^2 + \left(\frac{3}{3}\right)^2\right)\Big] + \frac{3}{1}[\ 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right)]$

=0.1125

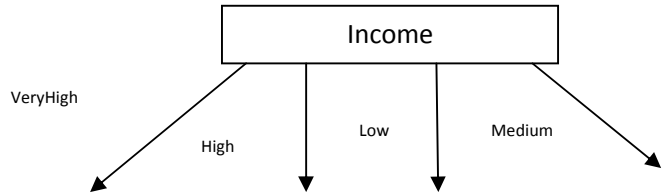
Age->

Split = $\frac{4}{1}g\ (Y\ \ ) + \frac{5}{1}g\ (M\quad) + \frac{3}{1}g\ (O\ d)$

$$= \frac{4}{1}\left[1-\left(\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right)\right]+\frac{5}{1}\left[1-\left(\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right)\right]+\frac{3}{1}\left[1-\left(\left(\frac{1}{3}\right)^2+\left(\frac{2}{3}\right)^2\right)\right]$$

=0.4375

Split value of Income is smallest, so we will select Income as root node.



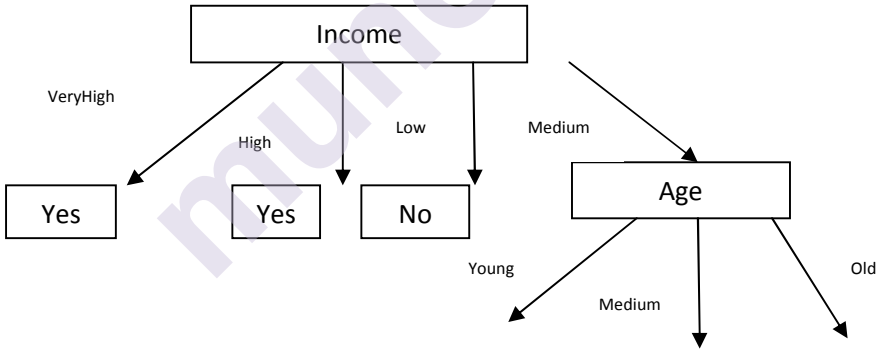
From the database we can see that,

Own Car = Yes for Income = Very High, so we can directly write down 'Yes' for Very High branch.

Own Car = Yes for Income = High, so we can directly write down 'Yes' for High branch.

Own Car = No for Income = Low, so we can directly write down 'No' for Low branch.

Since Income is taken as root node, now we have to decide on the Age attribute, so we will take Age as next node below Medium branch.



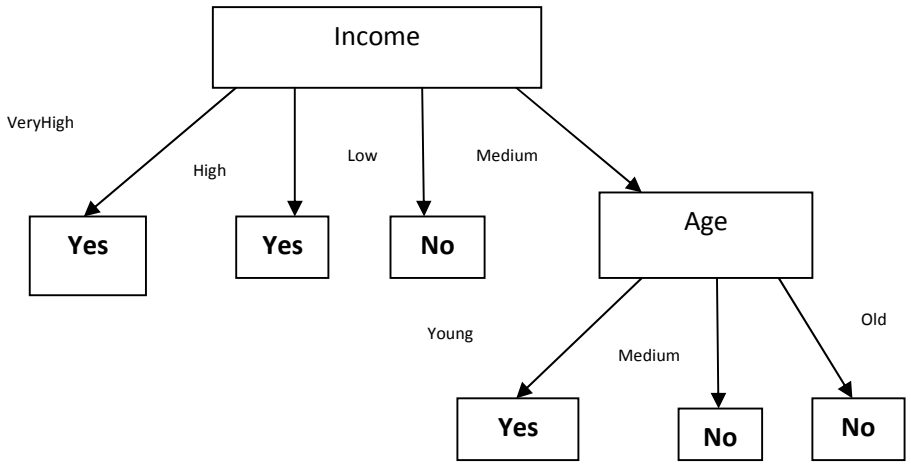
From the database we can see that,

Own Car = Yes for Income = Medium and Age = Young, so we can directly write down 'Yes' for Young branch.

Own Car = No for Income = Medium and Age = Medium, so we can directly write down 'No' for medium branch.

Own Car = No for Income = Medium and Age = Old, so we can directly write down 'No' for Old branch.

Final Decision Tree is,



10.3 RANKING AND PROBABILITY ESTIMATION TREES

- Decision trees divide the instance space into segments, by learning ordering on those segments the decision trees can be turned into rankers.
- Decision trees can access local class distribution, means how many number of +ve and –ve instances are there in each class.
- Decision trees are directly used to construct leaf ordering in an optimal way in the training data.
- One commonly used procedure is empirical probability ,means if +ve and –ve instances in a class are equal then give highest priority to positive class(means use ranking).
- The ranking obtained from the empirical probabilities in the leaves of a decision tree yields a convex ROC curve on the training data.

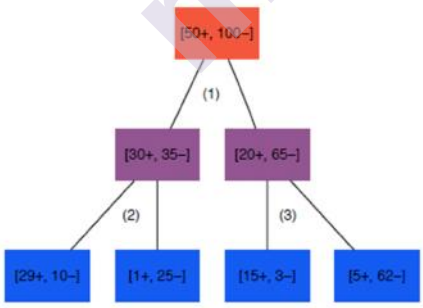


Figure 10.3.1.a

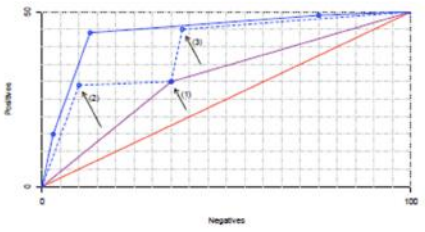


Figure 10.3.1.b

- Consider the tree in above figure 10.3.1.a. Each node is labelled with the numbers of positive and negative examples covered by it: so, for instance, the root of the tree is labelled with the overall class distribution (50 positives and 100 negatives), resulting in the trivial

ranking [50+,100-]. The corresponding one-segment coverage curve is the ascending diagonal figure 10.3.1.b.

- Adding split (1) refines this ranking into [30+,35-][20+,65-], resulting in a two-segment curve.

- Adding splits (2) and (3) again breaks up the segment corresponding to the parent into two segments corresponding to the children.

- However, the ranking produced by the full tree [15+,3-][29+,10-][5+,62-][1+,25-] is different from the left-to-right ordering of its leaves, hence we need to reorder the segments of the coverage curve, leading to the top-most, solid curve. This reordering always leads to a convex coverage curve

- Figure 10.3.1.a is an abstract representation of a tree with numbers of positive and negative examples covered in each node. Binary splits are added to the tree in the order indicated.

- Figure 10.3.1.b shows after adding a split to the tree how it will add new segments to the coverage curve as indicated by the arrows. After a split is added the segments may need reordering, and so only the solid lines represent actual coverage curves.

10.3.1 Choosing a labelling based on costs

- Assume the training set class ratio *clr* = 50/100 is representative. We have a choice of five labellings, depending on the expected cost ratio $C=C_{FN} / C_{FP}$ of misclassifying a positive in proportion to the cost of misclassifying a negative:

+-+- would be the labelling of choice if C= 1, or more generally if10/29 < C< 62/5;
+-++would be chosen if 62/5<C< 25/1;
++++would be chosen if 25/1 < C; i.e., we would always predict positive if false negatives are more than 25 times as costly as false positives, because then even predicting positive in the second leaf would reduce cost;
--+-would be chosen if 3/15<C< 10/29;

----would be chosen if C< 3/15; i.e., we would always predict negative if false positives are more than 5 times as costly as false negatives, because then even predicting negative in the third leaf would reduce cost.

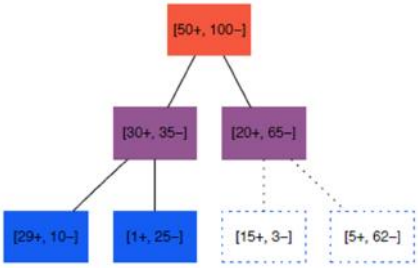


Figure 10.3.2.a

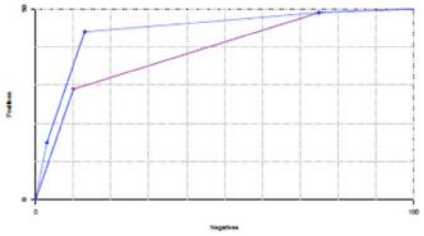


Figure 10.3.2.b

- Figure 10.3.2.ais used to achieve the labeling+-++ we don't need the right-most split, which can therefore be pruned away.
- In figure 10.3.2.bpruning doesn't affect the chosen operating point, but it does decrease the ranking performance of the tree.
- If we know class distribution in leaves of feature tree then it is converted in to Rankers, Probability estimation and Classifiers.
- Rankers will give simple order to leaves in descending order based on empirical probability
- Probability estimation predicts empirical probabilities of each leave and calculate Laplace or m estimation to smooth curve
- In classifier we have to choose the operating conditions and find operating point that fits the condition.
- First and foremost, I would concentrate on getting good ranking behaviour, because from a good ranker I can get good classification and probability estimation, but not necessarily the other way round.
- I would therefore try to use an impurity measure that is distribution-insensitive, such as pGini; if that isn't available and I can't hack the code, I would resort to oversampling the minority class to achieve a balanced class distribution.
- I would disable pruning and smooth the probability estimates by means of the Laplace correction (or the *m*-estimate).
- Once I know the deployment operation conditions, I would use these to select the best operating point on the ROC curve (i.e., a threshold on the predicted probabilities, or a labeling of the tree).
- (optional) Finally, I would prune away any sub tree whose leaves all have the same label.

10.4 REGRESSION TREES

Classification trees are used to divide the dataset into classes belonging to the target variable. Mainly the target variable has two classes that can be yes or no. When the target variable type is categorical classification trees are used.

In certain applications the target variable is numeric or continuous in that case regression trees are used. Let's take an example of prediction of price of a flat. Hence regression trees are used for problems or tasks where we want to predict some data instead of classifying the data.

Based on the similarity of the data the records are classified in a standard classification tree. Let's take an example of an Income tax evades. In this example we have two variables, Income and marital status that predict if a person is going to evade the income tax or not. In our training data if it showed that 85% of people who are married does not evade the income tax, we split the data here and Marital status becomes a root node in tree. Entropy or Ginny index is used in classification trees.

The main basic working of regression tree is to fit a model. The target or response variable does not have classes so a regression model is fit using each independent variable to the target variable. Then the data is split at various split points for each independent variable. At each split point sum of squared errors (SSE) is calculated by taking the square of the difference between predicted and actual value. The criteria for root node is to select the node which is having minimum SSE among all split point errors. The further tree is built using the recursive procedure.

10.4.1 Example of Regression Tree:

Example 1:

Buying_Price	Lug_Boot	Safety	Maintenance_ Price? (in thousand)
Low	Small	High	25
Low	Small	Low	30
Medium	Small	High	46
High	Small	High	45
High	Big	High	52
High	Big	Low	23
Medium	Big	Low	43
Low	Small	High	35
Low	Big	High	38
High	Big	High	46
Low	Big	Low	48
Medium	Small	Low	52
Medium	Big	High	44
High	Small	Low	30

Standard deviation ->
A decision tree is built up top down from root node and involved partitioning the data into subsets that contain instances with similar values. We use SD to calculate homogeneity of a numerical sample.

$$\text{SD, S} = \sqrt{\frac{\sum(x-\mu)^2}{n}} = 9.32$$

SD Reduction ->

It is based on the decrease in SD after a dataset is split on an attribute. Constructing a tree is all about finding attribute that returns highest SDR.

Step 1->
SD (Maintenance_Price?) = 9.32

Step 2->
The dataset is then split on the different attribute.SD for each branch is calculated. The resulting SD is subtracted from SD before split.

		Maintenance_ Price(SD)
Buying_Price	**Low**	7.78
	Medium	3.49
	High	10.87
SD(Maintenance_Price, Buying_Price) =P(Low)SD(Low)+P(Medium)SD(Medium)+P(High)SD(High) $=\frac{5}{1}*7.78+\frac{4}{1}*3.49+\frac{5}{1}*10.87=7.66$ SDR = SD(Maintenance_ Price)- SD(Maintenance_ Price, Buying_Price) = 9.32-7.66=1.66		

		Maintenance_ Price (SD)
Lug_Boot	**Small**	9.36
	Big	8.37
SD(Maintenance_ Price, Lug_Boot)=P(Small)SD(Small)+P(Big)SD(Big) $=\frac{7}{1}*9.36+\frac{7}{1}*8.37=8.86$ SDR= SD(Maintenance_ Price)- SD(Maintenance_ Price, Lug_Boot)= 9.32-8.86= 0.46		

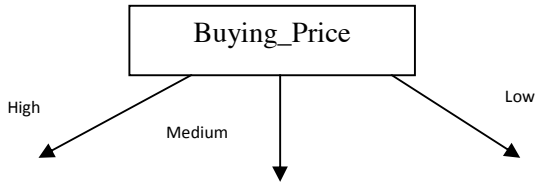
		Maintenance_ Price (SD)
Safety	**High**	7.87
	Low	10.59

SD(Maintenance_ Price, Safety)=P(High)SD(High)+P(Low)SD(Low)

$$= \frac{8}{1} * 7.87 + \frac{6}{1} * 10.59 = 9.02$$

SDR= SD(Maintenance_ Price)- SD(Maintenance_ Price, Safety)= 9.32-9.02=0.3

SDR of Buying_Price is highest so we select Buying_Price as our root node.



Step 2 ->

Now we will consider the records of 'High'. For High SD is 7.66 (which is not less than 50% global SD therefore branch will be splitted.

Buying_Price	Lug_Boot	Safety	Maintenance_ Price? (in thousand)
High	Small	High	45
High	Big	High	52
High	Big	Low	23
High	Big	High	46
High	Small	Low	30

For Buying_Price = High, SD=10.87
We will calculate SDR of only Lug_Boot and Safety

		Maintenance_ Price (SD)
Lug_Boot	**Small**	7.5
	Big	12.49

SD(High, Lug_Boot)=P(Small)SD(Small)+P(Big)SD(Big)

$$= \frac{2}{5} * 7.5 + \frac{3}{5} * 12.49 = 10.49$$

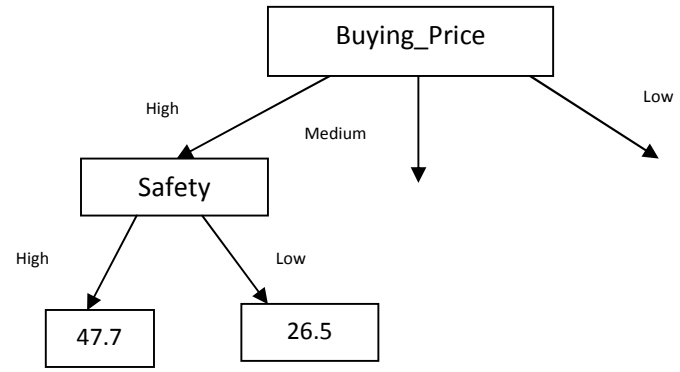
SDR= SD(High)- SD(Maintenance_ Price, Lug_Boot) = 10.87-10.49 =0.38

		Maintenance_ Price (SD)
Safety	**High**	3.09
	Low	3.50

SD(High, Safety)=P(High)SD(High)+P(Low)SD(Low)= $\frac{3}{5} * 3.09 + \frac{2}{5} * 3.5 = 3.25$

SDR= SD(High)- SD(Maintenance_ Price, Safety)= 10.87-3.25=7.62

SDR of Safety is highest so we select Safety as next node below High branch.



For Buying_Price =High and Safety= High, SD is 3.50 which is less than 50% SD of database, so we can directly write down the answer.

For Buying_Price =High and Safety= Low, SD is 3.09 which is less than 50% SD of database, so we can directly write down the answer.

To write down the answer we take average of values of following records,

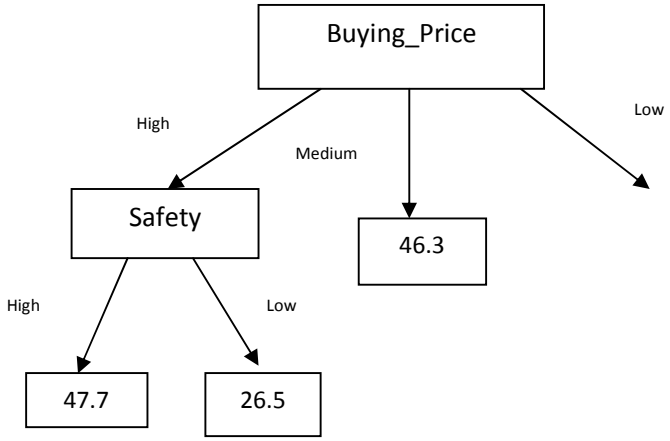
For Buying_Price =High and Safety= High, Maintenance _Price = 45+52+46/3= 47.7

For Buying_Price =High and Safety= Low, Maintenance _Price = 23+30/2= 26.5

Step 3->
Now we will consider the records of 'Medium'

Buying_Price	**Lug_Boot**	**Safety**	**Maintenance_ Price? (in thousand)**
Medium	Small	High	46
Medium	Big	Low	43
Medium	Small	Low	52
Medium	Big	High	44

For Buying_Price = Medium SD is 3.49 which is less than 50% SD of database, so we can directly write down the answer as 46.3. The answer is calculated by taking the average of values of Maintenance_ Price for Medium records (average of 46, 43, 52, and 44).



Step 4 ->

Now we will consider the records of 'Low'. For Low SD is 7.78 (which is not less than 50% global SD therefore branch will be splitted.

Buying_Price	**Lug_Boot**	**Safety**	**Maintenance_ Price? (in thousand)**
Low	Small	High	25
Low	Small	Low	30
Low	Small	High	35
Low	Big	High	38
Low	Big	Low	48

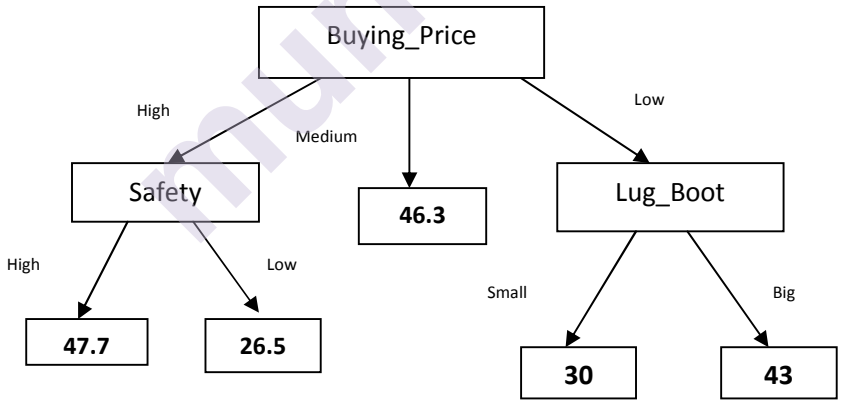
For Buying_Price= Low, SD=7.78

Now only Lug_Boot attribute is remaining, so we can directly take Lug_Boot as a next node below Low branch.

To write down the answer we take average of values of following records, For Buying_Price = Low and Lug_Boot = Small, Maintenance_Price = 25+30+35/3= 30

For Buying_Price = Low and Lug_Boot = Big, values of Maintenance_Price = 38+ 48/2=43.

Final Regression Tree is,



10.5 CLUSTERING TREES

Imagine you are a collector of vintage Hammond tonewheel organs. You have been monitoring an online auction site, from which you collected some data about interesting transactions:

Model	Condition	Leslie	Price	Resesrve	Bids
B3	Excellent	No	45	30	22
T202	Fair	Yes	6	0	9
A100	Good	No	11	8	13
T202	Good	No	3	0	1
M102	Good	Yes	9	5	2
A100	Excellent	No	18	15	15
T202	Fair	No	1	0	3
A100	Good	Yes	19	19	1
E112	Fair	No	1	0	5

- The means of the three numerical features are (13.3, 8.6,7.9) and their variances are (158,101.8,48.8). The average squared Euclidean distance to the mean is then the sum of these variances, which is 308.6.
- For the A100 cluster these vectors are (16,14,9.7) and (12.7,20.7,38.2),with average squared distance to the mean 71.6; for the T202 cluster they are (3.3, 0,4.3) and (4.2, 0,11.6), with average squared distance 15.8.
- Using this split we can construct a clustering tree whose leaves are labeled with the mean vectors (Figure 10.5.1).
- A clustering tree learned from the data in example using Euclidean distance on the numerical features.

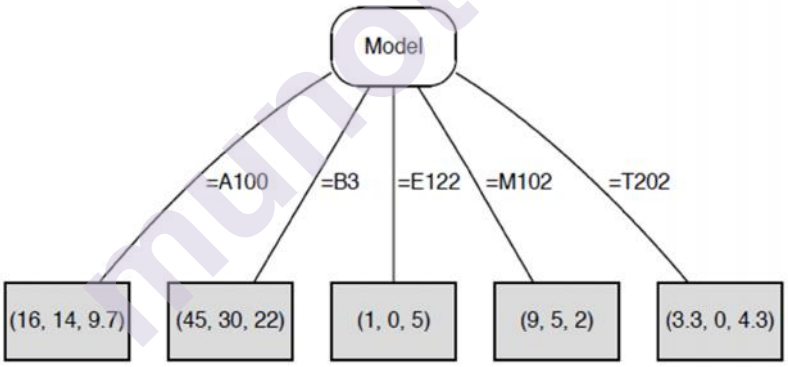


Figure 10.5.1

SUMMARY

In this chapter we have seen Tree model which can be seen as a particular type of rule model where the if-parts of the rules are organised in a tree structure. Decision trees are very strong and most suitable tools for classification and prediction. If we want to predict class of instances then we have to use decision trees whereas if we want to predict the value of target variable then we have to use regression tree.Rankers will give simple order to leaves in descending order based on empirical probability. Probability estimation predicts empirical probabilities of each leave and calculate Laplace or m- estimation to smooth curve. In classifier we have

to choose the operating conditions and find operating point that fits the condition. Getting good ranking behaviour, because from a good ranker. We can get good classification and probability estimation, but not necessarily the other way round.

UNIT END QUESTIONS

1. Create a decision tree for the attribute " class" using the respective values

Eyecolour	Married	Sex	Hairlength	class
Brown	yes	male	Long	Football
Blue	yes	male	Short	Football
Brown	yes	male	Long	Football
Brown	no	female	Long	Netball
Brown	no	female	Long	Netball
Blue	no	male	Long	Football
Brown	no	female	Long	Netball
Brown	no	male	Short	Football
Brown	yes	female	Short	Netball
Brown	no	female	Long	Netball
Blue	no	male	Long	Football
Blue	no	male	Short	Football

2. Write short note on Issues in Decision Tree

3. For the given data determine the entropy after classification using each attribute for classification seperatley and find which attribute is taken as decision attribute for the root by finding information gain with respect to entropy of Temperature as reference attribute.

Sr. no.	Temperature	Wind	Humidity
1	Hot	Weak	High
2	Hot	Strong	High
3	Mild	Weak	Normal
4	Cool	Strong	High
5	Cool	Weak	Normal
6	Mild	Strong	Normal
7	Mild	Weak	High
8	Hot	Strong	High
9	Mild	Weak	Normal
10	Hot	Strong	Normal

4. For a Sunburn dataset given below, construct a decision tree

Name	Hair	Height	Weight	Location	Class
Swati	Blonde	Average	Light	No	Yes
Sunita	Blonde	Tall	Average	Yes	No
Anita	Brown	Short	Average	Yes	No
Lata	Blonde	Short	Average	No	Yes
Radha	Red	Average	Heavy	No	Yes
Maya	Brown	Tall	Heavy	No	No
Leena	Brown	Average	Heavy	No	No
Rina	Blonde	Short	Light	Yes	No

LIST OF REFERENCES

- https://www.kdnuggets.com/2019/06/main-approaches-machine-learning-models.html
- Foster Provost, pedro Domingos," Tree Induction for Probability-Based Ranking", Machine Learning,
- SpringerLink, volume 52, pages199–215 (2003)
- https://www.solver.com/regression-trees
- Luke Zappia, Alicia Oshlack, "Clustering trees: a visualization for evaluating clusterings at multiple resolutions", *GigaScience*, Volume 7, Issue 7, July 2018, giy083,

*****

UNIT V

11

PROBABILISTIC MODEL

Unit Structure

11.0 OBJECTIVE

Probabilistic modelling provides a framework for understanding what learning is, and has therefore emerged as one of the principal theoretical and practical approaches for designing machines that learn from data

acquired through experience. The probabilistic framework, which describes how to represent and manipulate uncertainty about models and predictions, has a central role in scientific data analysis, machine learning, robotics, cognitive science and artificial intelligence.

11.1 INTRODUCTION: PROBABILISTIC MODEL

Probabilistic models are widely used in text mining and applications range from topic modelling, language modelling, document classification and clustering to information extraction. And the examples of models are the: topic modelling methods PLSA and LDA are special applications of mixture models.

The probabilistic model is a model that uses probability theory to model the uncertainty in the data. Like terms in topics are modelled by multinomial distribution and the observations for a random field are modelled by Gibbs distribution. Which are the major probabilistic models being Mixture and Models Mixture models are used for clustering data points, where each component is a distribution for that cluster, and each data point belongs to one cluster with a certain probability. Finite mixture models require the user to specify the number of clusters. The typical applications of mixture models in text mining include topic models, like PLSA and LDA. Bayesian Nonparametric Models Bayesian nonparametric models refer to probabilistic models with infinite-dimensional parameters, which usually have a stochastic process that is infinite-dimensional as the prior distribution. Infinite mixture model is one type of nonparametric model, which can deal with the problem of selecting the number of clusters for clustering. The Dirichlet process mixture model belongs to the infinite mixture model, and can help to detect the number of topics in topic modelling.

Mixture Models: Mixture model is a probabilistic model originally proposed to address the multi-modal problem in the data, and now is frequently used for the task of clustering in data mining, machine learning and statistics. defines the distribution of a random variable, which contains multiple components and each component represents a different distribution following the same distribution family with different parameters. Mixture Model Basic framework of mixture models, Their variations and applications in text mining area, and the standard learning algorithms for them. General Mixture Model Framework In a mixture model, given a set of data points, e.g., the height of people in a region, they are treated as an instantiation of a set of random variables. According to the observed data points, the parameters in the mixture model can be learned. For example, we can learn the mean and standard deviation for female and male height distributions, if we model height of people as a mixture model of two Gaussian distributions. Assume n random variables X1,X2, . . . , xn with observations x1, x2, . . . , xn, following the mixture model with K components.

11.1.2 Normal Distribution and its Geometric Interpretation:

Normal Distribution is an important concept in statistics and the backbone of Machine Learning. A Data Scientist needs to know about Normal Distribution when they work with Linear Models (perform well if the data is normally distributed), Central Limit Theorem, and exploratory data analysis. As discovered by **Carl Friedrich Gauss**, **Normal Distribution/Gaussian Distribution** is a continuous probability distribution. It has a bell-shaped curve that is symmetrical from the mean point to both halves of the curve.

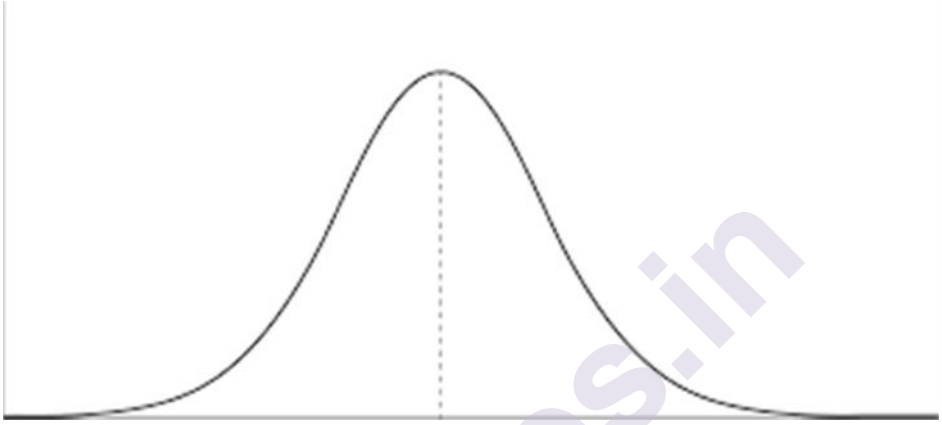


Figure 11.1.2 Normal Distribution and its Geometric Interpretation

A continuous random variable "x" is said to follow a normal distribution with parameter μ(mean) and σ(standard deviation), if it's probability density function is given by,

$$y = \frac{1}{\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation
$\pi \approx 3.14159$
$e \approx 2.71828$

This is also called a **normal variate.**

11.2 STANDARD NORMAL VARIATE

If "x" is a normal variable with a mean(μ) and a standard deviation(σ) then,

$$Z = \frac{X - \mu}{\sigma}$$

where z = standard normal variate

11.2.1 Standard Normal Distribution:

The simplest case of the normal distribution, known as the Standard Normal Distribution, has an expected value of μ(mean) 0 and σ(s.d.) 1, and is described by this probability density function,

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

Where $-\infty < z < \infty$

Distribution Curve Characteristics:

1. The total **area under** the **normal curve** is equal to 1.
2. It is a continuous distribution.
3. It is symmetrical about the mean. Each half of the distribution is a mirror image of the other half.
4. It is asymptotic to the horizontal axis.
5. It is unimodal.

Area Properties:

The normal distribution carries with it assumptions and can be completely specified by two parameters: the mean and the standard deviation. If the mean and standard deviation are known, you can access every data point on the curve.

The empirical rule is a handy quick estimate of the data's spread given the mean and standard deviation of a data set that follows a normal distribution. It states that:

- 68.26% of the data will fall within 1 sd of the mean(μ±1σ)
- 95.44% of the data will fall within 2 sd of the mean(μ±2σ)
- 99.7% of the data will fall within 3 sd of the mean(μ±3σ)
- 95% — (μ±1.96σ)
- 99% — (μ±2.75σ)

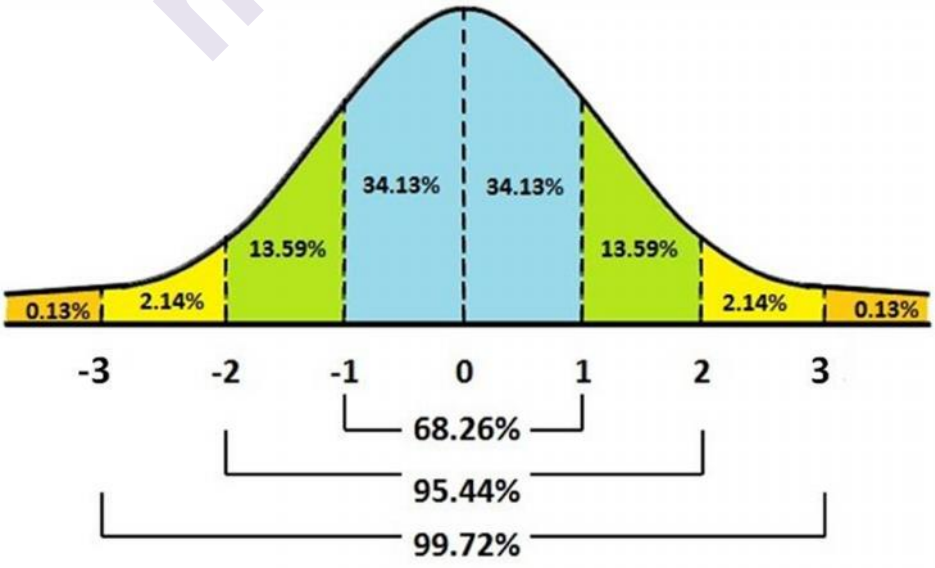


Figure 11.2.1 slandered normal distribution

Thus, almost all the data lies within **3 standard deviations**. This rule enables us to check for **Outliers** and is very helpful when determining the normality of any distribution.

11.2.2 Application in Machine Learning:

In Machine Learning, data satisfying Normal Distribution is beneficial for model building. It makes math easier. Models like LDA, Gaussian Naive Bayes, Logistic Regression, Linear Regression, etc., are explicitly calculated from the assumption that the distribution is a bivariate or multivariate normal. Also, *Sigmoid functions* work most naturally with normally distributed data. Many natural phenomena in the world follow a log-normal distribution, such as *financial data* and *forecasting data*. By applying transformation techniques, we can convert the data into a normal distribution. Also, many processes follow normality, such as many *measurement errors in an experiment, the position of a particle that experiences diffusion, etc*. So, it's better to critically explore the data and check for the underlying distributions for each variable before going to fit the model.

Visualization Techniques:

```
df = pd.read_csv("boston-dataset/boston_data.csv")
df.head(2)
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.15876	0.0	10.81	0.0	0.413	5.961	17.5	5.2873	4.0	305.0	19.2	376.94	9.88	21.7
1	0.10328	25.0	5.13	0.0	0.453	5.927	47.2	6.9320	8.0	284.0	19.7	396.90	9.22	19.6

11.2.2.1 Histograms:

It is a kind of bar graph which is an estimate of the probability distribution of a continuous variable. It defines numerical data *and divided them into uniform bins which are consecutive, non-overlapping intervals of a variable*.

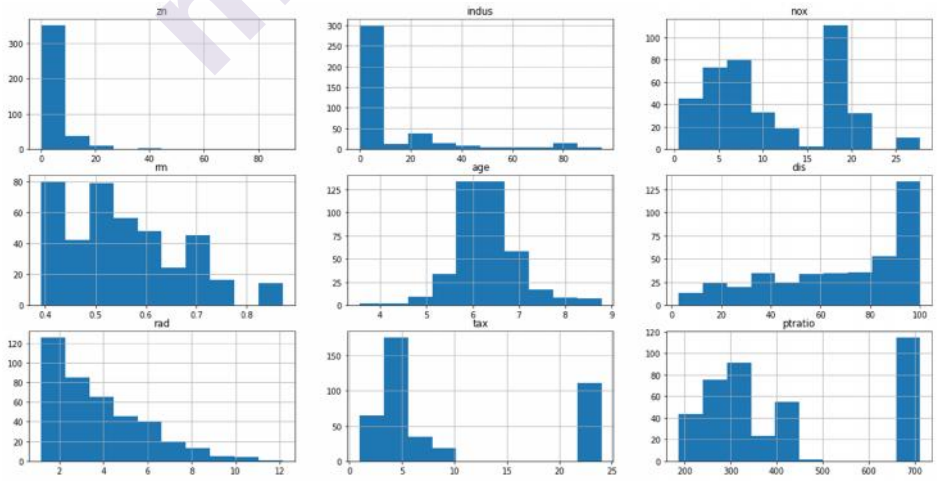


Figure 11.2.2.1Histograms1

kdeplot:
It is a Kernel Distribution Estimation Plot which depicts the probability density function of the continuous or non-parametric data variables i.e. we can plot for the univariate or multiple variables altogether.

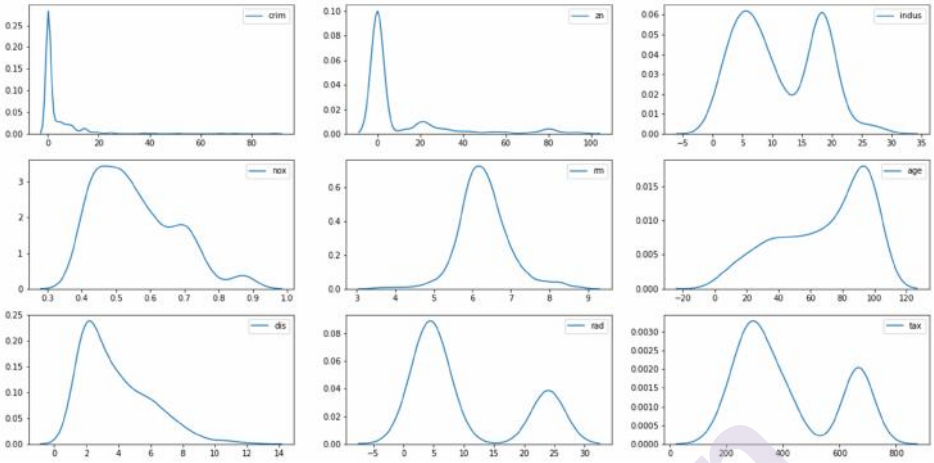
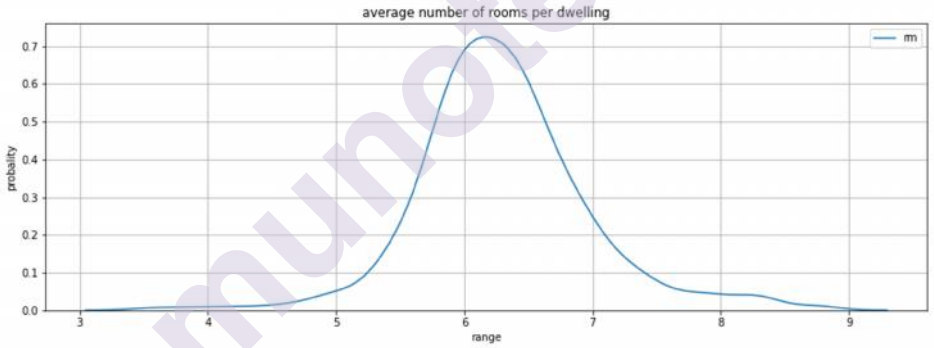


Figure 11.2.2.1Histograms2

11.2.3 Feature Analysis:

Let's take an example of feature rm(average number of rooms per dwelling) closely resembling a normal distribution.



Though it has some distortion in the right tail, We need to check how close it resembles a normal distribution. For that, we need to check the ***Q-Q Plot***.When the quantiles of two variables are plotted against each other, then the plot obtained is known as quantile — quantile plot or plot. This plot provides a summary of whether the distributions of two variables are similar or not with respect to the locations.

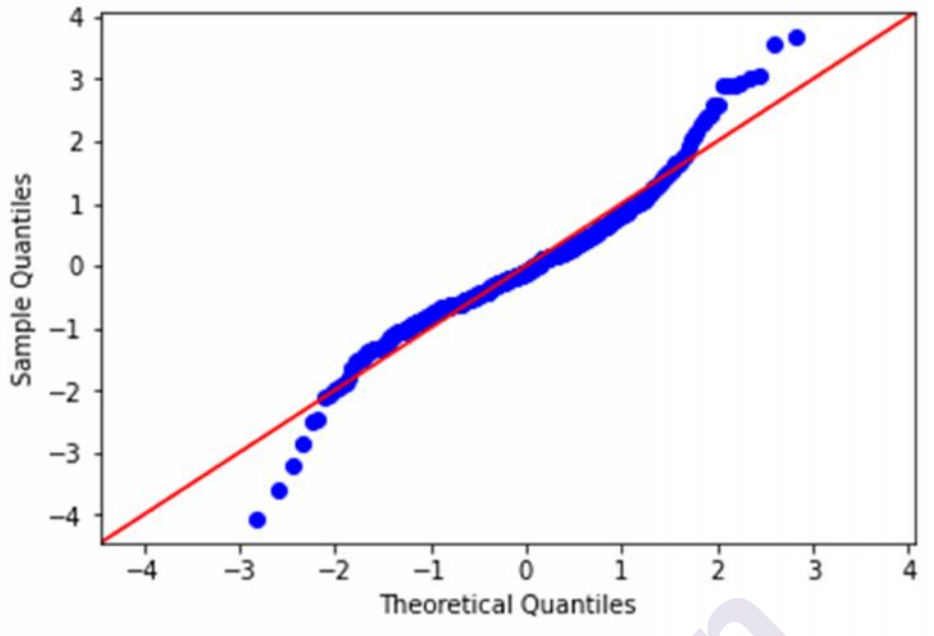


Figure 11.2.3 plots variable

Here we can clearly see that feature is not normally distributed. But it somewhat resembles it. We can conclude that standardizing (StandardScaler) this feature before feeding it to a model can generate a good result.

11.2.4 Central Limit Theorem and Normal Distribution:

CLT states that when we add a large number of independent random variables to a dataset, irrespective of these variables' original distribution, their normalized sum tends towards a Gaussian distribution. Machine Learning models generally treat training data as a mix of *deterministic* and *random* parts. Let the *dependent variable(Y)* consists of these parts. Models always want to express the *dependent variables(Y)* as some function of several *independent variables(X)*. If the function is sum (or expressed as a sum of some other function) and the number of ***X*** is really high, then ***Y*** should have a normal distribution. Here ml models try to express the *deterministic part as a sum of deterministic independent variables(X):**deterministic + random = fun (deterministic*** (1)) +…+ fun(deterministic(n)) + model error If the whole deterministic part of ***Y*** is explained by ***X,*** then the ***model error*** depicts only the ***random part*** and should have a normal distribution. So if the error distribution is normal, then we may suggest that the model is successful. Else some other features are absent in the model but have a large enough influence on ***Y,*** or the model is incorrect.

11.2.5 Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.

- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

11.2.6 Why is it called Naïve Bayes:

- The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, which can be described as:
- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.
- Bayes' Theorem:
- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

Outlook	Play	
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Table 11.2.6 :Frequency table for the Weather Conditions

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Table 11.2.6 :Frequency table for the Weather Conditions solution

11.2.7 Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.
- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

11.2.8 Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.

- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

11.2.9 Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

11.2.10 Descriptive Learning maximum like hood:

Density estimation is the problem of estimating the probability distribution for a sample of observations from a problem domain. There are many techniques for solving density estimation, although a common framework used throughout the field of machine learning is maximum likelihood estimation. Maximum likelihood estimation involves defining a likelihood function for calculating the conditional probability of observing the data sample given a probability distribution and distribution parameters. This approach can be used to search a space of possible distributions and parameters. This flexible probabilistic framework also provides the foundation for many machines learning algorithms, including important methods such as linear regression and logistic regression for predicting numeric values and class labels respectively, but also more generally for deep learning artificial neural networks.

- Maximum Likelihood Estimation is a probabilistic framework for solving the problem of density estimation.
- It involves maximizing a likelihood function in order to find the probability distribution and parameters that best explain the observed data.
- It provides a framework for predictive modelling in machine learning where finding model parameters can be framed as an optimization problem.

Descriptive Learning maximum like hood divided into three parts:

1. Problem of Probability Density Estimation

2. Maximum Likelihood Estimation
3. Relationship to Machine Learning

11.2.11 Problem of Probability Density Estimation:

A common modelling problem involves how to estimate a joint probability distribution for a dataset.

For example, given a sample of observation (*X*) from a domain (*x1, x2, x3, ...,xn*), where each observation is drawn independently from the domain with the same probability distribution (so-called independent and identically distributed, i.i.d., or close to it).

Density estimation involves selecting a probability distribution function and the parameters of that distribution that best explain the joint probability distribution of the observed data (*X*).

- How do you choose the probability distribution function?
- How do you choose the parameters for the probability distribution function?

This problem is made more challenging as sample (*X*) drawn from the population is small and has noise, meaning that any evaluation of an estimated probability density function and its parameters will have some error.

- There are many techniques for solving this problem, although two common approaches are:
- Maximum a Posteriori (MAP), a Bayesian method.
- Maximum Likelihood Estimation (MLE), frequentist method.
- The main difference is that MLE assumes that all solutions are equally likely beforehand, whereas MAP allows prior information about the form of the solution to be harnessed.
- In this post, we will take a closer look at the MLE method and its relationship to applied machine learning.

11.2.12 Maximum Likelihood Estimation:

One solution to probability density estimation is referred to as Maximum Likelihood Estimation, or MLE for short.

Maximum Likelihood Estimation involves treating the problem as an optimization or search problem, where we seek a set of parameters that results in the best fit for the joint probability of the data sample (*X*).

First, it involves defining a parameter called *theta* that defines both the choice of the probability density function and the parameters of that distribution. It may be a vector of numerical values whose values change smoothly and map to different probability distributions and their parameters.

In Maximum Likelihood Estimation, we wish to maximize the probability of observing the data from the joint probability distribution given a specific probability distribution and its parameters, stated formally as:

- P(X | theta)

This conditional probability is often stated using the semicolon (;) notation instead of the bar notation (|) because *theta* is not a random variable, but instead an unknown parameter. For example:

- P(X ; theta)

or

- P(x1, x2, x3, …, xn ; theta)

This resulting conditional probability is referred to as the likelihood of observing the data given the model parameters and written using the notation *L()* to denote the likelihood function. For example:

- L(X ; theta)

The objective of Maximum Likelihood Estimation is to find the set of parameters (*theta*) that maximize the likelihood function, e.g. result in the largest likelihood value.

- maximize L(X ; theta)We can unpack the conditional probability calculated by the likelihood function.

Given that the sample is comprised of n examples, we can frame this as the joint probability of the observed data samples *x1, x2, x3, ...,xn* in *X* given the probability distribution parameters (*theta*).

- L(x1, x2, x3, …, xn ; theta)

The joint probability distribution can be restated as the multiplication of the conditional probability for observing each example given the distribution parameters.

- product i to n P(xi ; theta)

Multiplying many small probabilities together can be numerically unstable in practice, therefore, it is common to restate this problem as the sum of the log conditional probabilities of observing each example given the model parameters.

- sum i to n log (P(xi ; theta))

Where log with base-e called the natural logarithm is commonly used.

11.2.13 Relationship to Machine Learning:

This problem of density estimation is directly related to applied machine learning. We can frame the problem of fitting a machine learning model as the problem of probability density estimation. Specifically, the choice of

model and model parameters is referred to as a modelling hypothesis *h*, and the problem involves finding *h* that best explains the data *X*.

- P(X ; h)

We can, therefore, find the modelling hypothesis that maximizes the likelihood function.

- maximize L(X ; h)

Or, more fully:

- maximize sum i to n log(P(xi ; h))

This provides the basis for estimating the probability density of a dataset, typically used in unsupervised machine learning algorithms.

11.2.14 Probabilistic model with hidden variable:

Mathematics is the foundation of Machine Learning, and its branches such as Linear Algebra, Probability, and Statistics can be considered as integral parts of ML. As a Computer Science and Engineering student, one of the questions I had during my undergraduate days was in which ways the knowledge that was acquired through math courses can be applied to ML and what are the areas of mathematics that play a fundamental role in ML. I believe this is a common question among most of the people who are interested in Machine Learning. Therefore, I decided to write a blog series on some of the basic concepts related to "Mathematics for Machine Learning". In this series, my intention is to provide some directions into which areas to look at and explain how those concepts are related to ML. I am not going deep into the concepts and I believe there are a lot of resources with quite good examples that explain each of these concepts in a detailed manner.

As the first step, I would like to write about the relationship between probability and machine learning. In machine learning, there are probabilistic models as well as non-probabilistic models. In order to have a better understanding of probabilistic models, the knowledge about basic concepts of probability such as random variables and probability distributions will be beneficial. I will write about such concepts in my next blog. However, in this blog, the focus will be on providing some idea on what are probabilistic models and how to distinguish whether a model is probabilistic or not.

What are Probabilistic Machine Learning Models?

In order to understand what is a probabilistic machine learning model, let's consider a classification problem with N classes. If the classification model (classifier) is probabilistic, for a given input, it will provide probabilities for each class (of the N classes) as the output. In other words, a probabilistic classifier will provide a probability distribution over the N classes. Usually, the class with the highest probability is then selected as the Class for which the input data instance belongs.

However, logistic regression (which is a probabilistic binary classification technique based on the Sigmoid function) can be considered as an exception, as it provides the probability in relation to one class only (usually Class 1, and it is not necessary to have "*1 — probability of Class1 = probability of Class 0"* relationship). Because of these properties, Logistic Regression is useful in Multi-Label Classification problems as well, where a single data point can have multiple class labels.

Some examples for probabilistic models are Logistic Regression, Bayesian Classifiers, Hidden Markov Models, and Neural Networks (with a SoftMax output layer).If the model is Non-Probabilistic (Deterministic), it will usually output only the most likely class that the input data instance belongs to. Vanilla "Support Vector Machines" is a popular non-probabilistic classifier.

Let's discuss an example to better understand probabilistic classifiers. Take the task of classifying an image of an animal into five classes — {Dog, Cat, Deer, Lion, Rabbit} as the problem. As input, we have an image (of a dog). For this example, let's consider that the classifier works well and provides correct/ acceptable results for the particular input we are discussing. When the image is provided as the input to the probabilistic classifier, it will provide an output such as (Dog (0.6), Cat (0.2), Deer(0.1), Lion(0.04), Rabbit(0.06)). But, if the classifier is non-probabilistic, it will only output "Dog".

11.3 WHY PROBABILISTIC ML MODELS

One of the major advantages of probabilistic models is that they provide an idea about the uncertainty associated with predictions. In other words, we can get an idea of how confident a machine learning model is on its prediction. If we consider the above example, if the probabilistic classifier assigns a probability of 0.9 for 'Dog' class instead of 0.6, it means the classifier is more confident that the animal in the image is a dog. These concepts related to uncertainty and confidence are extremely useful when it comes to critical machine learning applications such as disease diagnosis and autonomous driving. Also, probabilistic outcomes would be useful for numerous techniques related to Machine Learning such as Active Learning.

11.3.1 Objective Functions:

In order to identify whether a particular model is probabilistic or not, we can look at its Objective Function. In machine learning, we aim to optimize a model to excel at a particular task. The aim of having an objective function is to provide a value based on the model's outputs, so optimization can be done by either maximizing or minimizing the particular value. In Machine Learning, usually, the goal is to minimize prediction error. So, we define what is called a loss function as the objective function and tries to minimize the loss function in the training phase of an ML model.

If we take a basic machine learning model such as Linear Regression, the objective function is based on the squared error. The objective of the training is to minimize the Mean Squared Error / Root Mean Squared Error (RMSE) (Eq. 1). The intuition behind calculating Mean Squared Error is, the loss/ error created by a prediction given to a particular data point is based on the difference between the actual value and the predicted value (note that when it comes to Linear Regression, we are talking about a regression problem, not a classification problem).The loss created by a particular data point will be higher if the prediction gives by the model is significantly higher or lower than the actual value. The loss will be less when the predicted value is very close to the actual value. As you can see, the objective function here is not based on probabilities, but on the difference (absolute difference) between the actual value and the predicted value.

$$\text{RMSE} = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}\left(y_{true} - y_{predict}\right)^2}$$

Here, n indicates the number of data instances in the data set, true is the correct/ true value and predict is the predicted value (by the linear regression model).

When it comes to Support Vector Machines, the objective is to maximize the margins or the distance between support vectors. This concept is also known as the 'Large Margin Intuition'. As you can see, in both Linear Regression and Support Vector Machines, the objective functions are not based on probabilities. So, they can be considered as non-probabilistic models. On the other hand, if we consider a neural network with a SoftMax output layer, the loss function is usually defined using Cross-Entropy Loss (CE loss) (Eq. 2). Note that we are considering a training dataset with 'n' number of data points, so finally take the average of the losses of each data point as the CE loss of the dataset. Here, Yi means the true label of the data point i and p(Yi) means the predicted probability for the class Yi (probability of this data point belongs to the class Yi as **assigned by the model**).

The intuition behind Cross-Entropy Loss is ; if the probabilistic model is able to predict the correct class of a data point with high confidence, the loss will be less. In the example we discussed about image classification, if the model provides a probability of 1.0 to the class 'Dog' (which is the correct class), the loss due to that prediction = -log(P('Dog')) = -log(1.0)=0. Instead, if the predicted probability for 'Dog' class is 0.8, the loss = -log(0.8)= 0.097. However, if the model provides a low probability for the correct class, like 0.3, the loss = -log(0.3) = 0.523, which can be considered as a significant loss.

$$CELoss = -\frac{1}{n}\sum_{i=1}^{n}(log(p(y_i))$$

In a binary classification model based on Logistic Regression, the loss function is usually defined using the Binary Cross Entropy loss (BCE loss).Here y_i is the class label (1 if similar, 0 otherwise) and p(s_i) is the predicted probability of a point being class 1 for each point 'i' in the dataset. N is the number of data points. Note that as this is a binary classification problem, there are only two classes, class 1 and class 0.As you can observe, these loss functions are based on probabilities and hence they can be identified as probabilistic models. Therefore, if you want to quickly identify whether a model is probabilistic or not, one of the easiest ways is to analyse the loss function of the model.

So, that's all for this article. I hope you were able to get a clear understanding of what is meant by a probabilistic model. In the next blog, I will explain some probability concepts such as probability distributions and random variables, which will be useful in understanding probabilistic models. If you find anything written here which you think is wrong, please feel free to comment.

11.3.2 Maximization method:

In the real-world applications of machine learning, it is very common that there are many relevant features available for learning but only a small subset of them are observable. So, for the variables which are sometimes observable and sometimes not, then we can use the instances when that variable is visible is observed for the purpose of learning and then predict its value in the instances when it is not observable. On the other hand, ***Expectation-Maximization algorithm*** can be used for the latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables) too in order to predict their values with the condition that the general form of probability distribution governing those latent variables is known to us. This algorithm is actually at the base of many unsupervised clustering algorithms in the field of machine learning.
It was explained, proposed and given its name in a paper published in 1977 by Arthur Dempster, Nan Laird, and Donald Rubin. It is used to find the *local maximum likelihood parameters* of a statistical model in the cases where latent variables are involved and the data is missing or incomplete.

1. Algorithm:

1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.
3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.

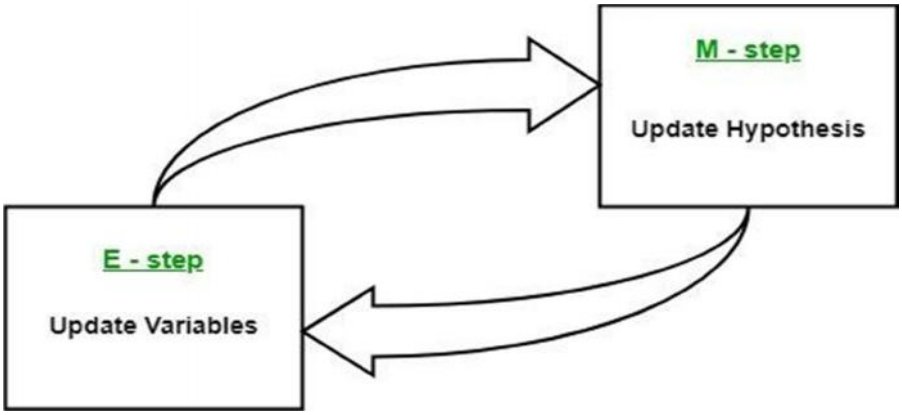


Figure 11.3.2 statistics of parameter

- The essence of Expectation-Maximization algorithm is to use the available observed data of the dataset to estimate the missing data and then using that data to update the values of the parameters. Let us understand the EM algorithm in detail.
- Initially, a set of initial values of the parameters are considered. A set of incomplete observed data is given to the system with the assumption that the observed data comes from a specific model.
- The next step is known as "Expectation" – step or *E-step*. In this step, we use the observed data in order to estimate or guess the values of the missing or incomplete data. It is basically used to update the variables.
- The next step is known as "Maximization"-step or *M-step*. In this step, we use the complete data generated in the preceding "Expectation" – step in order to update the values of the parameters. It is basically used to update the hypothesis.
- Now, in the fourth step, it is checked whether the values are converging or not, if yes, then stop otherwise repeat *step-2* and *step-3* i.e. "Expectation" – step and "Maximization" – step until the convergence occurs.

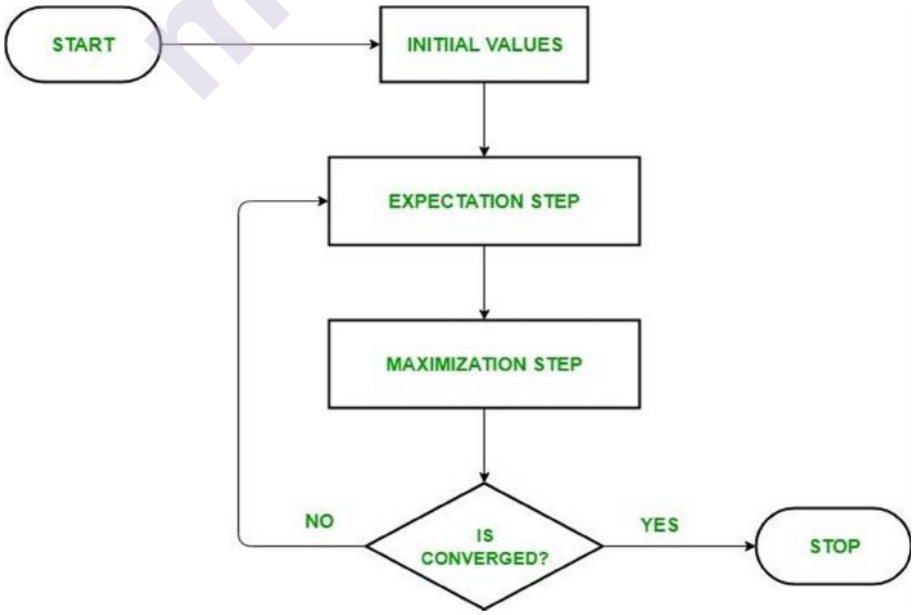


Figure 11.3.2 Algorithm of maximization

11.3.3 Usage of EM algorithm:

- It can be used to fill the missing data in a sample.
- It can be used as the basis of unsupervised learning of clusters.
- It can be used for the purpose of estimating the parameters of Hidden Markov Model (HMM).
- It can be used for discovering the values of latent variables.

11.3.4 Advantages of EM algorithm:

- It is always guaranteed that likelihood will increase with each iteration.
- The E-step and M-step are often pretty easy for many problems in terms of implementation.
- Solutions to the M-steps often exist in the closed form.

11.3.5 Disadvantages of EM algorithm:

- It has slow convergence.
- It makes convergence to the local optima only.
- It requires both the probabilities, forward and backward (numerical optimization requires only forward probability).

11.4 GAUSSIAN MIXTURE MODEL & COMPRESSION BASED MODEL

Suppose there are set of data points that needs to be grouped into several parts or clusters based on their similarity. In machine learning, this is known as Clustering.

- There are several methods available for clustering like:
- K Means Clustering
- Hierarchical Clustering
- Gaussian Mixture Models

Normal or Gaussian Distribution:
In real life, many datasets can be modeled by Gaussian Distribution (Univariate or Multivariate). So it is quite natural and intuitive to assume that the clusters come from different Gaussian Distributions. Or in other words, it is tried to model the dataset as a mixture of several Gaussian Distributions. This is the core idea of this model.

11.4.1 Gaussian Mixture Model:

Data scientists use various machine learning algorithms to discover patterns in large data that can lead to actionable insights. In general, high-dimensional data are reduced by obtaining a set of principal components so as to highlight similarities and differences. In this work, we deal with

the reduced data using a bivariate mixture model and learning with a bivariate Gaussian mixture model. We discuss a heuristic for detecting important components by choosing the initial values of location parameters using two different techniques: cluster means, *k*-means and hierarchical clustering, and default values in the "mixtools" R package. The parameters of the model are obtained via an expectation maximization algorithm. The criteria from Bayesian point are evaluated for both techniques, demonstrating that both techniques are efficient with respect to computation capacity. The effectiveness of the discussed techniques is demonstrated through a simulation study and using real data sets from different fields.

In real data such as engineering data, efficient dimension reduction is required to reveal underlying patterns of information. Dimension reduction can be used to convert data sets containing millions of functions into manageable spaces for efficient processing and analysis. Unsupervised learning is the main approach to reducing dimensionality. Conventional dimensional reduction approaches can be combined with statistical analysis to improve the performance of big data systems [1]. Many dimension reduction techniques have been developed by statistical and artificial intelligence researchers. Principal component analysis (PCA), introduced in 1901 by Pearson [2], is one of the most popular of these methods. The main purpose of PCA is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. Among the many PCA methods, singular value decomposition is used in numerical analysis and Korhonen–Loève expansion in electrical engineering. Eigenvector analysis and characteristic vector analysis are often used in the physical sciences. In image analysis, the Hotelling transformation is often used for principal component projection.

In recent years, there has been increasing interest in PCA mixture models. Mixture models provide a useful framework for the modelling of complex data with a weighted component distribution. Owing to their high flexibility and efficiency, they are used widely in many fields, including machine learning, image processing, and data mining. However, because the component distributions in a mixture model are commonly formalized as probability density functions, implementations in high-dimensional spaces are constrained by practical considerations.PCA mixture models are based on a mixture-of-experts technique, which models a nonlinear distribution through a combination of local linear sub models, each with a fairly simple distribution [3]. For the selection of the model, a PCA mixture model was proposed by Kim, Kim, and Bang [4], which has a more straightforward expectation maximization (EM) calculation, does not require a Gaussian error term for each mixture component, and uses an efficient technique for model order selection. The researchers applied the proposed model to the classification of synthetic data and eye detection [4].

For multimode processes, the Gaussian mixture model (GMM) was developed to estimate the probability density function of the process data under normal operating conditions. However, in the case of high and collinear process variables, learning from process data with GMM can be difficult or impossible. A novel multimode monitoring approach based on the PCA mixture model was proposed by Xu, Xie, and Wang [5] to address this issue. In this method, first, the PCA technique is applied directly to each Gaussian component's covariance matrix to reduce the dimension of process variables and to obtain non-singular covariance matrices. Then, an EM algorithm is used to automatically optimize the number of mixture components. A novel process monitoring scheme for the detection of multimode processes was developed using the resulting PCA mixture model. The monitoring performance of the proposed approach has been evaluated through case studies [5].In recent years, hyperspectral imaging has become an important research subject in the field of remote sensing. An important application of hyperspectral imaging is the identification of land cover areas. The rich content of hyperspectral data enables forests, urban areas, crop species, and water supplies to be recognized and classified. In 2016, Kutluk, Kayabol, and Akan [6] proposed a supervised classification and dimensionality reduction method for hyperspectral images, using a mixture of probability PCA (PPCA) models. The proposed mixture model simultaneously allows the reduction of dimensionality and spectral classification of the hyperspectral image. Experimental findings obtained using real hyperspectral data indicate that the proposed approach results in better classification than the state-of-the-art methods [6].

PROBABILITY CONCEPTS - SUMMARY

1. Probabilities of outcomes.

Experiment = observe something

Outcome = one of possible things we can observe

Sample space = S = set of possible outcomes = $\{a_1, \ldots, a_n\}$

$$Pr\{a\} = \lim_{N\to\infty} \frac{\#\text{ times } a \text{ occurs in } N \text{ repetitions}}{N}$$

for each outcome a (Intuitive defn)
$Pr\{a\} \geq 0$

for each a

$$\sum_{i=1}^{n} Pr\{a_i\} = 1.$$

Example 1. Company makes diodes. Pick a diode from production line.
S = {defective, good} = {d, g}.
Probability diode is defective = $Pr\{d\}$ = 0.003,
$Pr\{g\}$ = 0.997.

$$\frac{\#\text{ defective}}{\#\text{ diodes}} \to Pr\{d\} \qquad \text{as \# diodes} \to \infty,$$

Example 2. Roll a die.
S = {1, 2, 3, 4, 5, 6}.
Probability of 1 = $Pr\{ 1 \}$ = 1/6, etc

Example 3. Newsstand buys and sells *The Wall Street Journal*. Observe how many copies he sells in a day.
S = {0, 1, 2, 3, 4}.

Probability of selling zero = $Pr\{0\}$ = 0.21, $Pr\{1\}$ = 0.26, $Pr\{2\}$ = 0.32, $Pr\{3\}$ = 0.16, $Pr\{4\}$ = 0.05,

2. Probabilities of events.

Event = a set of outcomes = a subset of the sample space S.

$$Pr\{E\} = \lim_{n\to\infty} \frac{\#\text{ times outcome is in } E \text{ in } n \text{ independent repetitions}}{n}$$

(intuitivedefn)

$$Pr\{ \{b_1, b_2, ..., b_m\} \} = \sum_{i=1}^{m} Pr\{b_i\}$$

(precisedefn)

$Pr\{E \cup F\}$ = $Pr\{E\} + Pr\{F\}$ if E and F are disjoint
(*additivity*)

Example 4 (related to Example 1). Look at two diodes. S = {gg, gd, dg, dd}. $Pr\{ gg \}$ = 0.9943, $Pr\{ gd \}$ = 0.0027, $Pr\{ dg \}$ = 0.0027, $Pr\{ dd \}$ = 0.0003. E = {gg, gd} = event where first diode is good, etc

Probability first diode is good = $Pr\{E\} = Pr\{ \{gg, gd\} \} = Pr\{gg\} + Pr\{gd\}$ = 0.9943 + 0.0027 = 0.97, etc

Example 5 (Example 3 continued). Newsstand stocks 2 copies of *The Wall Street Journal* on a certain day. Probability enough copies for all the customers that want to buy one = $Pr\{ \{0, 1, 2\} \} = Pr\{0\} + Pr\{1\} + Pr\{2\}$ = 0.21 + 0.26 + 0.32 = 0.79.

Example 6 (Example 2 continued). Roll a die. Let A = {2, 4, 6} = even and B = {1, 3} = 1 or a 3. A and B are disjoint. $A \cup B$ = {1, 2, 3, 4, 6} anything except a

5. We have $Pr\{A\} = 1/2$, $Pr\{B\} = 1/3$, and $Pr\{A \cup B\} = 5/6$. Note $Pr\{A \cup B\} = Pr\{A\} + Pr\{B\}$.

Problem 1. Office Max keeps a certain number of staplers on hand. If they sell out on a certain day, they order 6 more from the distributor and these are delivered in time for the start of the next day. Thus the inventory at the start of a day can be 1, 2, 3, 4, 5, or 6. Probability of 1 stapler at start of a day = $Pr\{1\}$ = 0.09, $Pr\{2\}$ = 0.21, $Pr\{3\}$ = 0.29, $Pr\{4\}$ = 0.23, $Pr\{5\}$ = 0.12 and $Pr\{6\}$ = 0.06. Probability that there is at least 3 staplers at the start of the day = $Pr\{\{3, 4, 5, 6\}\}$ = 0.7.

3. Random variables.

random variable = function defined on sample space = function X that assigns value $X(a)$ to each outcome a

$\{X = x\}$ = $\{a: X(a) = x\}$ = set of outcomes such that $X(a) = x$
(abbreviation)

$Pr\{X = x\}$ = $Pr\{a: X(a) = x\}$ = Probability that X assumes value x

$Pr\{X \in A\}$ = $Pr\{a: X(a) \in A\}$ = Probability that value of X lies in A

$Pr\{X = x, Y = y\}$ = $Pr\{a: X(a) = x$ and $Y(a) = y\}$ = Probability that X assumes value x and Y assumes value y

$f(x)$ = $Pr\{ X = x \}$ = *probability mass function* of random variable X

Example 7 (Example 4 continued). Look at two diodes. X_1 = result of first diode and X_2 = result of second diode.

$X_1(gg) = g$, $X_1(gd) = g$, $X_1(dg) = d$, $X_1(dd) = d$

$X_2(gg) = g$, $X_2(gd) = d$, $X_2(dg) = g$, $X_2(dd) = d$

$\{X_1 = g\}$ = $\{gg, gd\}$ $Pr\{X_1 = g\}$ = $Pr\{gg, gd\}$ = $Pr\{gg\} + Pr\{gd\}$ = 0.9943 + 0.0027 = 0.97

Example 8 (Two rolls of a die). $S = \begin{Bmatrix} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) \end{Bmatrix}$. $Pr\{ (i, j) \}$ = 1/36 for all i, j

X_1 = result of first roll $X_1(i,j) = i$

X_2 = result of second roll $X_2(i,j) = j$

$\{X_1 = 3\}$ = event where first roll is 3 = {(3, 1), (3, 2), (3, 2), (3, 4), (3, 5), (3, 6)}

$Pr\{X_1 = 3\}$ = Probability the first roll is a 3 = $Pr\{(3, 1), (3, 2), (3, 2), (3, 4), (3, 5), (3, 6)\}$

$= Pr\{ (3, 1) \} + Pr\{ (3, 2) \} + Pr\{ (3, 2) \} + Pr\{ (3, 4) \} + Pr\{ (3, 5) \} + Pr\{ (3, 6) \}$

$= \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{1}{6}$

$Pr\{X_1 = i\} = Pr\{X_2 = j\} = \frac{1}{6}$ for all i, j

$Pr\{X_1 = i, X_2 = j\} = \frac{1}{36}$ for all i, j

$Pr\{X_1 \in \{2, 3, 4\} \}$ = Probability the first roll is 2, 3, or 4 = $Pr\{X_1 = 2\} + Pr\{X_1 = 3\} + Pr\{X_1 = 4\}$

$= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = ½$

T = sum of the two rolls = $X_1 + X_2$

$\{T = 7\}$ = event where sum is 7 = $\{(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)\}$

$Pr\{ T = 7 \} = Pr\{(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)\} = \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{1}{6}$

$Pr\{ T = n \} = f(n) = (6 - |n - 7|)/36$

n	2	3	4	5	6	7	8	9	10	11	12
$f(n)$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

Problem 2 (related to Example7). Look at three diodes. $S = \{ggg, ggd, gdg, gdd, dgg, dgd, ddg, ddd\}$.

$Pr\{ggg\} = (0.999)^3$

$Pr\{ggd\} = Pr\{gdg\} = Pr\{dgg\} = (0.999)^2(0.001)$

$Pr\{gdd\} = Pr\{dgd\} = Pr\{ddg\} = (0.999)(0.001)^2$

$Pr\{ddd\} = (0.001)^3$

X_1 = condition of first diode,

X_2 = condition of second diode,

X_3 = condition of third diode,

N = number of diodes in the batch of three that are defective

$N = X_1 + X_2 + X_3$ if $d = 1$ and $g = 0$

Find the probability mass functions of X_1,X_2, X_3 and N.

Example 9. Observe the condition of the department's copier at the start of each day on two successive days: good condition = g, poor condition = p, broken = b. $S = \{gg, gp, gb, pg, pp, pb, bg, bp, bb\}$.

X_1 = state of the copier on the first day, either g, p or b,

X_2 = state of the copier on the second day,

1. Conditional probability.

Conditional probability. Conditional probabilities adjust the probability of something happening according to given information.

$$Pr\{A \mid B\} = \frac{Pr\{A \cap B\}}{Pr\{B\}} = \textit{conditional probability} \text{ of event } A \textit{ given } \text{event } B$$

Example 10 (in context of Example 4). Take two successive diodes from production line. Test first diode; it is defective. Does this affect the probability that the second diode is defective?
Intuitive approach. Take a large number of pairs of diodes and test both diodes in each pair. Only consider pairs where first diode is defective. Count the number in which the second diode is also defective. For a large number of pairs

Probability second diode is defective if first diode is defective

$$\approx \frac{\text{\# which both are defective}}{\text{\# which the first is defective}}$$

$$= \frac{\dfrac{\text{\# which both are defective}}{\text{total \# of pairs}}}{\dfrac{\text{\# which the first is defective}}{\text{total \# of pairs}}} \to \frac{Pr\{\text{both are defective}\}}{Pr\{\text{first is defective}\}} =$$

$$\frac{0.0003}{0.003} = 0.1,$$

as the number of pairs $\to \infty$. The observation that the first diode is defective does affect the probability that the second diode is defective.

Example 11 (in context of Example 2). Roll a die. Find conditional probability number is even given that if is 4 or larger. Let $A = \{2, 4, 6\}$ = number is even and

$B = \{4, 5, 6\}$ number is 4 or larger. $Pr\{A \mid B\} = \frac{Pr\{A \cap B\}}{Pr\{B\}} = \frac{1/3}{1/2} = 2/3.$

Problem 3. You own stock in Megabyte Computer Corporation. There is an 80% chance of Megabyte making a profit if it makes a technological breakthrough, but only a 30% chance of making a profit if they don't make the breakthrough. There is a 40% chance of its making the breakthrough. Before you can find out if they made the breakthrough, you go on a 6 month vacation to Tahiti. Then one day you receive the following message from your stockbroker. "Megabyte made a profit." What is the probability that Megabyte made the breakthrough?
Ans: 0.64

2.Independent Events.

Events A and B are *independent* if the probability that the outcome is in A is the same as the probability that the outcome is in A given that the outcome lies in B, i.e.

$$Pr\{A \mid B\} = Pr\{A\}$$

Since $Pr\{A \mid B\} = \dfrac{Pr\{A \cap B\}}{Pr\{B\}}$ this is equivalent to

$Pr\{A \cap B\} = Pr\{A\}\, Pr\{B\}$

(most books use this defn)

Example 12 (Example 10 continued). In Example 10 the second diode being defective is not independent of the first diode being defective.

Example 13 (Example 11 continued). In Example 11 getting an even number is not independent of getting 4 or larger since $Pr\{\text{even} \mid \text{4 or larger}\} = 2/3$, while $Pr\{\text{even}\} = 1/2$. On the other hand if $C = \{3, 4, 5, 6\}$ is the event of getting 3 or larger then $Pr\{A \mid C\} = 1/2$ so getting an even number is independent of getting 3 or larger.

Problem 4. a) You roll a die twice as in Example 5. Consider the event where the sum of the numbers on the two rolls is 7. Is this independent of rolling a 1 on the first roll? Ans: Yes.

b) Let B be the event of rolling a 1 on the first roll or second roll or both. Is event where the sum of the numbers on the two rolls is 7 independents of B Ans: No.

3.Independent Random Variables.

Random variables X and Y are *independent* if knowledge of the values of one of them doesn't influence the probability that the other assumes various values, i.e.

$$Pr\{X = x \text{ and } Y = y\} = Pr\{X = x\}\ Pr\{Y = y\} \qquad \text{for any } x \text{ and } y$$

Example 14 (in the context of Example 5). Roll two dice. X_1 = result of first roll and X_2 = result of second roll are independent since

$$Pr\{X_1 = i, X_2 = j\} = \frac{1}{36} = \frac{1}{6}\times\frac{1}{6} = Pr\{X_1 = i\}\ Pr\{X_2 = j\} \qquad \text{for all } i, j$$

Example 15 (in the context of Example 4). Look at two diodes. X_1 condition of first diode and X_2 = condition of second diode are not independent since

$$Pr\{X_1 = d, X_2 = d\} = 0.0003$$
$$Pr\{X_1 = d\}\ Pr\{X_2 = d\} = (0.003) \times (0.003) = 0.000009$$

Example 16 (variation on Example 4). Look at two diodes. X_1 condition of first diode and X_2 = condition of second diode. Suppose the probability mass functions of X_1 and X_2 are given by

$f(g) = Pr\{ X_1 = g \} = 0.997$ $\qquad$ $f(d) = Pr\{ X_1 = d \} = 0.003$

$h(g) = Pr\{ X_2 = g \} = 0.997$ $\qquad$ $h(d) = Pr\{ X_2 = d \} = 0.003$

Furthermore, suppose X_1 and X_2 are independent. Find the probabilities of the four outcomes gg, gd, dg and dd.

Using the independence one has

$Pr\{\ gg\ \} = Pr\{\ X_1 = g,\ X_2 = g\ \} = Pr\{\ X_1 = g\ \}\ Pr\{\ X_2 = g\ \} = f(g)h(g) =$ (0.997)(0.997) = 0.994009

$Pr\{\ gd\ \} = Pr\{\ X_1 = g,\ X_2 = d\ \} = Pr\{\ X_1 = g\ \}\ Pr\{\ X_2 = d\ \} = f(g)h(d) =$ (0.997)(0.003) = 0.002991

$Pr\{\ dg\ \} = Pr\{\ X_1 = d,\ X_2 = g\ \} = Pr\{\ X_1 = d\ \}\ Pr\{\ X_2 = g\ \} = f(d)h(g) =$ (0.003)(0.997) = 0.002991

$Pr\{\ dd\ \} = Pr\{\ X_1 = d,\ X_2 = d\ \} = Pr\{\ X_1 = d\ \}\ Pr\{\ X_2 = d\ \} = f(d)h(d) =$ (0.003)(0.003) = 0.000009

This illustrates a common situation where we describe a probability model by giving one or more random variables along with their probability mass functions together with some information, such as independence, about the joint behavior of the random variables. We do this instead of listing the elements in the sample space along with their probabilities.

Random variables, $X_1, \ldots, X_n$, then they are *independent* if knowledge of the values of some of the variables doesn't change the probability that the others assume various values, i.e.

$Pr\{\ X_1 = x_1, \ldots, X_n = x_n\ \} = Pr\{X_1 = x_1\} \cdots Pr\{X_n = x_n\}$ for any $x_1, \ldots, x_n$.

Example 17 (continuation of Example 16). Look at two diodes. X_i = condition of i^{th} diode Let X_i be the condition of the i^{th} diode for $i = 1, 2, \ldots, n$. Suppose all the X_i have the same probability mass function given by

$f_i(g) = Pr\{\ X_i = g\ \} = 0.997$ $\qquad$ $f_i(d) = Pr\{\ X_i = d\ \} = 0.003$

Furthermore, suppose all the X_i are independent. Find the probability that all n diodes are good.

Using the independence one has

$Pr\{\ gg\ldots g\ \} = Pr\{\ X_1 = g, X_2 = g, \ldots, X_n = g\ \} = = Pr\{\ X_1 = g\ \}\ Pr\{\ X_2 = g\ \} \cdots Pr\{\ X_n = g\ \}$

$= (0.997)\ (0.997)\ \cdots\ (0.997) = (0.997)^n$

7. Averages of Data

$x_1, x_2, \ldots, x_n$ = sequence of observations of something

$\overline{x}$ = average of $x_1, x_2, \ldots, x_n$ = $\dfrac{x_1 + x_2 + \cdots + x_n}{n}$

Example 18. You are a wholesaler for gasoline and each week you buy and sell gasoline.

q_1 = \$2.70, q_2 = \$2.60, q_3 = \$2.80, q_4 = \$2.70, q_5 = \$2.80: wholesale price of gasoline for last five weeks

$$\overline{q} = \frac{q_1 + q_2 + q_3 + q_4 + q_5}{5} = \frac{2.70 + 2.60 + 2.80 + 2.70 + 2.80}{5} = \frac{13.60}{5} = \$2.72$$

8. Means of Random Variables

X = random variable

$x_1, \ldots, x_m$ = the values X can assume

$f(x) = Pr\{X = x\}$ = probability mass function of X
$\mu_X = E(X)$ = *mean of X* = *expected value of X*

$$= Pr\{X = x_1\}\, x_1 + Pr\{X = x_2\}\, x_2 + \cdots + Pr\{X = x_m\}\, x_m = \sum_{k=1}^{m} Pr\{X = x_k\}\, x_k$$

$$= f(x_1)x_1 + f(x_2)x_2 + \cdots + f(x_m)x_m = \sum_{k=1}^{m} f(x_k)x_k$$

Rationale. We are modeling a situation where we are going to make a sequence of related observations by a sequence $X_1, X_1, \ldots, X_n$ of random variables where X_j is the result of the j^{th} observation. Suppose each of the random variables X_j takes on the values $x_1, \ldots, x_m$ and all the random variables have the same probability mass function $f(x)$ where $f(x_k) = Pr\{X_j = x_k\}$ for each j and k. Suppose $q_1, q_1, \ldots, q_n$ are the values we actually observe for the random variables $X_1, X_1, \ldots, X_n$. In our computation of $\overline{q}$ let's group all the values of q_j that equal x_1 together and all the values of q_j that equal x_2 together, etc. Then we have

$$\overline{q} = \frac{q_1 + q_2 + \cdots + q_n}{n} =$$

$$\frac{(x_1 + x_1 + \cdots + x_1) + (x_2 + x_2 + \cdots + x_2) + \cdots + (x_m + x_m + \cdots + x_m)}{n}$$

$$= \frac{g_1x_1 + g_2x_2 + \cdots + g_mx_m}{n} = \frac{g_1}{n}x_1 + \frac{g_2}{n}x_2 + \cdots + \frac{g_m}{n}x_m$$

where g_j is the number of times that x_j appears in $q_1, q_1, \ldots, q_n$. As $n \to \infty$ we expect $\frac{g_k}{n} \to Pr\{X = x_k\} = f(x_k)$ where X denotes any of the X_j. So as $n \to \infty$ we expect

$$\overline{q} \to f(x_1)x_1 + f(x_2)x_2 + \cdots + f(x_m)x_m = \mu_X$$

The fact that this holds if the X_j are independent is actually an important theorem in probability theory called the Law of Large Numbers.

Example 19. Suppose in Example 18 the set of possible values for the wholesale gasoline prices for any particular week is $S = \{2.60, 2.70, 2.80, 2.90, 3.00\}$. Let X_j be the wholesale price of gasoline on week j where week one is the first full week of July of this year. The X_j can be regarded as random variables. Assume each of the X_j has the same probability distribution and the probabilities that the gasoline price X_j takes on the values in S for the j^{th} week is as follows

$Pr\{X_j = 2.60\} = 0.25 Pr\{X_j = 2.70\} = 0.4 Pr\{X_j = 2.80\} = 0.2 Pr\{X_j = 2.90\} = 0.1 Pr\{X_j = 3.00\} = 0.05$

$\mu_X = (0.25)(2.60) + (0.4)(2.70) + (0.2)(2.80) + (0.1)(2.90) + (0.05)(3.00)$
$= 0.52 + 1.08 + 0.56 + 0.29 + 0.15 = 2.73$

If the X_j are all independent, then we would expect the average $\overline{q}_n$ of the actual prices over n weeks to approach \$2.73 as $n \to \infty$.

Problem 5. Newsstand buys and sells *The Wall Street Journal*. X = number he sells in a day. $Pr\{X = 0\} = 0.21$, $Pr\{X = 1\} = 0.26$, $Pr\{X = 2\} = 0.32$, $Pr\{X = 3\} = 0.16$, $Pr\{X = 4\} = 0.05$, Find μ_X. Answer: 1.58

Problem 6. Look at a diode. $S = \{d, g\}$. $Pr\{d\} = 0.003$, $Pr\{g\} = 0.997$. Random variable X is defined by $X(d) = 1$ and $X(g) = 0$. Find μ_X. Ans: $\mu_X = 0.003 = Pr\{d\}$

9. Properties of Means

$E(X + Y) = E(X) + E(Y)$
$E(cX) = cE(X)$
$E(XY) = E(X)E(Y)$ if X and Y are independent
$E(g(X)) = \sum_{k=1}^{m} g(x_k) f(x_k)$

Example 20. A company produces transistors. They estimate that the probability of any one of the transistors is defective is 0.1. Suppose a box contains 20 transistors. What is the expected number of defective transistors in a box?

Solution. Let $X_j = 1$ if the j^{th} transistor is defective and $X_j = 0$ if isis good. The number N of defective transistors is $N = X_1 + \cdots + X_{20}$. $E(N) = E(X_1) + \cdots + E(X_{20}) = (0.1) + \cdots + (0.1) = 2$.

This example illustrates the following general proposition.

Example 21. Consider a random walk where the probability of a step to the right is ½ and the probability of a step to the left is ½. After 4 steps your position Z could be either -2, 0 or 2 with probabilities ¼, ½ and ¼ respectively. Compute $E(Z^2)$.

Solution. $E(Z^2) = (-2)^2 Pr\{Z = -2\} + (0)^2 Pr\{Z = 0\} + (2)^2 Pr\{Z = 2\} = (4)(1/4) + (0)(1/2) + (4)(1/4) = 2$. If we were to compute $E(Z^2)$ from the definition (2) then $E(Z^2) = (4)\, Pr\{Z^2 = 4\} + (0)^2 Pr\{Z^2 = 0\} = (4)(1/2) + (0)^2 (1/2) = 2$.

The Law of Large Numbers. Let X_1, X_1, …, X_n, … be a sequence of independent random variables all taking on the same set of values and having the same probability mass function $f(x)$. Let $\overline{X}_n = \frac{X_1 + X_2 + \cdots + X_n}{n}$. Then $Pr\{a: \overline{X}_n(a) \rightarrow \mu_X\} = 1$ as $n \rightarrow \infty$.

QUESTIONS

1. What is a Probabilistic model?
2. What is the difference between deterministic and Probabilistic machine learning modes??
3. How is probabilities used in machine learning
4. What is a Probabilistic model in information retrieval?

REFERENCES

- Mitchell Machine Learning, Tata McGraw-Hill Education; First edition
- Python Willi Richert, Shroff/Packt Publishing Building Machine Learning Systems With; First edition (2013) .

*****

12

MACHINE LEARNING IN HYPER -AUTOMATION

Unit Structure

12.0 OBJECTIVE

Hyper parameters can be classified as model hyper parameters, that cannot be inferred while fitting the machine to the training set because they refer to the model selection task, or algorithm hyper parameters, that in principle have no influence on the performance of the model but affect the speed and quality of the learning process.

12.1 INTRODUCTION

Hyper-automation, which Gartner has identified as an IT mega-trend, is the likelihood that virtually everything in an organization can be automated. Legacy business procedures, for example, should actually be automated. The COVID-19 crisis has significantly increased the adoption of this phenomenon, which is otherwise known as intelligent process automation and digital process automation.

Machine learning and AI are crucial aspects and indispensable propellers of hyper-automation (in addition to various innovations such as process automation tools). For the sake of effectiveness, hyper-automation processes cannot rely on statically packaged software. Automated enterprise processes must have had the ability to adapt to evolving conditions and respond to unexpected situations.

12.1.1 Business Analysis and Predictions:

This strategy allows experts to collect and analyse a set of data over a time frame that is thereafter screened and used to make smart decisions. Machine learning networks can provide suppositions with accuracy as much as about 95% whenever trained using multiple data sets. In 2021 and beyond, we can expect that companies should integrate recurrent neural

networks for high-fidelity prediction. For instance, machine learning solutions can be integrated to unravel hidden trends and precise predictions. A clear illustration of this can be seen in insurance companies identifying likely frauds that could in one way or another have a great impact on them.

12.1.2 Automated Machine Learning:

The next phase of growth in machine learning trends is automated machine learning. It is a good idea for individuals who are not professionals in the complex world of machine learning as well as for professional data analysts and scientists. Automated machine learning enables these data experts to build machine learning models with increased productivity and efficiency while featuring extremely high quality. Thus, a tool such as AutoMachine Learning can be utilized to train top-quality custom ML models for clustering, regression, and classification without extensive knowledge of how to program. It can seamlessly produce an adequate level of customization without an in-depth understanding of the complicated workflow of machine learning. It can also be useful in utilizing machine learning best practices while saving resources and time. A good example of such AutoML is automated machine learning created by Microsoft Azure which can design and launch predictive models.

12.1.3 Synchronization of Machine Learning and Iot:

The Internet of Things is already a developed technology that involves the connection of several “things” or devices across a network, each having the ability to interact with each other. These devices are continually increasing, at such a rate that there is the possibility of having over 64 billion IoT devices by the end of 2025.The function of all of these devices is to collect data that can be evaluated and processed to generate useful insights. This is where ML becomes very pertinent. ML algorithms can be utilized to transform the data gathered by IoT devices into useful actionable outcomes.

A good example of this is Green Horizons, which is a project launched by IBM’s Research Lab in China whose goal is to regulate the pollution rates to better breathable levels. This is achievable with the use of an IoT network wherein sensors gather emissions from automobiles, traffic levels, weather, airflow direction, and pollen levels, among other things. Then machine learning algorithms are employed to figure out the most effective way to minimize these emissions. The synchronization of machine learning and IoT can as well be observed in the area of smart cars where autonomous-driving vehicles must be very precise and each part must communicate with one another in split seconds on the road. This demonstrates how essential the integration of these technologies is.In fact, Gartner forecasts that over 80% of business IoT projects will utilize AI and ML in one way or the other by 2022. This paced growth is a lot greater than the 10% of enterprise projects currently utilizing it.

12.1.4 Faster Computing Power:

AI analysts are basically close to the beginning of understanding the field of artificial neural networks and the most suitable approach to arranging them. This suggests that within the next year, algorithmic successes will continue increasing at an overwhelming pace with pragmatic progress and better problem-solving mechanisms. Similarly, cloud ML solutions are adding momentum as third-party cloud service platforms support the deployment of machine learning algorithms in the cloud. AI can resolve a reasonable range of unfavourable issues that require discovering insights and making decisions. Although, in the absence of the ability to lay hands on a machine's proposition, people will assume that it is cumbersome to accept that suggestion. With defined lines, conceive continued development in the transitional period increasing the explain ability and transparency regarding Artificial Intelligence algorithms.

12.1.5 Reinforcement Learning:

Reinforcement Learning (RL) is also one of the machine learning trends to look out for this year. It can generally be used by businesses in the near future. It is an innovative use of deep learning which makes use of its personal experiences to enhance the effectiveness of accumulated data RL, Artificial Intelligence programming is deployed with several conditions that determine what kind of activity will be executed by the software. In respect to various actions and outcomes, the software auto-learns actions to work to achieve the proper ultimate goal. A typical illustration of RL is a chatbot that attends to basic user queries such as consultation calls, order booking, greetings. ML Development Organizations can make use of Reinforcement Learning to ensure the ingenuity of the chatbot by including sequential conditions to it – for instance, differentiating prospective buyers and transferring calls to the appropriate service agent. Some other applications of reinforcement Learning are aircraft control, industrial automation, robot motion control, as well as business processes and strategic planning.

12.1.6 Machine Learning in Cybersecurity:

Machine learning continues to skyrocket in this contemporary time, it is likewise spreading its applications in several various sectors. One of the most common among these industries is the cybersecurity industry. Machine learning has a lot of applications in cybersecurity, some of which are detection of cyber threats, combating cyber-crime that equally utilizes ML capabilities, enhancing available antivirus software, among other things.

Machine learning is also employed in the creation of smart antivirus software that can detect any malware or virus by its irregular behaviour instead of just utilizing its signature like regular antivirus. Hence, the smart antivirus has the capacity to detect older threats from formerly experienced viruses and also fresh threats from viruses that were newly created by screening their abnormal behaviour. Several companies are integrating machine learning into cybersecurity.

12.2 MODELS' SYMBOLS BAGGING AND BOOSTING

This blog will explain 'Bagging and Boosting' most simply and shortly. But let us first understand some important terms which are going to be used later in the main content. Let's start with an example, If we want to predict 'sales' of a particular company based on its certain features, then many algorithms like Linear Regression and Decision Tree Regressor can be used. But both of these algorithms will make different predictions. Why is it so? One of the key factors is how much bias and variance they produce.Cool, but what if we don't know anything about Bias and Variance. So let's jump to Bias and Variance first.

12.2.1 Bias and Variance:

Bias: When we train our model on the given data, it makes certain assumptions, some correct and some incorrect. But when the model is introduced with testing or validation data, these assumptions (obviously incorrect ones) may create a difference in predicted value. So, to conclude from this," Bias is the difference between the Predicted Value and the Expected Value". When there is a **high bias** error, it results in a model that is not capable of taking so much variation. Since it does not learn enough from the training data, it is called **Underfitting.**

Variance is the error that occurs when the model captures fluctuations or noises of the data.

To explain further, the model learns too much from the training data, so that when it is introduced with new testing data, it is unable to predict the result accurately. When there is a high **variance** error, your model is so specific to the trained data, it is called **Overfitting.**

12.2.2 Ensemble Methods:

Let's take an example, If you want to purchase a new laptop, then you would browse different shopping websites/apps and check the price, features, and reviews/ratings and then compare them. You would also visit some local stores and also ask your friends or relatives for their opinion. Then at the end, you take a decision considering all these factors. Ensemble models in machine learning work on a similar idea. Ensemble methods create multiple models (called base learners/weak learners.) and combine/aggregate them into one final predictive model to decrease the errors (variance or bias). This approach allows us to produce better and more accurate predictive performance compared to a single model.

Ensemble methods can be divided into two groups:

12.2.2.1Parallel ensemble methods:

In this method base learners are generated parallelly, hence encouraging independence between the base learners. Due to the application of averages, the overall error is reduced.

12.2.2.2 Sequential ensemble methods:
In this method base learners are generated by sequence try; hence base learners are dependent on each other. Overall performance of the model is then increased by allocating higher weights to previously mislabelled/mis-predicted learners.

Boosting and bagging are the two most popularly used ensemble methods in machine learning. Now as we have already discussed prerequisites, let's jump to this blog's main content.

12.3 BAGGING

Bagging stands for **B**ootstrap **Agg**regating or simply Bootstrapping + Aggregating.

12.3.1Bootstrapping in Bagging refers to a technique where multiple subsets are derived from the whole (set) using the replacement procedure.

12.3.2 Aggregation in Bagging refers to a technique that combines all possible outcomes of the prediction and randomizes the outcome.

Hence many weak models are combined to form a better model.

Bagging is a **Parallel** ensemble method, where every model is constructed independently. Bagging is used when the aim is to **reduce variance.** So, now let's see how bagging is performed.

12.4 HOW IS BAGGING PERFORMED

The whole process of Bagging is explained in just a few steps. Please refer to the diagram below for a clearer understanding and visualization.

1. 'n' number of data subsets (d1, d2, d3…. dn) are generated randomly with replacement from the original dataset 'D'; **Bootstrapping**.
2. Now these multiple sub-datasets are used to train multiple models (which are called 'Weak Learners') like $m_1, m_2, m_3 \ldots m_n$.
3. Final prediction (Ensemble model) is given based on the aggregation of predictions from all weak models; **Aggregating**.

In the case of Regressor: the average/mean of these predictions is considered as the final prediction.

In the case of Classifiers: the majority vote gained from the voting mechanism is considered as the final prediction.

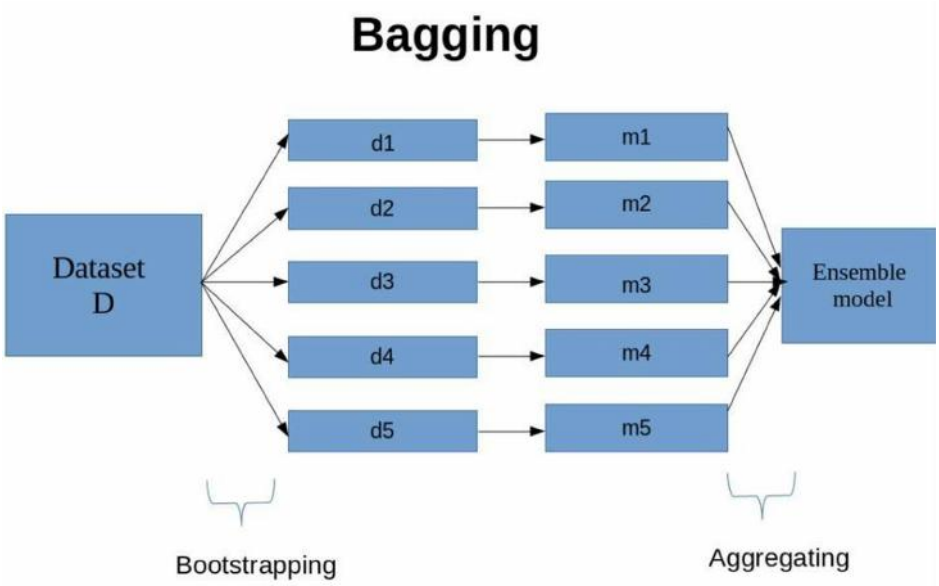


Figure 12.4.1 Bagging Representation

As "random sampling with replacement" is used here therefore every element has the same probability to appear in a new training sub-dataset and also some observations may be repeated. Ensemble model produced with these weak models is much more robust and with low variance.

12.4.1 Implementation of Bagging:

Random forest algorithm uses the concept of Bagging. Here is a sample code for how Bagging can be implemented practically. Remember it's just a sample code, just to introduce you to the required library.

12.5 BOOSTING

Boosting is a **Sequential** ensemble method, where each consecutive model attempts to correct the errors of the previous model.

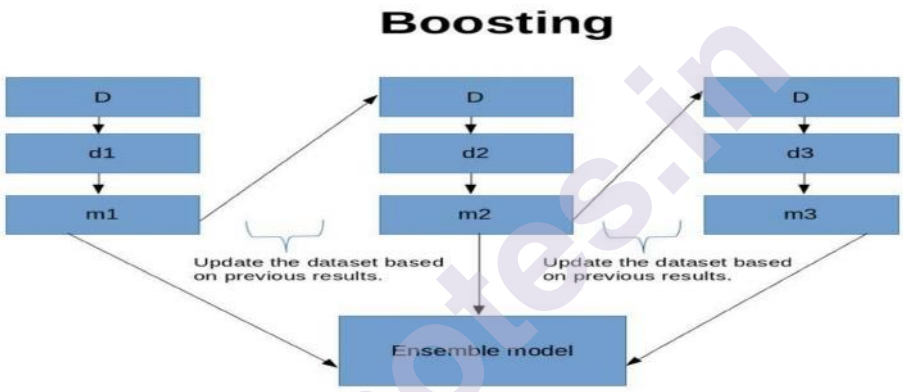
If a base classifier is misclassified in one weak model, its weight will get increased and the next base learner will classify it more correctly. Since the output of one base learner will be input to another, hence every model is dependent on its previous model. Boosting is used when the aim is to reduce bias**.** So now let's see how bagging is performed.

12.5.1 How is Boosting performed:

The whole process of Boosting is explained in just a few steps. Please refer to the diagram below for a clearer understanding and visualization.

1. let 'd1' be the data-subset, which is generated randomly with replacement from the original dataset 'D'.
2. Now this subset is used to train the model 'm1'(which is called a weak learner).

3. This model is then used to make predictions on the original(complete) dataset. Elements or instances which are misclassified/mis-predicted by this model, will be given more weights while choosing the next data-subset.
4. Let ‘d2’ be the data-subset, which is generated randomly with replacement from the dataset ‘D’(which is now updated with weights). In this step, instances which have more weights (concluded from the previous step) will be more likely to be chosen.
4. Now this subset is again used to train the model ‘m2’(which is called a weak learner).
5. Above steps are repeated for ‘n’ number of times, to get ‘n’ such models(m1,m2,m3…..mn)
6. Results of these ‘n’ weak models are combined to make a final prediction.



12.5.1 Figure Boosting

Implementation:

- AdaBoost
- Gradient boosting

These algorithms use Boosting in a different-different manner which we will see in detail in the next blog.

Here is a sample code for how boosting can be implemented practically on the AdaBoost algorithm. Remember it’s just a sample code, just to introduce you to the required library.

12.5.2 Similarities Between Bagging and Boosting:

1. Both of them are ensemble methods to get N learners from one learner.
2. Both of them generate several sub-datasets for training by random sampling.
3. Both of them make the final decision by averaging the N learners (or by Majority Voting).

4. Both of them are good at providing higher stability.

12.5.3 Bagging Vs Boosting:

1. The Main Goal of Bagging is to decrease variance, not bias. The Main Goal of Boosting is to decrease bias, not variance.
2. In Bagging multiple training data-subsets are drawn randomly with replacement from the original dataset. In Boosting new sub-datasets are drawn randomly with replacement from the weighted(updated) dataset
3. In Bagging, every model is constructed independently. In Boosting, new models are dependent on the performance of the previous model.
4. In Bagging, every model receives an equal weight. In Boosting, models are weighted by their performance.
5. In Bagging, the final prediction is just the normal average. In Boosting, the final prediction is a weighted average.
6. Bagging is usually applied where the classifier is unstable and has a high variance. Boosting is usually applied where the classifier is stable and has a high bias.
7. Bagging is used for connecting predictions of the same type. Boosting is used for connecting predictions that are of different types.
8. Bagging is an ensemble method of type Parallel. Boosting is an ensemble method of type Sequential

12.5.4 Multitask learning:

In most machine learning contexts, we are concerned with solving a *single* task at a time. Regardless of what that task is, the problem is typically framed as using data to solve a single task or optimize a single metric at a time. However, this approach will eventually hit a performance ceiling, oftentimes due to the size of the data-set or the ability of the model to learn meaningful representations from it. Multi-task learning, on the other hand, is a machine learning approach in which we try to learn *multiple* tasks simultaneously, optimizing multiple loss functions at once. Rather than training independent models for each task, we allow a single model to learn to complete all of the tasks at once. In this process, the model uses all of the available data across the different tasks to learn generalized representations of the data that are useful in multiple contexts. Multi-task learning has seen widespread usage across multiple domains such as natural language processing, computer vision, and recommendation systems. It is also commonly leveraged in industry, such as at Google, due to its ability to effectively leverage large amounts of data in order to solve related tasks.

12.5.4.1 When to use multi-task learning:

Before going into the specifics of how to implement a multi-task learning model, it is first important to go through situations in which multi-task learning is, and is not, appropriate. Generally, multi-task learning should

be used when the tasks have some level of **correlation**. In other words, multi-task learning improves performance when there are underlying principles or information shared between tasks. For example, two tasks involving classifying images of animals are likely to be correlated, as both tasks will involve learning to detect fur patterns and colours. This would be a good use-case formulti-task learning since learning these images features is useful for both tasks.

On the other hand, sometimes training on multiple tasks results in **negative transfer** between the tasks, in which the multi-task model performs worse than the equivalent single-task models. This generally happens when the different tasks are unrelated to each other, or when the information learned in one task contradicts that learned in another task.

12.5.4.2 Building a multi-task model:

Now that we know when we should use multi-task learning, we will go through a simple model architecture for a multi-task model. This will focus on a neural network architecture (deep multi-task learning), since neural networks are by far the most common type of model used in multi-task learning.

12.6 LEARNING A SHARED REPRESENTATION

At its core, deep multi-task learning aims to learn to produce generalized representations that are powerful enough to be shared across different tasks. I will focus on hard parameter sharing here, in which the different tasks use exactly the same base representation of the input data.

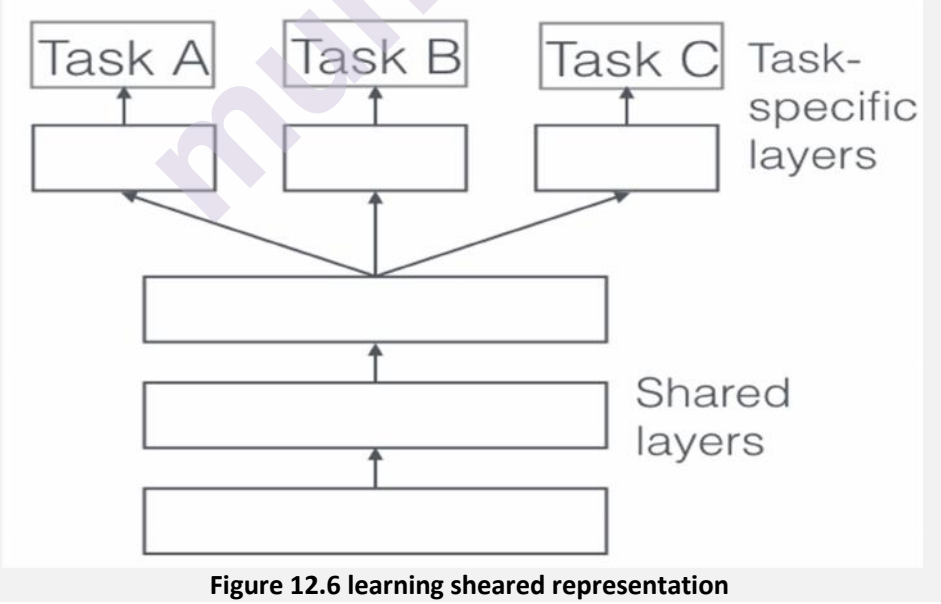


Figure 12.6 learning sheared representation

As we can see, hard parameter sharing forces the model to learn an intermediate representation that conveys enough information for all of the tasks. The task-specific portions of the network all start with the same base representation from the last shared layer.

Multi-task learning improves the generalizability of this representation because learning multiple tasks forces the model to focus on the features that are useful across all of the tasks. Assuming the tasks are correlated, a feature that is important for Task A is also likely to be important for Task

C. The opposite is also true; unimportant features are likely to be unimportant across all of the tasks.

Multi-task learning also effectively increases the size of your data-set, since you are combining the data-sets from each task. By adding more samples to the training set from different tasks, the model will learn to better ignore the task-specific noise or biases within each individual data-set.

12.6.1 Optimizing for multiple tasks:

Once the model's architecture has been decided, we need to decide what loss function to optimize. The simplest approach is to minimize a linear combination of the individual tasks' loss functions. Each task will have its own individual loss function **Li**. So in our multi-task model, we simply weight each loss function and minimize the sum of these weighted losses.

Now that we know how to build a multi-task model, we need to identify how to apply this method to a given task. In many cases, it may seem like you truly only have one task to solve, and it may not be obvious how to adapt your problem into a multi-task learning situation. In cases where you do not explicitly have multiple tasks, you can create **auxiliary tasks** that aim to solve a problem that is related, but not identical, to your single original task. By creating an auxiliary task, you can still apply multi-task learning to your single primary task, and hopefully improve performance on it.

Identifying an auxiliary task is generally domain-specific and there is no one-size-fits-all approach to coming up with one. However, there are some general principles that they will have in common. In general, the auxiliary task should be related to the primary task, and should nudge the network into learning important features for the primary task. For example, if the primary task is to classify sequences of data, we can create an auxiliary task that is to reconstruct the sequence with an autoencoder. This auxiliary task explicitly forces the network to learn a sequence encoder that produces a representation that is informative enough to be able to reconstruct the original sequence. This is likely to improve performance on the original task, classifying the sequence, as well, simply by producing a more informative intermediate representation of the sequence.

12.6.2 What is Online Machine Learning?:

While you may not know *batch* or *offline learning* by name, you surely know how it works. It's the standard approach to machine learning. Basically, you source a dataset and build a model on the whole dataset at once. This is why it's called batch learning. You may be wondering why it

goes by yet another name: offline learning. That's because offline learning is the polar opposite of another machine learning approach that you may not even be aware of. It's called *online learning* and you should know what it can do for you.

12.6.2.1 Objective:
My objective in this post is to introduce you to online learning, describe its use cases, and show you how to get started in Scikit-learn. To help motivate things, know that online learning is a powerful tool that opens up a whole new world. It's a tool you can add to your toolbox, giving you capabilities to tackle problems that may have once been beyond your reach.

12.6.2.2 Offline vs Online Learning:
So what differentiates offline and online learning? In the simplest sense, offline learning is an approach that ingests all the data at one time to build a model whereas online learning is an approach that ingests data one observation at a time. There's a perfect one-to-one analogy here for those familiar with Gradient Descent. Offline learning, also known as batch learning, is akin to batch gradient descent. Online learning, on the other hand, is the analog of stochasticgradient descent. In fact, as we'll see, implementing online learning in Scikit-learn will utilize stochastic gradient descent with a variety of loss functions to create online learning versions of algorithms like logistic regression and support vector machines. There's more to online learning, though.

Online learning is data efficient and adaptable. Online learning is data efficient because once data has been consumed it is no longer required. Technically, this means you don't have to store your data. Online learning is adaptable because it makes no assumption about the distribution of your data. As your data distribution morphs or drifts, due to say changing customer behaviour, the model can adapt on-the-fly to keep pace with trends in real-time. In order to do something similar with offline learning you'd have to create a sliding window of your data and retrain every time. And if you've been paying attention, you surely noticed that you can use this methodology to do streaming analytics.

12.6.2.3 Online Learning Use Cases:
Now that you know the difference between offline and online learning, you may be wondering when to consider the latter. Simply put, consider online learning when:

1. Your data doesn't fit into memory
2. You expect the distribution of your data to morph or drift over time
3. Your data is a function of time (e.g. stock prices)

12.7 SEQUENCES PREDICTION

12.7.1 Types of Sequence Prediction Problems:

Sequence prediction is a popular machine learning task, which consists of predicting the next symbol(s) based on the previously observed sequence of symbols. These symbols could be a number, an alphabet, a word, an event, or an object like a webpage or product. For example:

- A sequence of words or characters in a text
- A sequence of products bought by a customer
- A sequence of events observed on logs

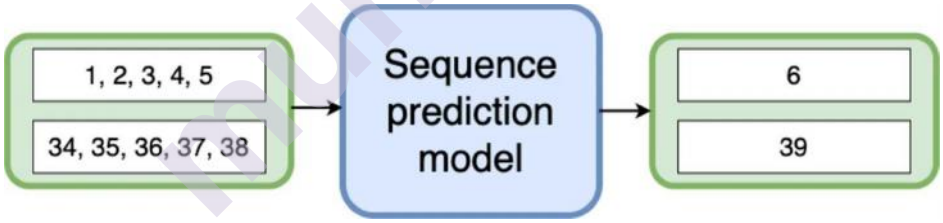
Sequence prediction is different from other types of supervised learning problems, as it imposes that the order in the data must be preserved when training models and making predictions.

Sequence prediction is a common problem which finds real-life applications in various industries. In this article, I will introduce to you three types of sequence prediction problems:

- Predicting the next value
- Predicting a class label
- Predicting a sequence

12.7.2 Predicting the next value:

Being able to guess the next element of a sequence is an important question in many applications. A sequence prediction model learns to identify the pattern in the sequential input data and predict the next value.



12.7.2. Figure: Sequence Prediction Model

12.7.3 Time-series forecasting:

Time-series refers to an ordered series of data, where the sequence of observations is sequentially in the time dimension. Time-series forecasting is about making predictions of what comes next in the series. Thus, Time-series forecasting involves training the model on historical data and using them to predict future observations.

But what makes time-series forecasting different from a regression problem? There are 2 things:

- Time series is time-dependent, which is ordered by time. But Regression can be applied to non-ordered data where a target variable is dependent on values taken by features.
- Time series looks for seasonality trends. For example, the power demand in a day will drop at night, and the number of air passengers will increase during the summer.

12.7.4 Webpage/product recommendation:
Have you searched for something, and every advertisement you saw next is related to what you searched for?

For example, after watching the movie *Avengers:* *Endgame*, I was searching for explanations of certain scenes. Ever since then, Google Discover Feed started to show me content revolve around the Marvel Cinematic Universe, even until today.

Even though it seems like Google Discover Feed is recommending a collection of webpages, each webpage is an individual output.

12.7.5 Predicting a class label:
Sequence classification uses labelled datasets with some sequence inputs and class labels as outputs, to train a classification model which can be used to predict the class label of an unseen sequence.

12.7.5 figure: Sequence Classification Model

12.7.5.1 Examples of sequence classification applications:
Text categorization. Assigning labels to documents written in a natural language has numerous real-world applications including sentiment classification and topic categorization.

Anomaly detection. Researchers explore detecting abnormal behaviours in 4 different time-series datasets, 1) electrocardiograms, 2) Space Shuttle Marotta valve, 3) power demand, and engine sensors datasets.

Genomic research. Researchers have been classifying protein sequences into categories. This work could be potentially useful for the discovery of a wide range of protein functions.

Health-informatics. Researchers use LSTM to classify electrocardiogram (ECG) signals into five different classes to identify the condition of a patient's heart. This allows end-to-end learning, extracting features related to ECG signals without any expert intervention.

Brain-computer interface. We extract brain signals from the Electroencephalogram, decoding of the user's intentions to operate the assistive devices.

12.8 WHAT IS ACTIVE LEARNING

Active learning is the subset of machine learning in which a learning algorithm can query a user interactively to label data with the desired outputs. In active learning, the algorithm proactively selects the subset of examples to be labelled next from the pool of unlabelled data. The fundamental belief behind the active learner algorithm concept is that an ML algorithm could potentially reach a higher level of accuracy while using a smaller number of training labels if it were allowed to choose the data it wants to learn from.

Therefore, active learners are allowed to interactively pose queries during the training stage. These queries are usually in the form of unlabelled data instances and the request is to a human annotator to label the instance. This makes active learning part of the human-in-the-loop paradigm, where it is one of the most powerful examples of success.

12.8.1 How does active learning work:

Active learning works in a few different situations. Basically, the decision of whether or not to query each specific label depends on whether the gain from querying the label is greater than the cost of obtaining that information. This decision making, in practice, can take a few different forms based on the data scientist's budget limit and other factors.

12.8.2 Stream-based selective sampling:

In this scenario, the algorithm determines if it would be beneficial enough to query for the label of a specific unlabelled entry in the dataset. While the model is being trained, it is presented with a data instance and immediately decides if it wants to query the label. This approach has a natural disadvantage that comes from the lack of guarantee that the data scientist will stay within budget.

12.8.3 Pool-based sampling:

This is the most well-known scenario for active learning. In this sampling method, the algorithm attempts to evaluate the entire dataset before it selects the best query or set of queries. The active learner algorithm is often initially trained on a fully labelled part of the data which is then used to determine which instances would be most beneficial to insert into the training set for the next active learning loop. The downside of this method is the amount of memory it can require.

Membership query synthesis:

This scenario is not applicable to all cases, because it involves the generation of synthetic data. The active learner in this method is allowed

to create its own examples for labelling. This method is compatible with problems where it is easy to generate a data instance.

SUMMARY

Bootstrap aggregating, also called bagging (from bootstrap aggregating), is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting.

UNIT END QUESTIONS

1. What is bagging and boosting
2. What is Active learning
3. What is sequences prediction
4. Explain in detail Boosting

REFERENCES

- Alpaydin Ethem,Introduction to Machine Learning by MIT; 2 edition (2010)
- Jacek M. Zurada , Jaico Publishing House Introduction to Artificial Neural Systems by ; First edition (25 January 1994).
- Simon Haykin, PHI Private Ltd Neural Networks and Learning Machines 2013.

*****