SPPU | Sem 7 | Computer Engineering

New Syllabus 2022-23

Strictly as per the New Syllabus (REV-2019 'C' Scheme) of Savitribai Phule Pune University w.e.f. academic year 2022-2023

Compulsory Subject (Course Code 410242)

MACHINE LEARNING

Prof. Baphana R. M.
Adjunct Faculty at C.O.E.P. (Pune)
Teaching Ph. D & M. Tech Students
(Artificial Intelligence & Robotics)

Dr. Chaitanya Kulkarni
H.O.D. Computer Engineering
Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati

Dr. Nilesh M. Patil
(SVKM's D. J. Sanghvi College of Engineering, Mumbai)

ISBN : 978-93-5583-194-1

TECH-NEO PUBLICATIONS
Where Authors Inspire Innovation
A Sachin Shah Venture

www.techneobooks.in
info@techneobooks.in

P7-72A

Price ₹ 345/-

Savitribai Phule Pune University

Machine Learning

(Course Code : 410242) (Compulsory Subject)

Semester VII - Computer Engineering

Strictly as per the New Syllabus (2019 Course) of Savitribai Phule Pune University w.e.f. Academic Year 2022-2023

Prof. R. M. Baphana
Adjunct Faculty,
Government College of Engineering, Pune (C.O.E.P)
(Teaching M. Tech & Ph.D students)

Dr. Chaitanya S. Kulkarni
Ph.D Computer Engineering, M.Tech. Computer
H.O.D. and Associate Professor
Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology, Baramati, Dist. Pune: 413 133

Dr. Nilesh M. Patil
Ph.D. (Computer Engineering)
Associate Professor, Computer Engg. Department,
SVKM's D J Sanghvi College of Engineering, Mumbai

TECH-NEO PUBLICATIONS
Where Authors Inspire Innovation
A Sachin Shah Venture

P7-72

▶ Unit II : Feature Engineering — 07 Hrs.

Concept of Feature, Preprocessing of data: Normalization and Scaling, Standardization, Managing missing values, Introduction to Dimensionality Reduction, Principal Component Analysis (PCA), Feature Extraction: Kernel PCA, Local Binary Pattern.

Introduction to various Feature Selection Techniques, Sequential Forward Selection, Sequential Backward Selection.

Statistical feature engineering: count-based, Length, Mean, Median, Mode etc. based feature vectorcreation.

Multidimensional Scaling, Matrix Factorization Techniques.

#Exemplar/CaseStudies : You are a Data Scientist, and a client comes to you with their data. Client is running a few campaigns from the past few months, but no campaign seemseffective. Client provides you the data of customers, product sales and past campaign success.

They want to increase their sales and figure out which marketing strategy isworking the best for them?

Questions for data scientists:

1. What data analysis approach will you follow ?
2. What statistical approach do you need to follow? How will you select important features? **(Refer chapter 2)**

End Sem Exam 70 Marks (Unit III, IV, V, VI)

▶ Unit III : Supervised Learning : Regression — 06 Hrs.

Bias, Variance, Generalization, Underfitting, Overfitting, Linear regression, Regression: Lasso regression, Ridge regression, Gradient descent algorithm.

Evaluation Metrics: MAE, RMSE, R2.

#Exemplar/Case Studies : Stock market price prediction. **(Refer chapter 3)**

▶ Unit IV : Supervised Learning : Classification — 08 Hrs.

Classification: K-nearest neighbour, Support vector machine. Ensemble Learning: Bagging, Boosting, Random Forest, Adaboost.

Binary-vs-Multiclass Classification, Balanced and Imbalanced Multiclass Classification Problems, Variants of Multiclass Classification: One-vs-One and One-vs-All

Evaluation Metrics and Score: Accuracy, Precision, Recall, Fscore, Cross-validation, Micro- Average Precision and Recall, Micro-Average F-score, Macro-Average Precision and Recall, Macro-Average F-score.

#Exemplar/Case Studies : Prediction of Thyroid disorders such as Hyperthyroid, Hypothyroid, Euthyroid-sick,and Euthyroid using multiclass classifier. **(Refer chapter 4)**

▶ Unit V : Unsupervised Learning — 07 Hrs.

K-Means, K-medoids, Hierarchical, and Density-based Clustering, Spectral Clustering. Outlier analysis: introduction of isolation factor, local outlier factor.

Evaluation metrics and score: elbow method, extrinsic and intrinsic methods Market basket analysis/Customer Segmentation.

#Exemplar/Case Studies : Market basket analysis/Customer Segmentation **(Refer chapter 5)**

▶ Unit VI : Introduction To Neural Networks — 06 Hrs.

Artificial Neural Networks: Single Layer Neural Network, Multilayer Perceptron, Back Propagation Learning, Functional Link Artificial Neural Network, and Radial Basis Function Network, Activation functions,

Introduction to Recurrent Neural Networks and Convolutional Neural Networks.

#Exemplar/Case Studies : Movie Recommendation System. **(Refer chapter 6)**

❑❑❑

INDEX

In Sem

End Sem

❑❑❑

UNIT III

CHAPTER 3

Supervised Learning : Regression

Syllabus

Bias, Variance, Generalization, Underfitting, Overfitting, Linear regression,
Regression: Lasso regression, Ridge regression, Gradient descent algorithm.
Evaluation Metrics : MAE, RMSE, R2

3.1 TRAINING ERROR AND GENERALIZATION ERROR

GQ. Explain and compare training error and generalization error.

3.1.1 Training Error

In **machine learning, training** a predictive model means finding a function which maps a set of values x to a value y. If we apply the model to the data it was trained on, we are calculating the **training error**. If we calculate the **error** on data which was unknown in the **training** phase, we are calculating the test **error**. Training error is calculated as follows:

$$E_{train} = \frac{1}{n} \sum_{i=1}^{n} \text{error } (f_D (X_i), Y_i)$$

In the above equation n represents the number of training examples. $f_D (X_i)$ represents the predicted value and Y_i represents the true or actual values, error $(f_D (X_i), Y_i)$ is used to represent that these two values are same or not and if not then these values differs by how much.

3.1.2 Generalization Error

For supervised learning applications in machine learning and statistical learning theory, **generalization error** is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data. Generalization error is calculated as follows:

$$E_{gen} = \int \text{error } (f_D (X_i), Y_i) P (Y, X) dX$$

In the above equation error is calculated over all possible values of X and Y. error $(fD (X_i), Y_i))$ is used to represent that these two values are same or not and if not then these values differes by how much. P(X, Y) represents how often we expect to see such X and Y.

3.1.3 Training Error versus Generalization Error

- The training error is the error of our model as calculated on the training dataset, while generalization error is the expectation of our model's error were we need to apply it to an infinite stream of additional data examples drawn from the same underlying data distribution as our original sample.
- Problematically, we can never calculate the generalization error exactly. That is because the stream of infinite data is an imaginary object. In practice, we must estimate the generalization error by applying our model to an independent test set constituted of a random selection of data examples that were withheld from our training set.
- Let's see an example. Consider a college student trying to prepare for his final exam. A diligent student will strive to practice well and test his abilities using exams from previous years. Nonetheless, doing well on past exams is no guarantee that he will excel when it matters. For instance, the student might try to prepare by rote learning the answers to the exam questions.
- This requires the student to memorize many things. She might even remember the answers for past exams perfectly. Another student might prepare by trying to understand the reasons for giving certain answers. In most cases, the latter student will do much better.

- Let's see one more example, consider the problem of trying to classify the outcomes of coin tosses (class 0: heads, class 1: tails) based on some contextual features that might be available. Suppose that the coin is fair.
- No matter what algorithm we come up with, the generalization error will always be 1/2. However, for most algorithms, we should expect our training error to be considerably lower, depending on the luck of the draw, even if we did not have any features! Consider the dataset {0, 1, 1, 1, 0, 1}. Our feature-less algorithm would have to fall back on always predicting the majority class, which appears from our limited sample to be 1.
- In this case, the model that always predicts class 1 will incur an error of 1/3, considerably better than our generalization error. As we increase the amount of data, the probability that the fraction of heads will deviate significantly from 1/2 diminishes, and our training error would come to match the generalization error.
- When we train our models, we attempt to search for a function that fits the training data as well as possible. If the function is so flexible that it can catch on to spurious patterns just as easily as to true associations, then it might perform *too well* without producing a model that generalizes well to unseen data.
- This is precisely what we want to avoid or at least control. Many of the techniques in deep learning are heuristics and tricks aimed at guarding against overfitting.
- When we have simple models and abundant data, we expect the generalization error to resemble the training error. When we work with more complex models and fewer examples, we expect the training error to go down but the generalization gap to grow.

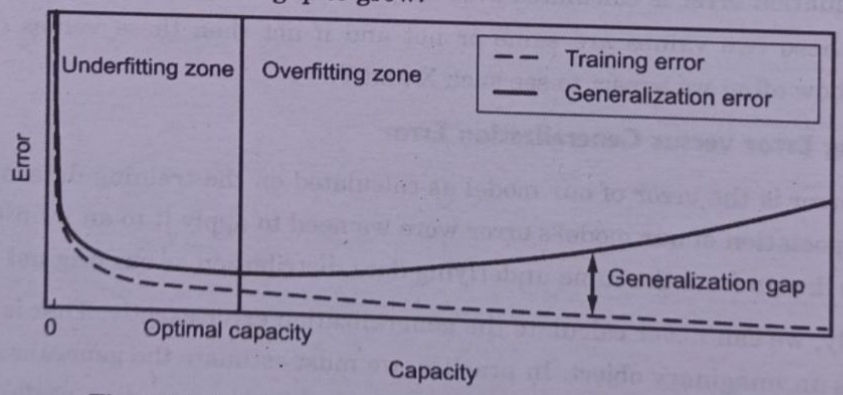


Fig. 3.1.1 : Training Error and Generalization Error

3.2 UNDERFITTING, OVERFITTING, BIAS AND VARIANCE TRADE OFF

GQ. Mention uses of Underfitting, Overfitting, Bias and Variance Trade Off.

- Let us consider that we are designing a machine learning model. A model is said to be a good machine learning model if it generalizes any new input data from the problem domain in a proper way. This helps us to make predictions in the future data, that data model has never seen.

- Now, suppose we want to check how well our machine learning model learns and generalizes to the new data. For that we have overfitting and underfitting, which are majorly responsible for the poor performances of the machine learning algorithms.

Before diving further let's understands two important terms:

- Bias – Assumptions made by a model to make a function easier to learn. (The algorithms error rate on the training set is algorithms bias.)
- Variance - If you train your data on training data and obtain a very low error, upon changing the data and then training the same previous model you experience high error, this is variance.
- (How much worse the algorithm does on the test set than the training set is known as the algorithms variance.)

3.2.1 Underfitting

- A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data.
- Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough.
- It usually happens when we have less data to build an accurate model and also when we try to build a linear model with a non-linear data.
- In such cases the rules of the machine learning model are too easy and flexible to be applied on such minimal data and therefore the model will probably make a lot of wrong predictions.
- Underfitting can be avoided by using more data and also reducing the features by feature selection.
- In a nutshell, Underfitting – High bias and low variance

3.2.2 Techniques to Reduce Underfitting

1. Increase model complexity
2. Increase number of features, performing feature engineering
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.

3.2.3 Overfitting

- A statistical model is said to be overfitted, when we train it with a lot of data . When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set.
- Then the model does not categorize the data correctly, because of too many details and noise.
- The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models.

- A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.
- In a nutshell, Overfitting – High variance and low bias

3.2.4 Techniques to Reduce Overfitting

GQ. What are Techniques to Reduce Overfitting.

1. Increase training data.
2. Reduce model complexity.
3. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
4. Ridge Regularization and Lasso Regularization
5. Use dropout for neural networks to tackle overfitting.

- Ideally, the case when the model makes the predictions with 0 error, is said to have a *good fit* on the data. This situation is achievable at a spot between overfitting and underfitting. In order to understand it we will have to look at the performance of our model with the passage of time, while it is learning from training dataset.
- With the passage of time, our model will keep on learning and thus the error for the model on the training and testing data will keep on decreasing. If it will learn for too long, the model will become more prone to overfitting due to the presence of noise and less useful details. Hence the performance of our model will decrease. In order to get a good fit, we will stop at a point just before where the error starts increasing. At this point the model is said to have good skills on training datasets as well as our unseen testing dataset.

Bias-variance trade-off

So what is the right measure? Depending on the model at hand, a performance that lies between overfitting and underfitting is more desirable. This trade-off is the most integral aspect of Machine Learning model training. As we discussed, Machine Learning models fulfil their purpose when they generalize well. Generalization is bound by the two undesirable outcomes - high bias and high variance. Detecting whether the model suffers from either one is the sole responsibility of the model developer.

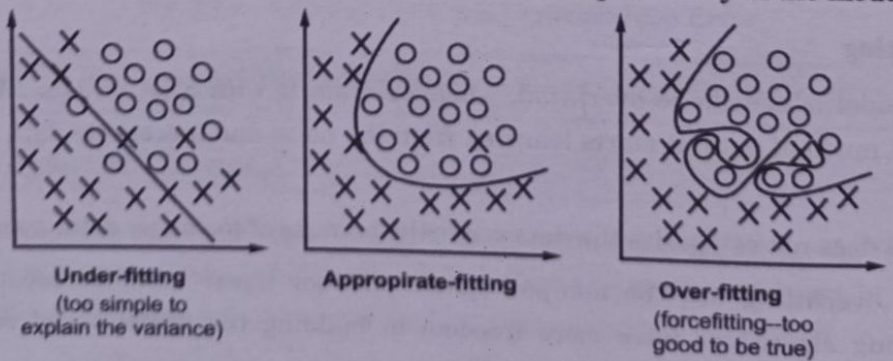


Fig. 3.2.1 : Underfitting and Overfitting

Bias-variance trade-off

So what is the right measure? Depending on the model at hand, a performance that lies between overfitting and underfitting is more desirable. This trade-off is the most integral aspect of Machine Learning model training. As we discussed, Machine Learning models fulfil their purpose when they generalize well. Generalization is bound by the two undesirable outcomes — high bias and high variance. Detecting whether the model suffers from either one is the sole responsibility of the model developer.

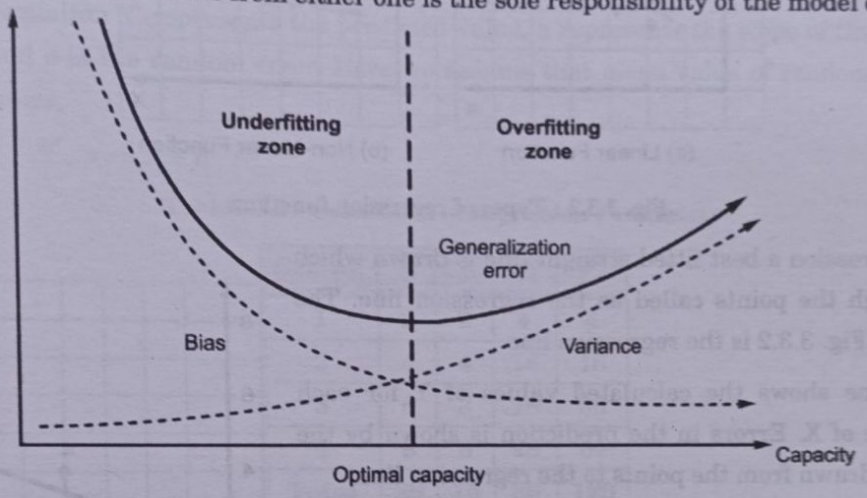


Fig. 3.2.2 : Bias variance Tradeoff as a function of model capacity

3.3 LINEAR REGRESSION

UQ. Explain Regression line, Scatter plot, Error in prediction and Best fitting line. **(Ref. – May 15, 4 Marks)**

UQ. Explain in brief Linear Regression Technique. **(Ref – May 16, 5 Marks)**

- One of the most important supervised learning tasks is regression. In regression set of records are present with X and Y values and this values are used to learn a function, so that if you want to predict Y from an unknown X this learn function can be used. In regression we have to find value of Y, So, a function is required which predicts Y given X. Y is continuous in case of regression.
- Here Y is called as criterion variable and X is called as predictor variable. There are many types of functions or modules which can be used for regression. Linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.

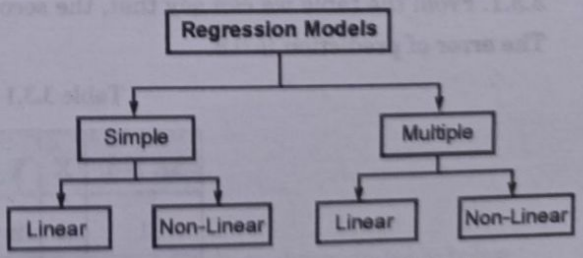
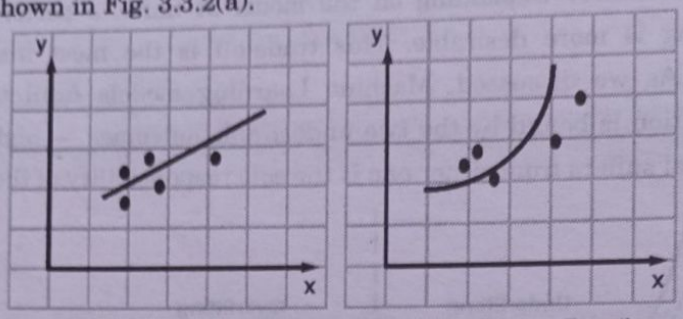


Fig. 3.3.1 : Types of Regression Models

3.3.1 Simple Linear Regression

- Let's see simple regression first, in this X contains a single feature. In multiple regressions, X contains more than one feature. In simple regression training records are plotted as value of X vs. value of Y. Next task is to find a function, so that if a random unknown X value is given we can predict Y.

- There are different types of functions that can be used. In linear regression, we assume that the function is linear as shown in Fig. 3.3.2(a).



(a) Linear Function (b) Non-Linear Function

Fig. 3.3.2 : Types of regression functions

- In Linear regression a best fitted straight line is drawn which passes through the points called as the regression line. The line shown in Fig. 3.3.2 is the regression line.
- Regression line shows the calculated values of Y for each possible value of X. Errors in the prediction is shown by the vertical lines drawn from the points to the regression line.
- Error in prediction is less when the point is very near to the regression line as shown by first and second point in Fig. 3.3.3. When the point is far from the regression line then the error in prediction is high, as shown by the third and fourth point in Fig. 3.3.3.

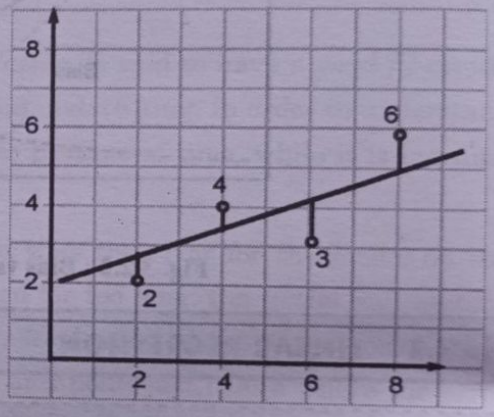


Fig. 3.3.3 : Prediction and the Error in the Prediction

- The difference between the value of point and the predicted value is called as error of prediction. Predicted value is the value of point on the line.
- Let's take an example, the predicted values (Y′) and the errors of prediction (Y-Y′) are shown in Table 3.3.1. From the table we can say that, the second point has Y value as 4 and a predicted Y value as 3.2. The error of prediction is 0.8.

Table 3.3.1 : Regression data

Sr. No.	X	Y	Y′	Y-Y′	(Y-Y′)²
1	2	2	2.2	– 0.2	0.04
2	4	4	3.2	0.8	0.64
3	6	3	4.3	– 1.3	1.69
4	8	6	5.4	0.6	0.36

- The best-fit line is called as the line that will minimize the sum of the squared errors of prediction. This criterion is used to draw the line in Fig. 3.3.3. The squared errors of prediction are shown in the last column of Table 3.3.1.
- The regression line is represented using the following equation,

$$Y' = aX + b + e$$

- In the above equation Y' represents the predicted value, a represents the slope of the line, b shows the Y intercept and e is the random error. Here we assume that mean value of random error is 0, so the equation becomes,

$$Y' = aX + b$$

Table 3.3.2 : Calculation of Regression Parameters

Sr. No.	X	Y	XY	X^2
1	2	2	4	4
2	4	4	16	16
3	6	3	18	36
4	8	6	48	64
Total	**20**	**15**	**86**	**120**

- The regression line equation is,

$$Y' = aX + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = \frac{4 \times 86 - 20 \times 15}{4 \times 120 - 400} = 0.55$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = \frac{1}{4}(15 - 0.55 \times 20) = 1$$

- Now the equation for the line becomes,

$$Y' = 0.55X + 2.2$$

For X = 2, $Y' = (0.55)(2) + 1 = 2.2$

For X = 4, $Y' = (0.55)(4) + 1 = 3.2$

For X = 6, $Y' = (0.55)(6) + 1 = 4.3$

For X = 8, $Y' = (0.55)(8) + 1 = 5.4$

3.3.2 Examples of Linear Regression

Ex. 3.3.1 : The expenditure of an organization (in thousand) for every month is shown in table below :

X (Month)	1	2	3	4	5
Y (Expenditure)	12	19	29	37	45

Find regression line, Y = a X + b using least square method.

Estimate the expenditure of company in 6th month using line as a model.

☑ **Soln. :**

Sr. No.	X	Y	XY	X^2
1	1	12	12	1
2	2	19	38	4
3	3	29	87	9
4	4	37	148	16
5	5	45	225	25
Total	**15**	**142**	**510**	**55**

(a) The equation for the regression line is,

$$Y' = aX + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = \frac{5 \times 510 - 15 \times 142}{5 \times 55 - 225} = 8.4$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = \frac{1}{5}(142 - 8.4 \times 15) = 3.2$$

Now the equation for the line becomes,

$$Y' = 8.4X + 3.2$$

(b) In 6^{th} month

$$Y' = 8.4 \times 6 + 3.2$$

$$Y' = 53.6 \text{ Thousand}$$

Ex. 3.3.2 : Consider the set of data as {(– 1, – 1), (2, 2), (3, 2)} (a) Find the equation of regression line. (b) Draw the scatter plot of data and regression line.

☑ **Soln. :**

Sr. No.	X	Y	XY	X^2
1	– 1	– 1	1	1
2	2	2	4	4
3	3	2	6	9
Total	4	3	11	14

(a) The equation for the regression line is,

$$Y' = aX + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = \frac{3 \times 11 - 4 \times 3}{3 \times 14 - 16} = 0.807$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = \frac{1}{3}(3 - 0.807 \times 4) = -0.076$$

Now the equation for the line becomes,

$$Y' = 0.807X - 0.076$$

(b) Now we can plot the regression line given by equation $Y' = 0.807\ X - 0.076$ and the given data points.

Sr. No.	X	Y'
1	– 1	– 0.883
2	2	1.54
3	3	2.34

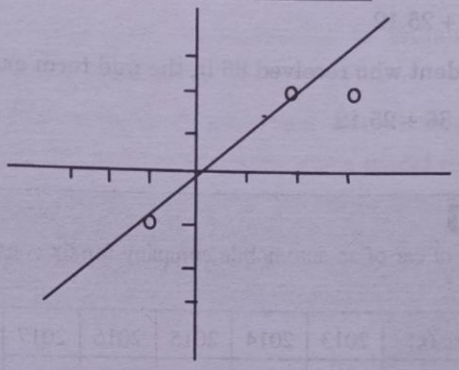


Fig. Ex. 3.3.2 : Scatter plot of data

UEx. 3.3.3 Ref. - May 17, 10 Marks

Following table shows the midterm and final exam grades obtained for students in a database course. Use the method of least squares using regression to predict the final exam grade of a student who received 86 in the mid term exam.

Midterm exam(X)	72	50	81	74	94	86	59	83	86	33	88	81
Final exam(Y)	84	53	77	78	90	75	49	79	77	52	74	90

☑ **Soln. :**

Sr. No.	X	Y	XY	X^2
1	72	84	6048	5184
2	50	53	2650	2500
3	81	77	6237	6561
4	74	78	5772	5476
5	94	90	8460	8836
6	86	75	6450	7396
7	59	49	2891	3481
8	83	79	6557	6889
9	86	77	6622	7396
10	33	52	1716	1089
11	88	74	6512	7744
12	81	90	7290	6561
Total	**887**	**878**	**67205**	**69113**

The equation for the regression line is,

$$Y' = aX + b$$

$$a = \frac{n\sum XY - \sum X \sum Y}{n\sum X^2 - (\sum X)^2} = 0.65$$

$$b = \frac{1}{n}(\sum Y - a \times \sum X) = 25.12$$

Now the equation for the line becomes,

$$Y' = 0.65X + 25.12$$

The final exam grade of a student who received 86 in the mid term exam,

$$Y' = 0.65 \times 86 + 25.12$$

$$Y' = 81.02$$

UEx. 3.3.4 **Ref. - Dec. 19, 10 Marks**

Given the following data for the sales of car of an automobile company for six consecutive years. Predict the sales for next two consecutive years.

Years (x)	2013	2014	2015	2016	2017	2018
Sales (y)	110	100	250	275	230	300

Soln :

We will take $t = x - 2013$

Sr. No.	t	Y	tY	t^2
0	0	110	0	0
1	1	100	100	1
2	2	250	500	4
3	3	275	825	9
4	4	230	920	16
5	5	300	1500	25
Total	**15**	**1265**	**3845**	**55**

The equation for the regression line is,

$$Y' = at + b; \qquad a = 39; \qquad b = 113.33$$

Now the equation for the line becomes,

$$Y' = 39t + 113.33$$

The sale of company for next two years,

$$X = 2019, t = 6; \qquad Y' = 39 \times 6 + 113.33 = 347.33$$

$$X = 2020, t = 7; \qquad Y' = 39 \times 7 + 113.33 = 386.33$$

3.4 LASSO REGRESSION

GQ. Explain lasso regression in detail.

- In machine learning, lasso (least absolute shrinkage and selection operator, also Lasso or LASSO) is a regression analysis method.
- It performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the resulting statistical model.
- Here we shall see the techniques used to overcome **overfitting** for a lasso regression model. Regularization is one of the methods, used to make the given model more generalized.

3.4.1 What is Lasso Regression ?

- Lasso regression is a regularization technique. It uses shrinkage.
- Shrinkage is where data values are shrunk towards a central point as the mean.
- The lasso procedure encourages simple, sparse model, i.e.; models with fewer parameters.
- This type of regression is suitable for models showing high levels of multicollinearity or when one wants to automate certain parts of model selection, like variable selection/parameter elimination.
- Lasso Regression uses L1 regularization technique, when there are more features. It is because it automatically performs feature selection.
- Lasso is a statistical formula for the regularization of data models and feature selection.

3.4.2 Regularization

- Regularization is an important concept. It is used to avoid over fitting of the data, when the trained and test data are much varying.
- Regularization is implemented by adding a **'penalty'** term to the best fit derived from the trained data. It achieves a **lesser variance** with the tested data and also restricts the influence of predictor variables over the output variable and this is achieved by compressing their coefficients.
- In regularization, we keep the same number of features but reduce the magnitude of the coefficients. The magnitude of the coefficients can be reduced by using different types of techniques. First we discuss the techniques.

3.4.3 Lasso Regularization Techniques

- If a regression model uses the L1 regularization technique, it is called Lasso Regression.
- L1 regularization adds a penalty that is equal to the absolute value of the magnitude of the coefficient.
- This regularization type develops sparse models with few coefficients.
- Larger penalties result in coefficient values that are closer to zero.

3.4.4 Mathematical equation of Lasso Regression

- The mathematical equal of L.R. is given by :

- Residual sum of squares + λ. [Sum of the absolute value of the magnitude of coefficients]

i.e., $$\sum_{i=1}^{n}\left[y_i - \sum_{j} x_{ij} B_j\right]^2 + \lambda \sum_{j=1}^{P} |\beta_j|$$

where

(i) λ denotes the amount of shrinkage,

(ii) λ = 0 implies all features are considered and it is equivalent to the linear regression where only the residual sum of squares is considered. That builds a predictive model.

(iii) λ = ∞ implies no feature is considered i .e., as λ is very large, it eliminates more and more features.

(iv) The bias increases with increase in λ.

(v) Variance increases with decrease in λ.

☞ Lasso regression example

```
import numpy as np
```

Creating a New Train and Validation Datasets

```
from sklearn.model_selection import train_test_split
data_train, data_val = train_test_split(new_data_train, test_size = 0.2, random_state = 2)
```

Classifying Predictors and Target

```
#Classifying Independent and Dependent Features
#_____
#Dependent Variable
Y_train = data_train.iloc[:, -1].values
#Independent Variables
X_train = data_train.iloc[:,0 : -1].values
#Independent Variables for Test Set
X_test = data_val.iloc[:,0 : -1].values
Evaluating The Model With RMLSE
def score(y_pred, y_true):
error = np.square(np.log10(y_pred +1) - np.log10(y_true +1)).mean() ** 0.5
score = 1 - error
return score
actual_cost = list(data_val['COST'])
actual_cost = np.asarray(actual_cost)
```

☞ Building the Lasso Regressor

```
#Lasso Regression
from sklearn.linear_model import Lasso
```

```
#Initializing the Lasso Regressor with Normalization Factor as True
lasso_reg = Lasso(normalize=True)
#Fitting the Training data to the Lasso regressor
lasso_reg.fit(X_train,Y_train)
#Predicting for X_test
y_pred_lass =lasso_reg.predict(X_test)
#Printing the Score with RMLSE
print("\n\nLasso SCORE : ", score(y_pred_lass, actual_cost))
```

Output

```
0.7335508027883148
The Lasso Regression attained an accuracy of 73% with the given Dataset.
```

3.4.5 Bayesian Interpretation

- Lasso can be interpreted as linear regression for which the coefficients have Laplace prior distributions.
- The Laplace distribution is sharply peaked at zero (its first derivative is discontinuous at zero) and it concentrates its probability mass closer to zero than the normal distribution.
- This gives an alternative explanation of why lasso tends to set some coefficients to zero.

3.4.6 Choice of Regularisation Parameter

- Choosing the regularization parameter (λ) is a fundamental part of lasso. A good value is essential to the performance of lasso, because it controls the strength of shrinkage and variable selection. It improves both prediction accuracy and interpretability.
- But, if the regularization is too strong, then important variables may be omitted and coefficients get shrunk excessively. This harms both predictive capacity and inferencing.
- Cross-validation is often used to find the regularization parameter.
- Information criterion such as the Bayesian information criterion (BIC) is preferable to cross-validation, because it is faster to compute and its performance is less volatile in small samples.

3.4.7 Selected Applications

LASSO has been applied in economics and finance, and it is found to improve prediction and to select sometimes neglected variables, for example, in corporate bankruptcy prediction literature, or high growth firms prediction.

3.5 RIDGE REGRESSION

GQ. What is Ridge regression models ?

- Ridge **regression** is a model tuning method. It is used to analyse any data that suffers from multicollinearity.

- This method performs L2 regularization. When we get multicollinearity problem, least-squares method are unbiased, and variances are large. This makes predicted values far away from the actual values.
- The cost function for ridge regression :

$$\text{Min}\left(||\, Y - X\,(\text{theta})\,||^2 + \lambda\, ||\, \text{theta}\,||^2 \right)$$

- Lambda is the penalty term, λ given here is denoted by an alpha-parameter in the ridge function. So, by changing the values of alpha, we can control penalty term.
- The higher the values of alpha, the bigger is the penalty and therefore the magnitude of the coefficient is reduced.
  (i) It shrinks the parameters. Hence, it is used to prevent multicollinearty.
  (ii) It reduces the model complexity by coefficient shrinkage.

3.5.1 Ridge Regression Models

- The usual regression equation forms the base for any type of regression machine learning model. It is given by :

$$Y = XB + e$$

- Where Y is the dependent variable, X represent the independent variables, B is the regression coefficient to be estimated, and e represents the errors due to residuals.
- We add the lambda function to this equation, the variance that is not evaluated by the general method, is considered.
- When the data is ready and is part of L2 regularization, we can under take the steps :

3.5.2 Standardisation

- In ridge regression, the first step is to standardize the variables (both dependent and independent) by subtracting their means and dividing by their standard deviations.
- As far as standardization is concerned, all ridge regression calculations are based on standardized variables.
- When the final regression coefficients are displayed, they are adjusted back into their original scale.

3.5.3 Bias and Variance trade-off

- Bias and variance trade-off is generally complicated when it comes to building ridge regression models on an actual dataset.
- The general trends are :
  1. The bias increases as λ increases.
  2. The variance decreases as λ increases.

3.5.4 Assumptions of Ridge Regressions

- The assumptions of ridge regression are the same as that of linear regression : linearity, constant variance and independence.

- Since ridge regression does not give confidence limits, the distribution of errors need not be normal.

Example

Upload Required Libraries

```
import numpy as np
import pandas as pd
import os

import seaborn as sns
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import matplotlib.style
plt.style.use('classic')

import warnings
warnings.filterwarnings("ignore")

df = pd.read_excel("food.xlsx")
```

	Week	Final_price	Unit_price	Website_homepage_mention	Food_category	Cuisine	Center_type	Home_delivery	Nigh_service	Area_range	order
0	122.0	150.35	152.35	0.0	Beverages	Italian	Delhi	1.0	1	7.0	972
1	95.0	484.03	485.03	0.0	Desert	Italian	Noida	0.0	1	5.6	150
2	52.0	281.33	281.33	0.0	Starters	Italian	Gurgaon	0.0	1	3.0	55
3	10.0	167.81	196.94	0.0	Extras	Italian	Delhi	1.0	1	4.0	256
4	122.0	212.46	256.14	0.0	Salad	Italian	Dehil	0.0	1	3.4	82

After conducting all the EDA on the data, treatment of missing values, we shall now go ahead with creating dummy variables, as we cannot have categorical variables in the dataset.

```
df =pd.get_dummies(df, columns=cat,drop_first=True)
```

Where columns=cat is all the categorical variables in the data set.

After this, we need to standardize the data set for the Linear Regression method.

Ridge expression can be used for the analysis of prostrate-specific antigen and clinical measures among people who were about to have prostrates removed.

The performance of ridge regression is good when there is a subset of true coefficients which are small or even zero.

3.6 MULTICOLLINEARITY

- The term multicollinearity refers to collinearity concepts in statistics.
- In this model, one predicted value in multiple regression models is linearly predicted with others to attain a certain level of accuracy.
- The concept multicollinearity occurs when there are high correlation between more than two predicted values.

The Regularisation techniques are as follows

(i) Penalise the magnitude of coefficients of features,

(ii) Minimise the error between the actual and predicted observations.

3.6.1 Working of Ridge-Regression model

- Ridge Regression performs L2 regularization.
- Here the penalty equivalent is added to the square of the magnitude of coefficients.
- The minimization objective is as follows :
- Consider a response vector $y \in R_n$ and a predictor matrix $X \in R_{n \times p}$, the ridge regression coefficients are defined as :

(i) Here λ is the turning factor that controls the strength of the penalty term.

(ii) If $\lambda = 0$, the objective becomes similar to simple linear regression. Hence we get the some coefficients as simple linear regression.

(iii) If $\lambda = \infty$, the coefficient will be zero, because of infinite weightage on the square of coefficients as anything less than zero makes the objective infinite.

(iv) $0 < \lambda < \infty$, the magnitude of λ decides the weightage given to the different parts of the objective.

(v) The minimization objective

$$= \text{LS obj} + \lambda \text{ (sum of squares of coefficients)}$$

(vi) Where LS Obj is least square objective that is the linear regression objective without regularization.

3.6.2 Benefit of ridge-regression

- Ridge Regression solves the problem of overfitting, as just regular squared error regression fails to recognize the less important and uses all the them, leading to overfitting.
- Ridge regression adds a slight bias to fit the model according to the true values of the data.

3.6.3 Difference between Ridge Regression and Lasso Regression

(1) (i) Ridge regression is mostly used to reduce the overfitting in the model, and it includes all the features present in the model.

(ii) It reduces the complexity of the model by shrinking the coefficients

(2) Lasso Regression helps to reduce the overfitting in the model as well as feature selection.

3.7 THE GRADIENT DESCENT ALGORITHM

GQ. Explain the gradient descent algorithm.

- The Gradient Descent method lays the foundation for machine learning and deep learning techniques. Let's explore how does it work, when to use it and how does it behave for various functions.

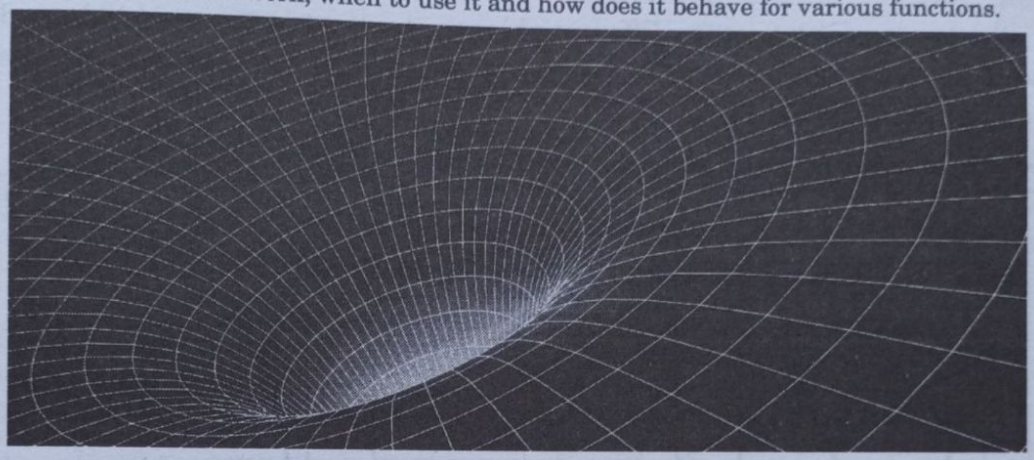


Fig. 3.7.1

3.8 INTRODUCTION

- Gradient descent (GD) is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in *machine learning* (ML) and *deep learning*(DL) to minimise a cost/loss function (e.g. in a linear regression). Due to its importance and ease of implementation, this algorithm is usually taught at the beginning of almost all machine learning courses.
- However, its use is not limited to ML/DL only, it's being widely used also in areas like:
  - Control engineering (robotics, chemical, etc.)
  - Computer games
  - Mechanical engineering
- That's why today we will get a deep dive into the math, implementation and behaviour of first-order gradient descent algorithm. We will navigate the custom (cost) function directly to find its minimum, so there will be no underlying data like in typical ML tutorials — we will be more flexible in terms of a function's shape.
- This method was proposed before the era of modern computers and there was an intensive development meantime which led to numerous improved versions of it but in this article, we're going to use a basic/vanilla gradient descent implemented in Python.

3.9 FUNCTION REQUIREMENTS

- Gradient descent algorithm does not work for all functions. There are two specific requirements. A function has to be:
  - Differentiable
  - Convex
- First, what does it mean it has to be differentiable? If a function is differentiable it has a derivative for each point in its domain - not all functions meet these criteria. First, let's see some examples of functions meeting this criterion:

let's see some examples of functions meeting this criterion:

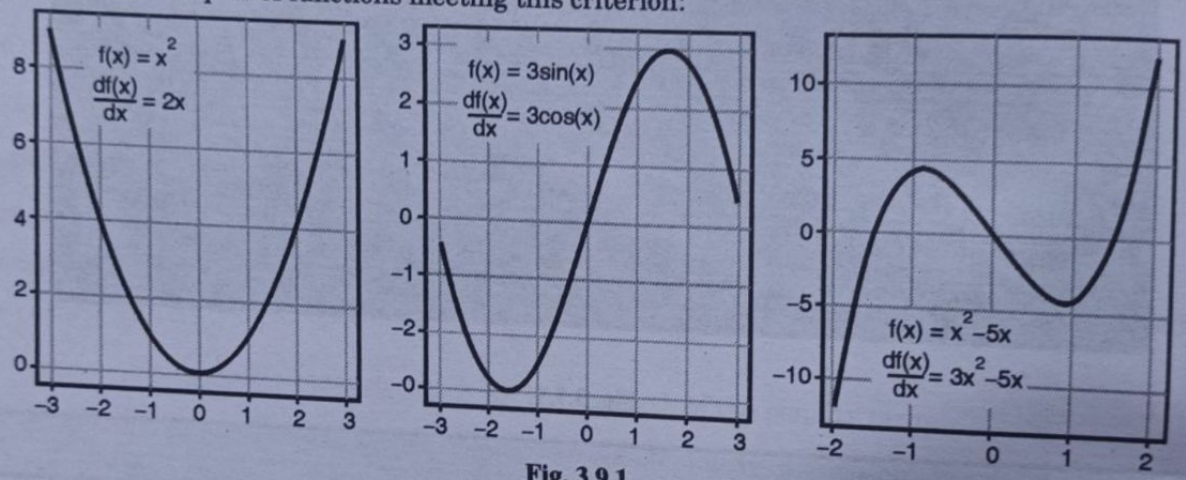


Fig. 3.9.1

- Typical non-differentiable functions have a step a cusp or a discontinuity:

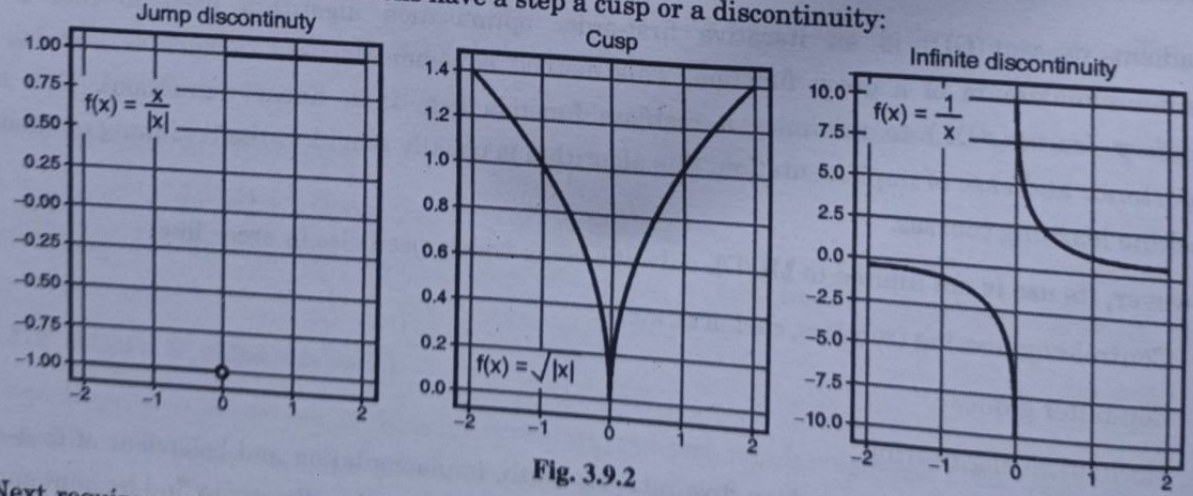


Fig. 3.9.2

- Next requirement - **function has to be convex**. For a univariate function, this means that the line segment connecting two function's points lays on or above its curve (it does not cross it). If it does it means that it has a local minimum which is not a global one.
- Mathematically, for two points x_1, x_2 laying on the function's curve this condition is expressed as:

$$f[\lambda x_1 + (1-\lambda) x_2] \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

- where λ denotes a point's location on a section line and its value has to be between 0 (left point) and 1 (right point), e.g. λ = 0.5 means a location in the middle.
- Below there are two functions with exemplary section lines.

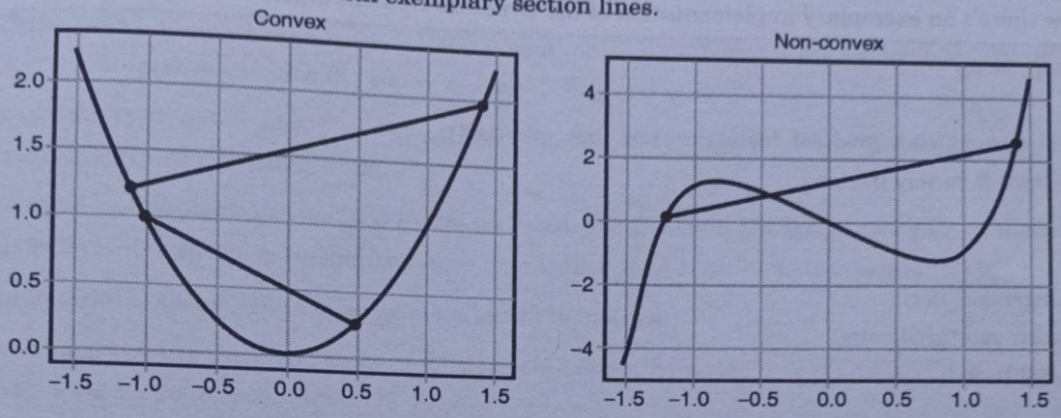


Fig. 3.9.3

- Exemplary convex and non-convex functions;

  i.e. $\frac{d^2 f(x)}{dx^2} > 0$

- Another way to check mathematically if a univariate function is convex is to calculate the second derivative and check if its value is always bigger than 0.

☞ Gradient Descent Algorithm

- Gradient Descent Algorithm iteratively calculates the next point using gradient at the current position, then scales it (by a learning rate) and subtracts obtained value from the current position (makes a step). It subtracts the value because we want to minimise the function (to maximise it would be adding). This process can be written as:

$$P_{n-1} = P_{n-\eta} \nabla f(P_n)$$

  - There's an important parameter η which scales the gradient and thus controls the step size. In machine learning, it is called learning rate and have a strong influence on performance.
  - The smaller learning rate the longer GD converges, or may reach maximum iteration before reaching the optimum point
  - If learning rate is too big the algorithm may not converge to the optimal point (jump around) or even to diverge completely.

- In summary, Gradient Descent method's steps are:

  (1) choose a starting point (initialisation)

  (2) calculate gradient at this point

  (3) make a scaled step in the opposite direction to the gradient (objective: minimise)

  (4) repeat points 2 and 3 until one of the criteria is met:

(i) maximum number of iterations reached

(ii) step size is smaller than the tolerance.

Below there's an exemplary implementation of the Gradient Descent algorithm (with steps tracking):

```
import numpy as np

def gradient_descent(start, gradient, learn_rate, max_iter, tol=0.01):
 steps = [start] # history tracking
 x = start

 for _ in range(max_iter):
  diff = learn_rate*gradient(x)
  if np.abs(diff)<tol:
   break
  x = x - diff
  steps.append(x) # history tracing

 return steps, x
```

☛ **This function takes 5 parameters**

1. starting point - in our case, we define it manually but in practice, it is often a random initialisation
2. gradient function - has to be specified before-hand
3. learning rate - scaling factor for step sizes
4. maximum number of iterations
5. tolerance to conditionally stop the algorithm (in this case a default value is 0.01)

3.10 EVALUATION METRICS

GQ. Explain Evaluation metrics.

☛ **Introduction**

- Machine Learning is a branch of Artificial Intelligence. It contains many algorithms to solve various real-world problems. Building a Machine learning model is not only the Goal of any data scientist but deploying a more generalized model is a target of every Machine learning engineer.
- Regression is also one type of supervised Machine learning and in this tutorial, we will discuss various metrics for evaluating regression Models and How to implement them using the sci-kit-learn library.

☛ **Regression**

- Regression is a type of Machine learning which helps in finding the relationship between independent and dependent variable.

- In simple words, Regression can be defined as a Machine learning problem where we have to predict discrete values like price, Rating, Fees, etc.

☞ **Why We require Evaluation Metrics?**

- Machine learning model cannot have 100 per cent efficiency otherwise the model is known as a biased model. which further includes the concept of overfitting and underfitting.
- It is necessary to obtain the accuracy on training data, But it is also important to get a genuine and approximate result on unseen data.
- So to build and deploy a generalized model we require to Evaluate the model on different metrics which helps us to better optimize the performance, fine-tune it, and obtain a better result.
- If one metric is perfect, there is no need for multiple metrics.
- Now, I hope you get the importance of Evaluation metrics. let's start understanding various evaluation metrics used for regression tasks.

3.11 MEAN ABSOLUTE ERROR(MAE)

GQ. Explain MAE, RMSE, R2

- MAE is a very simple metric which calculates the absolute difference between actual and predicted values.
- To better understand, let's take an example you have input data and output data and use Linear Regression, which draws a best-fit line.
- Now you have to find the MAE of your model. Find the difference between the actual value and predicted value
- so, sum all the errors and divide them by a total number of observations And this is MAE. And we aim to get a minimum MAE because this is a loss.

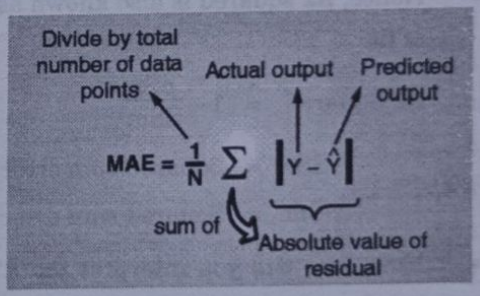


Fig. 3.11.1

☞ **Advantages of MAE**

- The MAE you get is in the same unit as the output variable.
- It is most Robust to outliers.

☞ **Disadvantages of MAE**

- The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.

```
from sklearn.metrics import mean_absolute_error
print("MAE",mean_absolute_error(y_test,y_pred))
```

3.12 ROOT MEAN SQUARED ERROR(RMSE)

As RMSE is clear by the name itself, that it is a simple square root of mean squared error.

$$RMSE = \sqrt{MSE}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)}$$

Advantages of RMSE

The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

Disadvantages of RMSE

- It is not that robust to outliers as compared to MAE.
- Most of the time people use RMSE as an evaluation metric and mostly when you are working with deep learning techniques the most preferred metric is RMSE.

3.13 R SQUARED (R2)

- R2 score is a metric that tells the performance of your model.
- In contrast, MAE and MSE depend on the context whereas the R2 score is independent of context.
- Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.

$$R2\ Squared = 1 - \frac{SSr}{SSm}$$

SSr = Squared sum error of regression line

SSm = Squared sum error of mean line

- Now, how will you interpret the R2 score ? suppose If the R2 score is zero then the above regression line by mean line is equal means 1 so 1-1 is zero. So, in this case, both lines are overlapping means model performance is worst, It is not capable to take advantage of the output column.
- Now the second case is when the R2 score is 1, it means when the division term is zero and it will happen when the regression line does not make any mistake, it is perfect. In the real world and it is not possible. So we can conclude that as our regression line moves towards perfection, R2 score move towards one. And the model performance improves.
- The normal case is when the R2 score is between zero and one like 0.8 which means your model is capable to explain 80 per cent of the variance of data.

Chapter Ends ...

□□□

UNIT IV

CHAPTER 4 Supervised Learning : Classification

Syllabus

Classification: K-nearest neighbour, Support vector machine.

Ensemble Learning: Bagging, Boosting, Random Forest, Adaboost.

Binary-vs-Multiclass Classification, Balanced and Imbalanced Multiclass Classification Problems,Variants of Multiclass Classification: One-vs-One and One-vs-All

Evaluation Metrics and Score: Accuracy, Precision, Recall, Fscore, Cross-validation, Micro- Average Precision and Recall, Micro-Average F-score, Macro-Average Precision and Recall, Macro-Average F-score.

4.1 CLASSIFICATION

GQ. What are the basic concepts of classification ?

We have studied classification and different classifiers earlier. Let's recall some key concepts of classification.

(1) **Classification** is a form of data analysis that extracts models describing data classes. It is a Supervised learning technique that is used to identify the category of new observations on the basis of training data. A classifier, or classification model, predicts categorical labels (classes). Numeric prediction models continuous-valued functions. Classification and numeric prediction are the two major types of prediction problems.

(2) **Decision tree induction** is a top-down recursive tree induction algorithm, which uses an attribute selection measure to select the attribute tested for each non-leaf node in the tree. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. ID3, C4.5, and CART are examples of such algorithms using different attribute selection measures. Tree pruning algorithms attempt to improve accuracy by removing tree branches reflecting noise in the data. Early decision tree algorithms typically assume that the data are memory resident.

(3) **Naive Bayesian classification** is based on Bayes' theorem of posterior probability. It assumes class-conditional independence that the effect of an attribute value on a given class is independent of the values of the other attributes. 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.

(4) **A rule-based classifier** uses a set of IF-THEN rules for classification. Rules can be extracted from a decision tree. These rules are easily interpretable and thus these classifiers are generally used to generate descriptive models. The condition used with "if" is called the antecedent and the predicted class of each rule is called the consequent. Rules may also be generated directly from training data using sequential covering algorithms.

4.2 K-NEAREST NEIGHBOR (KNN)

GQ. Explain KNN - algorithm.

K-means is an unsupervised learning algorithm used for clustering problems whereas KNN is a supervised learning algorithm used for classification and regression problems.

This is the basic difference between K-means and KNN algorithm.

4.2.1 K-NN Algorithm

K-NN algorithm at the raining phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example

- Suppose we have an image of a creature that looks similar to crow and Kingfisher, but we want to know either it is a Crow or Kingfisher.
- So for this identification, we can use KNN Algorithm as it works on a similarity measure. Our KNN-model will find the similar features of the new data set to the Crow and Kingfisher images and based on the most similar features it will put it in either Crow or Kingfisher category.
- KNN is one of the simplest machine learning algorithm based on supervised learning technique.
- KNN algorithm assumes the similarity between the new case or data and available cases and put the new case into the category that is most similar to the available category.
- KNN algorithm stares all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-defined category by using KNN algorithm.
- KNN-algorithm can be used for regression as well as for classification but mostly it is used for the classification problems.
- KNN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called as lazy learner algorithm because it does not lean from the training set immediately, instead it stores the dataset and at the time of classification, it performs an action on the dataset.

4.2.2 Need for KNN-Algorithm

GQ. What is the need of KNN

- Suppose we have two categories, i.e. Crow A and Kingfisher B and we have a new datapoint x_1, so this data point will lie in which of these categories.
- To solve this type of problem, we need a KNN algorithm and with this we can easily identify the category or class of a particular dataset.
- The K-NN working can be explained on the basis of the below algorithm.

Step 1 : Select the number K of the neighbors.

Step 2 : Calculate the euclidean distance of K number of neighbor.

Step 3 : Take the K nearest neighbors as per the calculate Euclidean distance.

Step 4 : Among these K-neighbors, count the number of data points in each category.

Step 5 : Assign the new data points to that category for which the number of the neighbor is maximum.

Now the model is ready to implement.

4.2.3 Selection of Value of K

- There is no particular method to determine the best value for 'K' in KNN-algorithm.
- So we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value of K such as K = 1 or K = 2 can be noisy and lead to effects of outliers in the model.
- Large values of K are good but it may find some difficulties.

4.2.4 Advantages of KNN Algorithm

GQ. Describe advantages and disadvantages of KNN algorithm.

1. It is simple to implement,
2. It is robust to the noisy training data.
3. It can be more effective if the training data is large.

4.2.5 Disadvantage of KNN Algorithm

1. Always needs to determine the value of K which may be complex some time.
2. The computation cost is high because of calculating the distance between the data points for all the training samples.

4.2.6 KNN Classification and Regression

GQ. Compare KNN Classification and Regression.

- In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k-nearest neighbor (K is a positive integer, typically small). If K = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In K-NN regression, the output is the property value for the object. This value is the average of the values of a K nearest neighbors.
- K-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evolution. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalising the training data can improve its accuracy dramatically.
- Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones, e.g. a common weighting scheme consists in giving each neighbor a weight of $\frac{1}{d}$, where d is the distance to the neighbor.
- The neighbors are taken from a set of objects for which the class (for K-NN classification) is known. This can be thought of as the training set for the algorithm, though no explicit training is required.
- A peculiarity of K-NN algorithm is that it is sensitive to the local structure of the data.

4.2.7 Parameter Selection

- The best choice of K depends upon the data, generally larger values of K reduces effect of the noise on the classification, but make boundaries between classes less distinct. A good value of K can be selected by various trial techniques. The special case where the class is predicted to be the class of the closest training sample (i.e. when K = 1) is called the nearest neighbor algorithm.

- The accuracy of the KNN algorithm is severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. A particularly popular approach is the use of evolutionary algorithm to optimise feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.
- In binary classification problem, it is helpful to choose K to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal K in this setting is via bootstrap method.

4.2.8 The 1-Nearest Neighbour Classifier

- The most intuitive nearest neighbor type classifier is the one nearest neighbor classifier that assigns a point x to the class of its closest neighbor in the feature space, i.e. $C_n^{1nn}(x) = y_{(1)}$.
- As the size of training data set approaches ∞, the one nearest neighbor classifier guarantees an error rate of no worse than twice the Bayer error rate (the minimum achievable error rate given the distribution of the data).

4.2.9 The Weighted Nearest Neighbour Classifier

- The K-nearest neighbor classifier can be viewed as assigning the K nearest neighbors a weight $\frac{1}{k}$ and all others '0' weight. This can be generalized to weighted nearest neighbor classifier. That is, where the i^{th} nearest neighbor is assigned a weight W_{n_i} with

$$\sum_{i=1}^{n} W_{ni} = 1$$

4.2.10 Properties

GQ. Describe KNN properties

- K-NN is a special case of a variable bandwidth, kernel density with a uniform kernel.
- The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets using an **approximate** nearest neighbor search algorithm makes K-NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years, those generally seek to reduce the number of distance evaluation actually performed.
- K-NN has some strong consistency results. As the amount of data approaches infinity, the two-class K-algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate. Various improvements to the K-NN are possible by using proximity graphs :

For multi-class KNN classification, an upper bound error rate of

$$R^* \le R_{K\text{-}NN} \le R^* \left[2 - \frac{MR^*}{(M-1)} \right]$$

- Where R* is the Bayes error rate (which is the minimal error rate possible), $R_{K\text{-}NN}$ is K-NN error rate, and M is the number of classes in the problem. For M = 2 and as the Byesian error rate R* approaches zero, the limit reduces the "not more than twice the Bayesian error rate."

4.2.11 Feature Extraction

- When the input data to an algorithm is too large to be processed and it is suspected to be redundant, then the input data will be transformed into a reduced representation set of features (also named feature vector). Transforming the input data into the set of features is called 'feature extraction'.
- If the features extracted are carefully chosen, it is expected that the feature set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying K-NN algorithm on the transformed data in the feature space.

4.2.12 Distance Functions

Euclidean : $\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$ **Manhatten :** $\sum_{i=1}^{k} |x_i - y_i|$

Minkoswski : $\left[\sum (|x_i - y_i|)^q\right]^{1/q}$

The above three distance measures are only valid for continuous variables. But in case of categorical variables, we use 'Hamming distance' which is a measure of number of instances.

Hamming distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

4.2.13 Data Points

- The number of data points that are taken into consideration is determined by the k-values. Thus, the k value is the 'core of the algorithm'. KNN classifier determines the class of a data point by the majority voting principle. If K is set to 5, the classes of 5 closest points are checked.
- KNN algorithm can also be used for regression problems.

4.2.14 Classification Accuracy

Classification accuracy of the KNN algorithm is found to be adversely affected by the presence of outliers, in the experimental datasets. An outlier score based on rank difference can be assigned to the points in these datasets by taking into consideration the distance and the density of their local and neighborhood points.

(A) Advantages

(i) Nonparametric architecture,

(ii) Simple and powerful,

(iii) Requires no training time

(iv) KNN can be used for recommendation systems. Although in the real world, more sophisticated algorithms are used for the recommendation system, KNN is not suitable for high dimensional data, but KNN is an excellent baseline approach for the system.

(v) KNN is not limited to merely predicting groups or values of data points. It can also be used in detecting anomalies. Identifying anomalies can be the end goal in itself, such as in fraud detection.

(B) Disadvantages

(i) Memory intensive

(ii) Classification and estimation are slow.

(iii) The value of K in the KNN-algorithm is related to the error rate of the model. Overfitting imply that the model is well on the training data but has poor performing when the new is coming.

4.2.15 Applications of KNN

(I) Applications of KNN in finance

(i) Forecasting stock market

(ii) Predict the price of a stock on the basis of company performance measures and economic data.

(iii) Currency exchange rate.

(iv) Bank bankruptcies

(v) Understanding and managing financial risk

(vi) Trading futures

(vii) Credit rating

(viii) Loan management

(ix) Bank customer profiling

(x) Money laundering analysis.

(II) Medicine

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.
- Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption spectrum of that person's blood.
- Identify the risk factors for prostate cancer, based on clinical and demographic variables.
- The KNN algorithm has been also applied for analyzing micro-array gene expression data, where the KNN algorithm has been coupled with genetic algorithms, which are used as a search tool.
- Other applications include the prediction of solvent accessibility in protein molecules, the detection of intrusions in computer systems, and the management of databases of moving objects such as computer with wireless connections.
- KNN is not limited to merely predicting groups or values of data points. It can also be used in detecting anomalies. Identifying anomalies can be the main aim, such as in fraud detection.

4.3 SUPERVISED LEARNING : SUPPORT VECTOR MACHINE

UQ. Define Support Vector Machine. Explain how margin is computed and optimal hyper-plane is decided ? **(Ref. - Dec. 19, 10 Marks)**

4.3.1 Optimal Decision Boundary

- A solution to the classification problem is a rule that partitions the features and assigns each and all the features and partition to the same class.
- The **'boundary'** of this partitioning is the decision boundary of the rule.
- The boundary that this rule produces is the optimal decision boundary.
- A decision boundary is the region of a problem space in which the output label of a **classifier** is **ambiguous**.
- If the decision surface is a hyperplane, then the classification problem is linear, and the classes are **linearly separable**.
- Decision boundaries are not always clear-cut. That is, the transition from one class in the **feature space** to another is not discontinuous, but gradual.

GQ. What is decision boundary in decision tree ?

- The first node of the tree called the "root node" contains the number of instances of all the classes respectively.
- Basically, we have to draw a line called "decision boundary" that separates the instances of different classes into different regions called "decision regions".

GQ. What is decision boundary ? How do you draw it and what is the advantage of it ?

A decision boundary is a line (in the case of two features), where all (or most) samples of one class are on one side of that line, and all samples of the other class are on one side of that line, and all samples of the other class are on the opposite side of the line. The line separates the two classes.

Maximum Margin Linear Separators

UQ. What is SVM ? Explain the following terms: separating hyperplane, margin and support vectors with suitable example.

UQ. What are the key terminologies of Support Vector Machine ? **(Ref. - May 15, 4 Marks)**

UQ. What is Support Vector Machine ? **(Ref. - May 16, 5 Marks)**

UQ. Illustrate Support Vector machine with neat labeled sketch. **(Ref. - May 17, Dec. 19, 4 Marks) (Ref. - May 19, 4 Marks)**

(1) Support Vector Machine is a type of supervised learning that can be used for classification or regression. Even if the data points are unseen (not from the training dataset), support vector machine classifies the data properly.

(2) Let's take an example of dataset that belongs to two different categories, and the distribution of data is proper means the data is separated from each other properly.

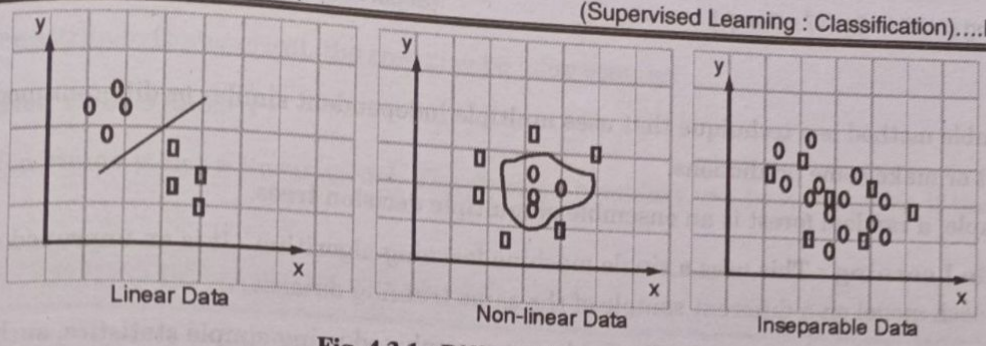


Fig. 4.3.1 : Different types of data

(3) In this case we can draw a straight line (decision boundary) on the graph in such a way that the input space is divided in to two regions.

(4) Data points that belongs to one category lies on one side of the decision boundary and the data points of other category lies on the opposite side. Such type of data is called as linearly separable data.

(5) **Separating hyperplane** is the line which is used to separate the dataset. If we are using simple 2-dimensional plots then it's just a line. We require a plane to separate the data if data is 3 dimensional. So we can say that if data is N dimensional, we require N-1 dimensional hyperplane.

(6) We want that our classifier should be designed in a manner that if a data point is far away from the decision boundary then we will be more confident about the prediction we have made.

(7) We would like to find the data point near to the separating hyperplane and also make sure that this point should be far away from the separating line as possible. This is called as **margin**. We would like to find the greatest possible margin, because if we trained our classifier on limited data or made a mistake, we would want it to be as robust as possible.

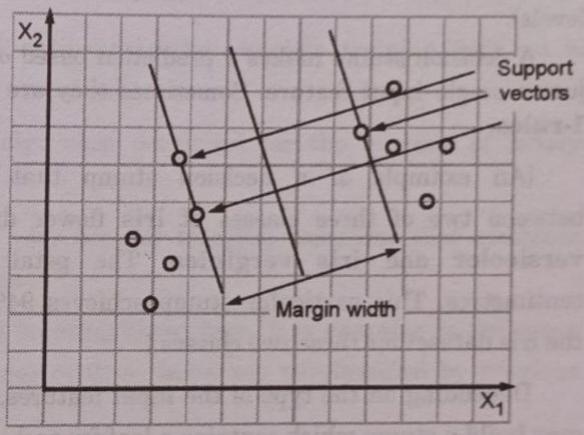


Fig. 4.3.2 : Support vectors and Margin

(8) **Support vectors** are the points which are nearest to the separating hyperplane. We have to maximize the distance between the support vectors and the separating line.

Distance between a hyperplane (w, b) and a point x is calculated as,

$$\text{Distance} = \frac{|\ w^t x + b\ |}{||w||}$$

4.4 ENSEMBLE LEARNING

GQ. Explain the concept of Ensemble learning. **(5 Marks)**

- An ensemble is a **machine learning model that combines** the predictions from two or more models.
- The models that contribute to the ensemble are called as ensemble members.
- They may be of the same type or of different types.
- They may or may not be trained on the same training data.

Remarks

1. An ensemble method is a technique that uses multiple independent similar or different models to derive an output or make some predictions.

   For example, a random forest is an ensemble of multiple decision trees.

2. **Ensemble Learning :** This uses a single machine learning algorithm. It is an unpruned decision tree. It trains each model on a different sample of the same training dataset.

   The predictions made by the ensemble members are combined using simple statistics, such as voting or averaging.

4.5 STUMPING ENSEMBLES

A decision stump is a machine learning model consisting of a one-level decision tree. That is, it is a decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its levels).

A decision stump makes a prediction based on the value of just a single input feature. Sometimes they are also called as **1-rules**.

[An example of a decision stump that discriminates between two of three classes of **iris** flower data set : **iris versicolor** and **iris verginica.** The petal width is in centimetres. This particular stump achieves 94% accuracy on the iris dataset for these two classes.]

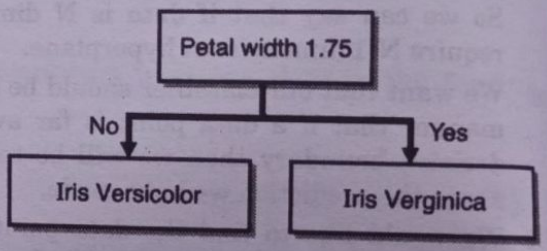


Fig. 4.5.1

Depending on the type of the input features, several variations are possible. For normal features, one may build a stump which contains a leaf for each possible feature value, or a stump with the two leaves, one of which corresponds to some chosen category, and the other leaf to all the other categories.

For binary features these two schemes are identical. A missing value may be treated as a yet another category.

4.5.1 For Continuous Features

Usually, some threshold feature value is selected, and the stump contains two leaves-for values below and above the threshold. However, rarely, multiple thresholds may be chosen and the stump therefore contains three or move leaves.

Decision stumps are often used as components (called "weak learners" or "base learner") in **machine learning ensemble** techniques such as **bagging** and **boosting**.

4.5.2 Remarks

(1) Meaning of stump in decision tree

- A decision stump is a decision tree, which uses only a single attribute for splitting.
- For discrete attributes, this means that the tree consists only of a single interior node (i.e., the root has only leaves as successor nodes).

- If the attribute is numerical, the tree may be more complex.

(2) Are decision stumps linear ?

A decision stump is not a linear model. The decision boundary can be a line, even if the model is not linear.

4.6 BOOSTING IN MACHINE LEARNING

GQ. What is Boosting in machine learning ?

- Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using a weak models in series.
- Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.
- AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification.
- AdaBoost is short for **Adaptive Boosting** and is a very popular boosting technique that combines multiple "weak classifiers" into a single "Strong Classifiers".
- AdaBoost is the best starting point for understanding boosting algorithms. It is adaptive in the sense that subsequent classifiers built are tweaked in favour of those instances misclassified by previous classifiers.
- It is sensitive to noisy data and outliers.
- AdaBoost uses multiple iterations to generate a single composite strong learner. It creates a strong learner by iteratively adding weak learners.
- During each phase of training, a new weak learner is added to the ensemble, and a weighting vector is adjusted to focus on examples that were misclassified in previous rounds. The result is a classifier that has higher accuracy than the weak learner classifiers.

4.7 TECHNIQUES TO IMPROVE CLASSIFICATION ACCURACY

GQ. Mention Techniques to Improve Classification Accuracy .

- In machine learning, no matter if we are facing a classification or a regression problem, the choice of the model is extremely important to have any chance to obtain good results.
- This choice can depend on many variables of the problem: quantity of data, dimensionality of the space, distribution hypothesis, etc.
- In ensemble learning theory, we call weak learners (or base models) models that can be used as building blocks for designing more complex models by combining several of them.

- The idea of ensemble methods is to try reducing bias and/or variance of such weak learners by combining several of them together in order to create a strong learner (or ensemble model) that achieves better performances.
- To outline the definition and practicality of Ensemble Methods, here we have used example of Decision tree classifier. However, it is important to note that Ensemble Methods do not only pertain to Decision Trees.
- A decision tree determines the predictive value based on series of questions and conditions. For instance, the simple Decision Tree shown in Fig. 4.7.1 determines on whether an individual should play outside or not.
- The tree takes several weather factors into account, and given each factor either makes a decision or asks another question. In this example, every time it is overcast, we will play outside.
- However, if it is raining, we must ask if it is windy or not? If windy, we will not play.
- But given no wind, tie those shoelaces tight because were going outside to play.

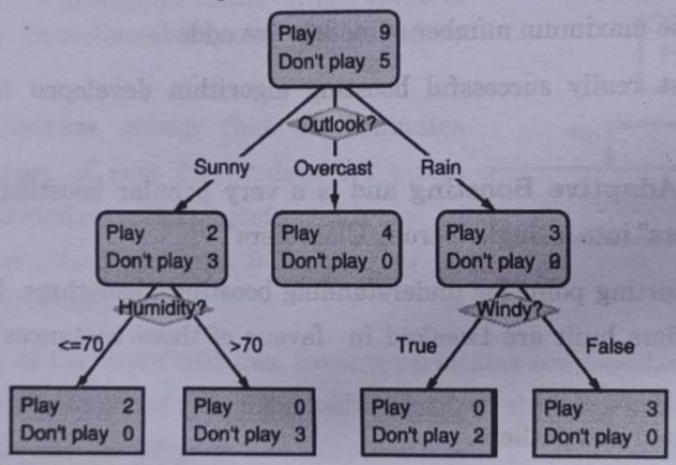


Fig. 4.7.1 : A decision tree to determine whether to play outside or not

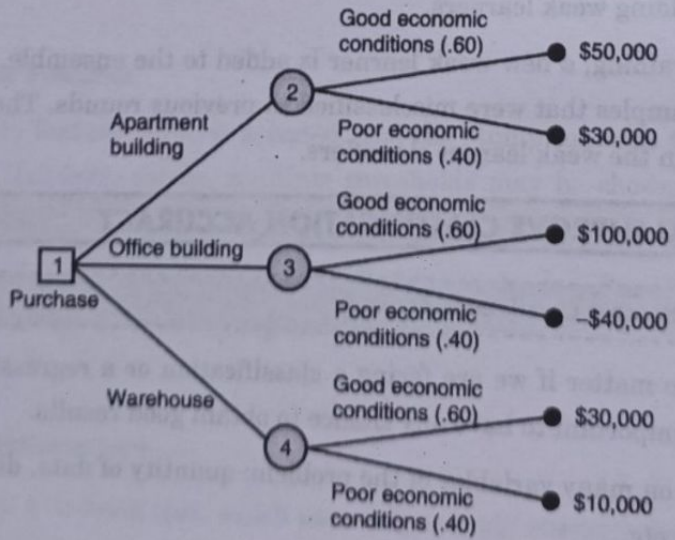


Fig. 4.7.2 : A decision tree to determine whether or not to invest in real estate

- Decision Trees can also solve quantitative problems as well with the same format. In the Tree to the left, we want to know whether or not to invest in a commercial real estate property. Is it an office building?
- A Warehouse? An Apartment building? Good economic conditions? Poor Economic Conditions? How much will an investment return? These questions are answered and solved using this decision tree.
- When making Decision Trees, there are several factors we must take into consideration: On what features do we make our decisions on? What is the threshold for classifying each question into a yes or no answer? In the first Decision Tree, what if we wanted to ask ourselves if we had friends to play with or not.
- If we have friends, we will play every time. If not, we might continue to ask ourselves questions about the weather. By adding an additional question, we hope to greater define the Yes and No classes.
- This is where Ensemble Methods come into picture! Rather than just relying on one Decision Tree and hoping we made the right decision at each split, Ensemble Methods allow us to take a sample of Decision Trees into account, calculate which features to use or questions to ask at eac h split, and make a final predictor based on the aggregated results of the sampled Decision Trees.

4.7.1 Introducing Ensemble Methods

- **Ensemble methods** is a machine learning technique that combines several base models in order to produce one optimal predictive model which helps to improve machine learning results.
- This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers and to allow them to vote. Ensembles tend to be more accurate than their component classifiers.

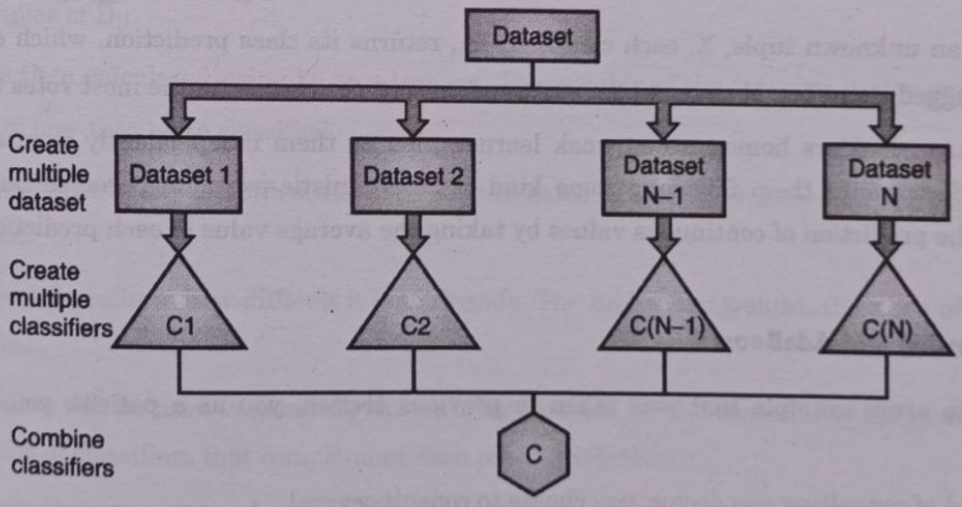


Fig. 4.7.3 : An overview of Ensemble methods/learning

GQ. Explain different types of ensemble classifiers.

- Different types of ensemble classifiers are:
  1. Bagging
  2. Boosting and AdaBoost
  3. Random Forests

4.7.2 Bagging (Bootstrap Aggregating)

- This approach combines **Bootstrapping** and **Aggregation** to form one ensemble model, that's why the name is **Bagging**.
- Consider yourself as a patient and you would like to have a diagnosis made based on the symptoms. Instead of asking one doctor, you may choose to ask several.
- If a certain diagnosis occurs more than any other, you may choose this as the final or best diagnosis.
- That is, the final diagnosis is made based on a majority vote, where each doctor gets an equal vote.
- If we replace each doctor by a classifier, and that's the basic idea behind bagging.
- Naturally, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.
- Given a sample of data, multiple bootstrapped subsamples are pulled.
- A Decision Tree is formed on each of the bootstrapped subsamples.
- Each training set is a bootstrap sample. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor.

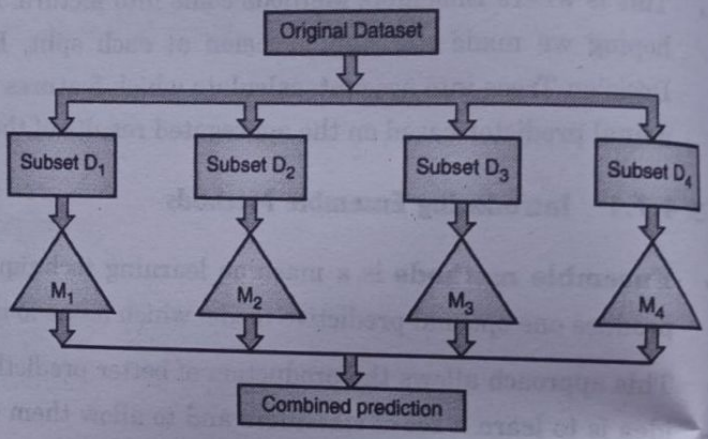


Fig. 4.7.4 : Bagging

- To classify an unknown tuple, X, each classifier, M_i, returns its class prediction, which counts as one vote. The bagged classifier, $M*$, counts the votes and assigns the class with the most votes to X.
- Bagging often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple.

4.7.3 Boosting and AdaBoost

- Consider the same example that was taken in previous section, you as a patient, you have certain symptoms.
- Now, instead of consulting one doctor, you choose to consult several.
- Suppose you assign weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made.
- The final diagnosis is then a combination of the weighted diagnoses. This is the basic idea behind **boosting.**

- Boosting often considers homogeneous weak learners, learns them sequentially in a very adaptative way (a base model depends on the previous ones) and combines them following a deterministic strategy.
- In boosting, weights are also assigned to each training tuple.
- A series of k classifiers is iteratively learned. After a classifier, M_i, is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to "pay more attention" to the training tuples that were misclassified by M_i.
- The final boosted classifier, $M*$, combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.
- In **adaptative boosting** (often called "adaboost"), we try to define our ensemble model as a weighted sum of L weak learners. It's a popular boosting algorithm.
- The basic idea is that when we build a classifier, we want it to focus more on the misclassified tuples of the previous round.
- Some classifiers may be better at classifying some "difficult" tuples than others. In this way, we build a series of classifiers that complement each other.
- We are given D, a data set of d class-labeled tuples, $(X_1, y_1),(X_2, y_2),...,(X_d, y_d)$, where y_i is the class label of tuple Xi. Initially, AdaBoost assigns each training tuple an equal weight of 1/d. Generating k classifiers for the ensemble requires k rounds through the rest of the algorithm.
- In round i, the tuples from D are sampled to form a training set, D_i, of size d. Sampling with replacement is used. This indicates the same tuple may be selected more than once.
- Each tuple's chance of being selected is based on its weight. A classifier model, M_i, is derived from the training tuples of D_i.
- Its error is then calculated using D_i as a test set. The weights of the training tuples are then adjusted according to how they were classified.
- If a tuple was incorrectly classified, its weight is increased. If a tuple was correctly classified, its weight is decreased.
- A tuple's weight reflects how difficult it is to classify. The higher the weight, the more often it has been misclassified.
- These weights will be used to generate the training samples for the classifier of the next round. This is how, a series of classifiers that complement each other are built.
- To compute the error rate of model M_i, we sum the weights of each of the tuples in D_i that M_i misclassified.

$$\text{error}(M_i) = \sum_{j=1}^{d} w_j \times \text{err}(X_j)$$

where $err(X_j)$ is the misclassification error of tuple X_j: If the tuple was misclassified, then $err(X_j)$ is 1; otherwise, it is 0. If the performance of classifier Mi is so poor that its error exceeds 0.5, then we abandon it. Instead, we try again by generating a new D_i training set, from which we derive a new M_i.

- The error rate of M_i affects how the weights of the training tuples are updated.
- If a tuple in round i was correctly classified, its weight is multiplied by $error(M_i)/(1 - error(M_i))$.
- Once the weights of all the correctly classified tuples are updated, the weights for all tuples (including the misclassified ones) are normalized so that their sum remains the same as it was before.
- To normalize a weight, we multiply it by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified tuples are increased and the weights of correctly classified tuples are decreased.
- To predict a class label for a tuple X, boosting assigns a weight to each classifier's vote, based on how well the classifier performed. The lower a classifier's error rate, the more accurate it is, and therefore, the higher its weight for voting should be. The weight of the classifier's vote is calculated as :

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

- For each class, c, we sum the weights of each classifier that assigned class c to X.
- The class with the highest sum is the "winner" and is returned as the class prediction for tuple X.
- Bagging is less susceptible to model overfitting. While bagging and boosting, both can significantly improve accuracy in comparison to a single model, boosting tends to achieve greater accuracy.

4.8 RANDOM FOREST

GQ. Explain random forest algorithm in detail.

4.8.1 Random Forest

- Random forest is a supervised machine learning algorithm that is used widely in classification and Regression problems.
- It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

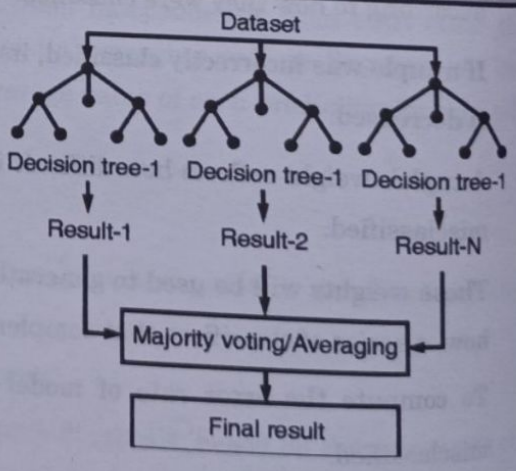


Fig. 4.8.1

4.8.2 Random Forest Algorithm

- Random forest Algorithm can handle the data set containing **continuous variables** in case of regression and **categorical variables** in case of classification.

- It performs better results for classification problems.

☞ Working of Random Forest Algorithm

Ensemble Technique

- Ensemble means combining multiple models. Here a collection of models make predictions rather than an individual model.
- Ensemble uses two types of methods
  1. **Bagging :** It creates a different training subset from sample training data with replacement. The final output is based on majority voting. For example, Random forest.
  2. **Boosting :** It creates sequential models such that the final model has the highest accuracy. Then it combines weak learners into strong learners. For example, ADA BOOST, XG BOOST.

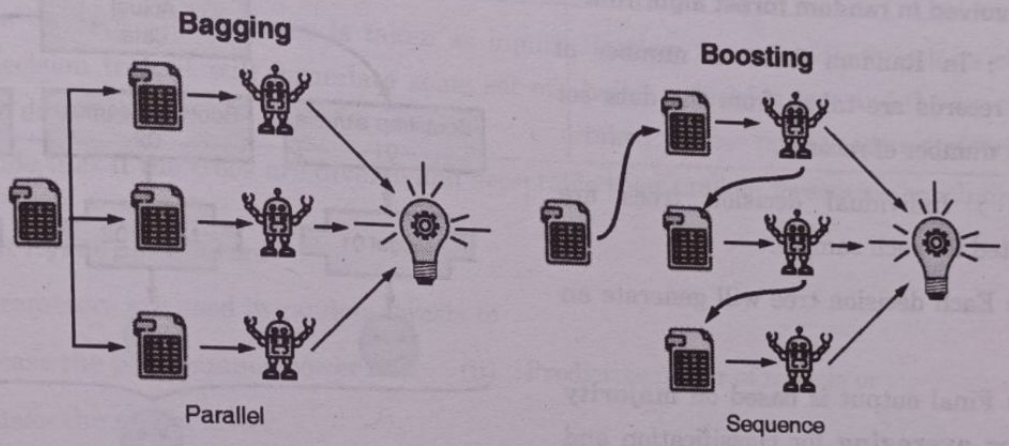


Fig. 4.8.2

Remark

Random forest works on the Bagging principle.

4.8.3 Bagging

- Bagging is also known as Bootstrap Aggregation. It is an ensemble technique used by random forest.
- Bagging chooses a random sample from the data set.
- Hence each model is generated from the samples (Bootstrap samples) provided by the original data with replacement. It is known as Row sampling.

4.8.4 Row Sampling

- This step of row sampling with replacement is called bootstrap.
- Each model is trained independently and it generates results. The final output is based on majority voting after combining the results of all models.

- The output that is based on majority voting is known as **aggregation.**
- We observe that the bootstrap sample is taken from the actual data (Bootstrap sample 01, Bootstrap sample 02 and Bootstrap sample 03) with a replacement.
- Clearly each sample will not contain unique data.
- Now, the model, Model 01, Model 02, Model 03 obtained from this bootstrap sample are trained independently.

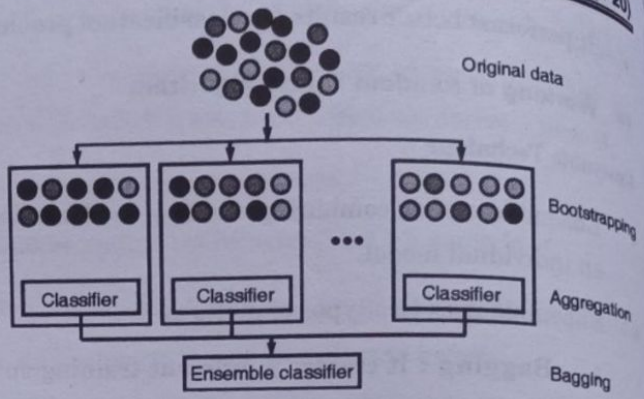


Fig. 4.8.3

- Each model generates results. And based on majority voting final output is obtained

☞ **Steps involved in random forest algorithm**

- **Step 1 :** In Random forest n number of random records are taken from the data set having k number of records.
- **Step 2 :** Individual decision trees are constructed for each sample.
- **Step 3 :** Each decision tree will generate an output.
- **Step 4 :** Final output is based on **majority voting or averaging** for classification and regression respectively.

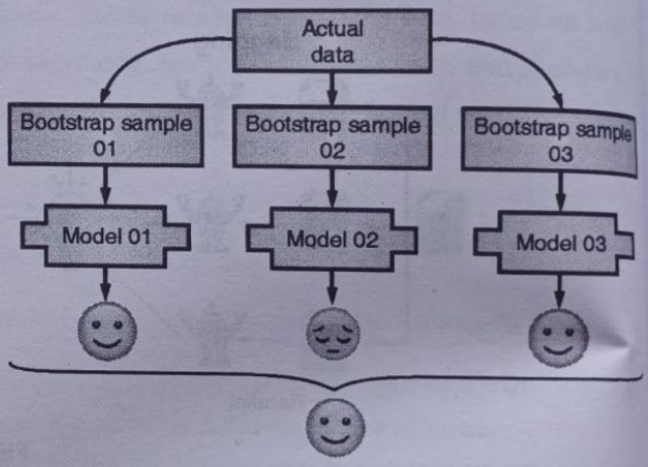


Fig. 4.8.4

4.8.5 Important Features of Random Forest

GQ. What are features of random forest.

1. **Diversity :** Not all attributes / Variables / features are considered while making an individual tree, each tree is different.
2. **Immune to the curse of dimensionality :**
   Since each tree does not consider all the features, the feature-space is reduced.
3. **Parallelisation :** Each tree is created independently out of different data and attributes. This implies that we can make full use of CPU to build random forests.
4. **Train-Test split :** In a random forest we don't have to segregate the data for training and testing. It is because there will always be 30% of the data which is not seen by the decision-tree.
5. **Stability :** Stability arises because the result is based on majority voting / averaging.

4.8.6 Difference Between Decision Tree and Random Forest

GQ. Mention difference between decision tree and random forest.

Actually, Random forest is a collection of decision trees, still there are differences in their behaviour.

Sr. No.	Decision Trees	Random Forest
1.	Decision trees normally suffer from the problem of overfitting. Since it is allowed to grow without any control.	Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2.	A single decision tree is faster in computation.	It is comparatively slower.
3.	When a dataset with features is taken as input by a decision tree, it will formulate some set of rules to do prediction.	Random forest randomly selects observations, builds a decision tree and the average result is taken. It does not use any set of formulas.

We conclude that if the trees are diverse and acceptable then random forests are much more successful than decision trees.

☞ **Important Hyper parameters**

Hyper parameters are used in random forests to

(i) Increase the performance power and (ii) Predictive power of models or

(iii) To make the model faster.

☞ **(I) Following hyper parameters increase the predictive power**

1. **n-estimators :** Number of trees the algorithm builds before averaging the predictions.
2. **Max-features :** Maximum number of features-random forest split a node.
3. **Mini-sample-leaf :** It determines the minimum number of leaves to split on internal node.

☞ **(II) Following hyper parameters increases the speed**

1. **n-jobs :** It tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is – 1, there is no limit.
2. **Random-state :** It controls the randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyper parameters and the same training data.
3. **OOB-score :** OOB implies out of the bag. It is a random forest cross-validation method. In this method, one-third of the sample is not used to train the data instead it is used to evaluate the performance.

These samples are called as out of bag samples.

4.8.7 Advantages and Disadvantages of Random Forest Algorithm

GQ. Explain advantages and disadvantages of random forest algorithm.

☞ **Advantages**

1. It can be used in classification and regression problems.
2. If solves the problem of overfitting as output is based on majority voting or averaging.
3. Even if data contains missing values or null values, it performs well.
4. Each decision tree created is independent of the other. It means that is shows the property of parallelisation.
5. It is highly stable as the average answers given by a large number of trees are taken.
6. If maintains diversity as all the attributes are not considered while making each decision tree even through it is not true in all cases.
7. It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature-space is reduced.
8. We need not segregate data to train and test, as there will always be 30% of the data which is not seen by the decision tree which is made out of **bootstrap**.

☞ **Disadvantages**

1. Random forest is highly complex when compared to decision trees where decisions can be made by following the path of the tree.
2. Training time is more compared to other models due to its complexity.

Whenever it has to make a prediction, each decision tree has to generate output for the given input data.

☞ **Remark**

1. Random forest can handle binary, continuous and categorical data.
2. Random forest can handle missing values.
3. Random forest is a fast, simple, flexible and robust model with some limitations.

4.9 COMPARISON BETWEEN BOOSTING RANDOM FOREST AND BOOSTING

GQ. Give comparison between boosting random forest and boosting.

Sr. No.	Random Forest	Boosting
1.	Random forest is a bagging technique	The decision trees are built additively, i.e. each decision tree is built one after another
2.	It is not a boosting technique.	In boosting, one is learning from other, which in turn boosts the learning.
3.	The trees in random forest run parallel.	Boosting is an approach to increases the complexity of models that suffer from high bias, that is, models that underfit the training data.

Sr. No.	Random Forest	Boosting
4.	There is no interaction between these trees while building the trees.	Boosting reduces error mainly by reducing bias, and also to some extent variance, by aggregating the output from many models.
5.	Random forest and bagging are "bagging" algorithms that aim to reduce the complexity of models that overfit the training data.	XG boost always gives more importance to functional space when reducing the cost of a model.
6.	Random forest uses fully grown decision trees (low bias, high variance). It tackles the error reduction in the opposite way : by reducing variance.	Gradient boosting may not be a good choice it there is lot of noise, as it can result in overfitting.
7.	Random forest tries to give more preferences to hyper parameters to optimise the model.	They also tend to be harder to tune than random forest.
8.	Random forest may not be give better performance compared to gradient boosting.	Gradient boosting trees can be more accurate than random forests.
9.	The basic idea in random forest is : although a single tree may be inaccurate, the collective decisions of a bunch of trees are likely to be right most of the time.	We train gradient boosting trees to correct each other's errors, they can capture complex patterns in the data.
10.	Forests are more robust and more accurate than a single tree. But, they are harder to interpret since each classification decision or regression output is not one but multiple decision paths.	If the data is noisy, the boosted trees may overfit and start modelling the noise.

4.10 BALANCED-IMBALANCED MULTI-CLASSIFICATION

- Class imbalance is a problem that occurs in machine learning classification problems. It says that the target class's frequency is highly imbalanced, i.e., the occurrence of one of the classes is very high compared to the other classes present.
- In other words, there is a bias or skewness towards the majority class present in the target.
- Fir example, suppose a binary classification has 10000 rows, and the minority target has only 100 rous. In this case the ratio is 100 : 1. This problem is a problem of a class-imbalance.
- Some frequently occurring examples are : Fraud detection, churn prediction, medical diagnosis, e-mail classification etc.

4.10.1 Handling of Multiclass Imbalanced Data

There are different methods of handling imbalance data, the most common methods are (i) oversampling and (ii) creating synthetic strategy.

(I) Oversampling Technique

SMOTE

SOMTE is an oversampling technique that generates synthetic samples from the data set which increases the predictive power for minority classes.

There is no loss of information but it has a few limitations.

Limitations

1. SMOTE is not very good for high dimensionality data.
2. Overlapping of classes may happen and can introduce more noise to the data.

So, to avoid this problem, we can assign weights for the class manually with the **'class-weight'** parameter.

Why to use class-weight?

Class weights modify the loss function directly by giving a penalty to the class with different weights.

It implies that, (we are)

Purposely increasing the power of the minority class and reducing the power of the majority class.

Hence, it gives better results than SMOTE,

(ii) Creating synthetic strategy

A few more techniques for getting the weights which can work for imbalanced learning problems :

1. Skleam utils
2. Counts to length, Ratio
3. Smoothen weights.
4. Sample weight strategy.

1. Skleam utils

Using skleam we can compute the class weight. Adding those weight to the minority classes, it can help the performance while classifying the models.

2. Counts to length, Ratio

Here, just divide the number of counts of each class with the number of rows.

3. Smoothen weights technique

This is the preferable method of choosing weights.

The log function smooths the weights for the imbalanced class.

4. Sample weight strategy.

Sample weight is an array of the same length as data, containing weights to apply to the model's loss for each sample.

This means that one should pass a weight for each class that is to be classified.

4.11 BALANCED ACCURACY

- Balanced accuracy is useful for multiclass classification.
- For a balanced dataset, the scores tend to be the same as Accuracy.

Issues with Balanced Accuracy

We mention some situations

(i) When data is balanced.

(ii) When model provides a wide range of possible outcomes (probability).

(iii) When the model is to give more preference to its positives than negatives.

(iv) In multiclass classification, where importance is not placed on some classes than others, bias can take place. It is because all classes have the same weights regardless of class frequency.

When there is a high skew, then balanced accuracy is not a perfect judge for the model.

4.12 VARIANTS OF MULTICLASS CLASSIFICATION

- In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of three or more classes.
- Many classification algorithms, for example multinomial logistic regression, can turn binary algorithms into multinomial classifiers by a variety of strategies.
- The existing multi-class classifications techniques can be categorized into :

(i) transformation to binary,
(ii) extension from binary, and
(iii) hierarchical classification

(i) Transformation to Binary

We discuss problem of multiclass classification to multiple binary classification.

It can be categorised into one vs rest and one vs one.

(a) One-vs-rest [or one-vs-all]

- One-vs-all strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.

- This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label.
- The training algorithm for one-vs-all learner constructed from a binary classification learner L is as follows :

Inputs

1. L, a learner (training algorithm for binary classifiers)
2. Samples X
3. labels y where $y_i \in \{1, ..., K\}$ is the label for the sample X_i.

Output

1. a list of classifiers f_k for KE {1, ...,K}

Procedure

For each K in {1, ..., K}

(i) Construct a new label vector Z where $Z_i = Y_i$; if $Y_k = K$ and $Z_i = 0$ otherwise.

(ii) Apply L to X, Z to obtain f_k

- Making decisions means applying all classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score.
- This strategy is popular, but it is a heuristic and suffers from several problems.
- Firstly, the scale of the confidence values may differ between the binary classifiers.
- Second, even if the class distribution is balanced, the binary classification learners see unbalanced distribution. It is because the set of negatives is much larger than the set of positives.

(b) One-vs-one

- In the one-vs-one reduction. One trains $\frac{K(K-1)}{2}$ binary classifiers fro a K-way multiclass problem. Each receives the samples of a pair of classes from the original training set, and learns to distinguish these two classes.
- At prediction time, a voting scheme is applied : All $\frac{K-(K-1)}{2}$ classifiers are applied to an unseen sample and the class that gets the highest number of '+1' predictions, gets predicted by the combined classifier.

- Like one vs all, one vs one suffers from ambiguities in that some regions of its input space may receive the same number of votes.

(ii) Extension from binary

- Here we discuss strategies of extending the existing binary classifiers to solve multi-class classification problems.
- Several algorithms are developed and they base on neural networks, decision trees. K-nearest neighbours, naïve Bayes, support vector machines to address multi-class classification problems. We have already discussed these techniques.

(iii) Hierarchical classification

- Hierarchical classification divides the out space into a tree of the multi-class classification problem.
- Each parent node is divided into multiple child nodes and the process is continued till each child node represents only one class.

4.13 CLASSIFICATION METRIC

GQ. Explain Classification Metric.

Classification models have discrete output, hence we need a metric that compares discrete classes in some form. Classification metrics evaluate a models performance and tell how good or bad the classification is

4.13.1 Performance Metrics in Machine Learning

- Performance metrics are a part of every machine learning. They tell us if we are making progress, and then put a number on it. All machine learning models, like linear regression, need a metric to judge performance.
- Every machine learning task can be broken down to either regression or classification.

4.13.2 Metrics are Different from Loss Functions

- Loss functions show a measure of model performance. They are used to train a machine learning model (using some kind of optimisation like gradient descent or stochastic gradient descent) and they are generally differentiable in the model's parameters.
- Metrics are used to monitor and measure the performance of a model (during training and testing) and don't need to be differentiable.

4.13.3 Classification Metrics

The most frequent classification evaluation metric should be 'Accuracy'. We mention some aspect

(1) The confusion matrix for a 2-class classification problem

(2) The key-classification metrics : Accuracy, Recall, precision and F1-score.

(3) The difference between Recall and Precision in specific cases

(4) Decision thresholds

(1) Confusion matrix

- Evaluation of the performance of a classification model is based on the counts of test records, correctly or incorrectly predicted by the model.
- The confusion matrix provides a more insightful picture which is not only the performance of a predictive model, but also it tells, which classes are being predicted correctly and incorrectly. And also it tells, what type of errors are being made.
- We see how the 4-classification metrics are calculated (TP, FP, FN, TN)
- We also present our predicted value compared with actual value in a confusion matrix

	Actual value	
	Positive	**Negative**
Positive	TP (True Positive)	FP (False Positive)
Negative	FN (False Negative)	TN (True Negative)

Fig. 4.13.1

True Positive (TP)	Observation is positive, and is predicted to be positive
False Negative (FN)	Observation is positive but is predicted negative
True Negative (TN)	Observation is negative and is predicted to be negative
False Positive (FP)	Observation is negative, but is predicted positive

The confusion matrix is useful for measuring recall (also known as sensitivity) precision, specificity, accuracy and also AUC ROC curve

4.13.4 The Equations of 4 Key Classification Metrics

(i) $$\text{Accuracy} = \text{ACC} = \frac{TP + TN}{TP + TN + FP + FN}$$

(ii) $$\text{Precision : Precision} = \frac{TP}{TP + FP}$$

(iii) $$\text{Recall : Recall} = \frac{TP}{TP + FN}$$

(iv) $$F_1 \text{ score : } F_1 = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

Recall versus precision

(a) **Precision** is the ratio of 'True positives' to all the positives predicted by the model.

Low precision : Model predicts the more false positives, the lower the precision.

(b) **Recall (sensitivity) :** It is the ratio of **True Positives** to all the positives in your dataset.

Low recall : The more false negatives the model predicts, the lower the recall.

We mention example to illustrate the difference in the above cases

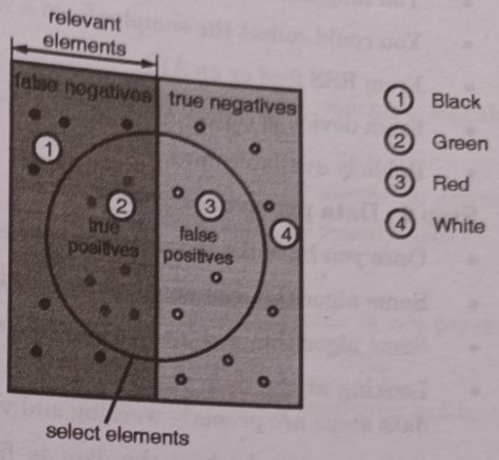


Fig. 4.13.2

(i) The result of TP will indicate that the COVID 19 residents diagnosed with COVID 19

(ii) The result of TN indicates healthy residents are with good health.

(iii) The result FP indicates that those actually healthy residents are predicted as COVID 19 residents

(iv) The result of FN indicates that those actual COVID 19 residents are predicted as the healthy residents

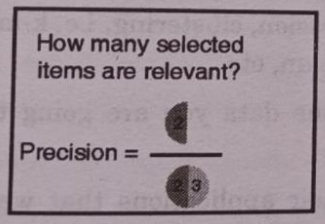


Fig. 4.13.3

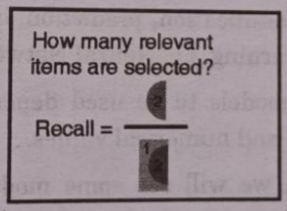


Fig. 4.13.4

- Let us study which picture will have the hightest cost ?
- If we predict COVID 19 residents as healthy patients, there would be a massive number of COVID 19 infections. The cost of **false negatives** is much higher than the cost of **false positives.**

4.14 STEPS IN ML MODELING

GQ. Describe steps in ML modeling in details.

Step 1 : Data Collection

- Given the problem you want to solve, you will have to investigate and obtain data that you will use to feed your machine.
- The quality and quantity of information you get are very important since it will directly impact how well or badly your model will work.

- You may have the information in an existing database or you must create it from scratch.
- You could collect the samples from a website and extracting data.
- From RSS feed or an API
- From device to collect wind speed measurement
- Publicly available data.

Step 2 : Data pre-processing

- Once you have the input data, you need to check whether it's in a useable format or not.
- Some algorithm can accept target variables and features as string; some need them to be integers.
- Some algorithm accepts features in a special format.
- Looking at the data you have passed in a text editor to check collection and preparation of input data steps are properly working and you don't have a bunch of empty values.
- You can also check at the data to find out if you can see any patterns or if there is anything obvious, such as a few data points greatly differ from remaining set of the data.
- Plotting data in 1, 2 or 3 dimensions can also help.
- Distil multiple dimensions down to 2/3 so that you can visualize the data.
- The importance of this step is that it makes you understand that you don't have any garbage value coming in.

Step 3 : Model Selection

- There are various models that we can select according to the objective that we might have: we will use algorithms of classification, prediction, linear regression, clustering, i.e. k-means or K-Nearest Neighbour, Deep Learning, i.e. Neural Networks, Bayesian, etc.
- There are various models to be used depending on the data you are going to process such as images, sound, text, and numerical values.
- In the 4.14.1 table, we will see some models and their applications that we can apply in our projects:

Table 4.14.1 : Model Selection

Model	Applications
Logistic Regression	Price prediction
Fully connected networks	Classification
Convolutional Neural Networks	Image processing
Recurrent Neural Networks	Voice recognition
Random Forest	Fraud Detection
Reinforcement Learning	Learning by trial and error
Generative Models	Image creation
K-means	Segmentation
k-Nearest Neighbors	Recommendation systems
Bayesian Classifiers	Spam and noise filtering

Step 4 : Model training

(1) Training

- A training set comprises of training examples which will be used to train machine learning algorithms.
- Good clean data from the first two steps is given as input to the algorithm. The algorithm extracts information or knowledge. This knowledge is mostly stored in a format that is readily useable by machine.
- In case of unsupervised learning, training step is not there because target value is not present. Complete data is used in the next step.
- You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.
- Remember to initialize the weights of your model randomly -the weights are the values that multiply or affect the relationships between the inputs and outputs- which will be automatically adjusted by the selected algorithm the more you train them.

(2) Testing

- To test machine learning algorithms what's usually done is to have a training set of data and a separate dataset, called a test set.
- In this step the information learned in the previous step is used. When you are checking an algorithm, you will test it to find out whether it works properly or not. In supervised case, you have some known values that can be used to evaluate the algorithm.
- In case of unsupervised, you may have to use some other matrices to evaluate the success. In either case, if you are not satisfied, you can again go back to step 4, change some things and test again.
- Mostly problem occurs in collection or preparation of data and you will have to go back to step 1.

(3) Process of Training and Testing

- Suppose we want to use a machine learning algorithm for classification. The next step is to train the algorithm, or allows it to learn. To train the algorithm we give as a input a quality data called as training set.
- Each training example has some features and one target variable. The target variable is what we will be trying to predict with our machine learning algorithms. In a training dataset the target variable is known. The machine learns by finding some relationship between the target variable and the features. In the classification tasks the target variables are known as classes. It is assumed that there will be a limited number of classes.
- The class or target variable that the training example belongs to is then compared to the predicted value, and we can get an idea about the accuracy of the algorithm.

- **Example :** First we will see some terminologies that are frequently used in machine learning methods. Let's take an example that we want to design a classification system that will classify the instances in to either Acceptable or Unacceptable. This kind of system is a fascinating topic often related with machine learning called *expert systems*.
- Four features of the various cars are stored inTable 4.14.2. The features or the attributes selected are Buying_Price, Maintenance_Price, Lug_Boot and Safety. Examples belong to Table 4.14.2 represents a record comprises of features.
- In Table 4.14.2 all the features are categorical in nature and takes limited disjoint values. The first two features represent the buying price and maintenance price of a car such as high, medium and low. Third feature shows the luggage capacity of a car as small, medium or big. Fourth feature represents whether the car has safety measures or not, which takes the value as low, medium or high.
- Classification is one of the important task in machine learning. In this application we want to evaluate the car out of a group of other cars. Suppose we have allinformation about car's Buying_Price, Maintenance_Price, Lug_Boot and Safety.
- Classification method is used to evaluate a given car as Acceptable or Unacceptable. Many machine learning algorithms are there that can be used for classification. The target or the response variable in this example is the evaluation of a car.
- Suppose we have selected a machine learning algorithm to use for classification. The main task in the classification is to train the algorithm, or allow it to learn. We give the experienced data as the input to train the algorithm which is called as training data.
- Let's assume training dataset contains 14 training records in Table 4.14.2. Suppose each training record has four features and one target or the response variable, as shown in Fig. 4.14.1. The machine learning algorithm is used to predict the target variable.
- In classification task the target variable takes a discrete value, and in the task of regression its value could be continuous.
- In a training dataset we have the value of target variable. The relationship that exists between the features and the target variable is used by machine for learning.
- The target variable is the evaluation of the car. Classes are the target variables in the classification task. In classification systems it is assumed that classes are to be of limited number.
- Attributes or features are the individual values that, when combined with other features, make up a training example. This is usually columns in a training or test set.
- A training dataset and a testing dataset, is used to test machine learning algorithms. First the training dataset is given as input to the program. Program uses this data to learn. Next, the test set is given to the program.
- The program decides which instance of test data belongs to which class. The predicted output is compared with the actual output of the program, and we can get an idea about the accuracy of the algorithm. There are best ways to use all the information in the training dataset and test dataset.

- Assume in car evaluation classification system, we have tested the program and it meets the desired level of accuracy.
- Knowledge representation is used to check what the machine has learned. There are many ways in which knowledge can be represented.
- We can use set of rules or a probability distribution to represent the knowledge.
- Many algorithms represent the knowledge which is more interpretable to humans than others. In some situations we may not want to build an expert system but we are interested only in the knowledge representation that's acquired from training a machine learning algorithm.

Table 4.14.2 : Car evaluation classification based on four features

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

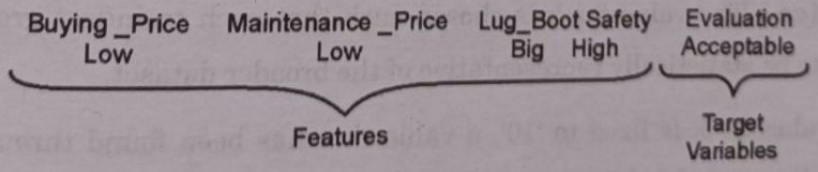


Fig. 4.14.1 : Features and target variable identified

(4) K-fold Cross validation

- Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
- The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation.

- When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.
- Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data.
- That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.
- It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.
- The general procedure is as follows :

  (1) Shuffle the dataset randomly.

  (2) Split the dataset into k groups

  (3) For each unique group:

  (i) Take the group as a hold out or test data set

  (ii) Take the remaining groups as a training data set

  (iii) Fit a model on the training set and evaluate it on the test set

  (iv) Retain the evaluation score and discard the model

  (4) Summarize the skill of the model using the sample of model evaluation scores

- Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.
- The k value must be chosen carefully for your data sample. Three common tactics for choosing a value for k are as follows:
- **Representative :** The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.
- **k=10 :** The value for k is fixed to '10', a value that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.
- **k=n :** The value for k is fixed to' n', where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.

Step 5 : Model Evaluation

- We have to check the machine designed against our evaluation data set that contains inputs that the model does not know and verify the precision of our already trained model.

- If the accuracy is less than or equal to 50%, that model will not be useful since it would be like tossing a coin to make decisions. If you reach 90% or more, you can have good confidence in the results that the model gives you.

(5) Accuracy

- The formula for accuracy is as follows.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- When dealing with classification problems we are attempting to predict a binary outcome. Is it fraud or not? Will this person default on their loan or not? Etc.
- So what we care about in addition to this overall ratio is number predictions that were falsely classified positive and falsely classified negative, especially given the context of what we are trying to predict.
- A 99% accuracy rate might be pretty good if we are trying to predict something like credit card fraud, but what if a false negative represents someone who has a serious virus that is apt to spreading quickly?
- Or a person who has cancer? That's why we have to breakdown the accuracy formula even further.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positive, TN = True Negatives,
FP = False Positives and FN = False Negatives.

- A **True Positive** is an outcome where the model correctly predicts the positive class.
- A **True Negative** is an outcome where the model correctly predicts the negative class.
- A **False Positive** is an outcome where the model incorrectly predicts the positive class.
- A **False Negative** is an outcome where the model incorrectly predicts the negative class.

(6) Precision

- Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.
- Precision is a good measure to determine, when the costs of False Positive is high. For instance, email spam detection.
- In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

$$\text{Precision} = \frac{TP}{TP + FP}$$

(7) Recall

- Recall actually calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive).
- Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.
- For instance, in fraud detection or sick patient detection. If a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be very bad for the bank.
- Similarly, in sick patient detection. If a sick patient (Actual Positive) goes through the test and predicted as not sick (Predicted Negative). The cost associated with False Negative will be extremely high if the sickness is contagious.

$$\text{Recall} = \frac{TP}{TP + Fn}$$

(8) Confusion matrix

- A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known.
- The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Let's start with an **example confusion matrix for a binary classifier** (though it can easily be extended to the case of more than two classes) :

N = 165	Predicted : NO	Predicted : YES
Actual : NO	50	10
Actual : YES	5	100

What can we learn from this matrix ?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.

In reality, 105 patients in the sample have the disease, and 60 patients do not.

Let's now define the most basic terms, which are whole numbers (not rates)

- **True Positives (TP) :** These are cases in which we predicted yes (they have the disease), and they do have the disease.

- **True Negatives (TN) :** We predicted no, and they don't have the disease.
- **False Positives (FP) :** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **False Negatives (FN) :** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")
- I've added these terms to the confusion matrix, and also added the row and column totals :

N = 165	Predicted : NO	Predicted : YES	
Actual : NO	TN = 50	FP = 10	60
Actual : YES	FN = 5	TP = 100	105
	55	110	

- This is a list of rates that are often computed from a confusion matrix for a binary classifier:
- **Accuracy :** Overall, how often is the classifier correct?
  - (TP + TN) / total = (100 + 50) / 165 = 0.91
- **Misclassification Rate :** Overall, how often is it wrong?
  - (FP + FN) / total = (10 + 5) / 165 = 0.09
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate :** When it's actually yes, how often does it predict yes?
  - TP / actual yes = 100 / 105 = 0.95
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate :** When it's actually no, how often does it predict yes?
  - FP / actual no = 10 / 60 = 0.17
- **True Negative Rate :** When it's actually no, how often does it predict no?
  - TN / actual no = 50 / 60 = 0.83
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"
- **Precision :** When it predicts yes, how often is it correct?
  - TP / predicted yes = 100 / 110 = 0.91
- **Prevalence :** How often does the yes condition actually occur in our sample?
  - actual yes / total = 105 / 165 = 0.64

Step 6 : Hyper Parameter Tuning

- If during the evaluation you did not obtain good predictions and your precision is not the minimum desired, it is possible that you have overfitting -or underfitting problems and you must return to the training step before making a new configuration of parameters in your model.
- You can increase the number of times you iterate your training data- termed epochs. Another important parameter is the one known as the "learning rate", which is usually a value that

multiplies the gradient to gradually bring it closer to the global -or local- minimum to minimize the cost of the function.

- Increasing your values by 0.1 units from 0.001 is not the same as this can significantly affect the model execution time. You can also indicate the maximum error allowed for your model. You can go from taking a few minutes to hours, and even days, to train your machine. These parameters are often called Hyperparameters.
- This "tuning" is still more of an art than a science and will improve as you experiment. There are usually many parameters to adjust and when combined they can trigger all your options. Each algorithm has its own parameters to adjust.
- To name a few more, in Artificial Neural Networks (ANNs) you must define in its architecture the number of hidden layers it will have and gradually test with more or less and with how many neurons each layer. This will be a work of great effort and patience to give good results.

Step 7 : Prediction

- You are now ready to use your Machine Learning model inferring results in real-life scenarios.
- In this step a real program is developed to do some task, and once again it is checked if all the previous steps worked as you expected.
- You might encounter some new data and have to revisit step 1-6.

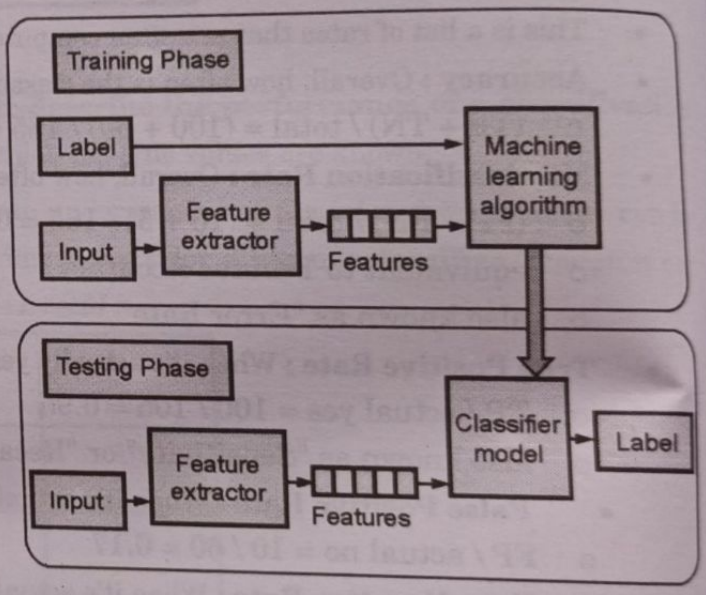


Fig. 4.14.2 : Typical example of Machine Learning Application

4.15 MODEL EVALUATION AND SELECTION

- Once our classification model is ready, we would like an estimate of how accurately the classifier can predict/classify the output class.
- Based on this, we will come to know whether training done is sufficient or not. We can even think of building more than one classifier and then compare their accuracy.
- Let's see now, what is accuracy? How can we estimate it? Are some measures of a classifier's accuracy more appropriate than others? How can we obtain a reliable accuracy estimate?

4.15.1 Metrics for Evaluating Classifier Performance

GQ. Mention various metrics for evaluating classifier performance

- The following list depicts various metrics/measures of evaluating how "accurate" your classifier is at predicting the class label of tuples :
  (1) Accuracy (2) Error rate
  (3) Precision (4) Sensitivity (Recall)
  (5) Specificity (6) F1 score
  (7) AOC-ROC (8) Log Loss
- Before discussing these measures, we need to understand with certain terminologies related to Confusion matrix.
- It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes.
- **A confusion matrix** is nothing but a table with two dimensions used for analyzing how well your classifier can recognize tuples of different classes viz. "Actual" and "Predicted" and furthermore, both the dimensions have "True Positives (TP)", "True Negatives (TN)", "False Positives (FP)", "False Negatives (FN)" as shown below:

		Actual Class 1	Actual Class 0	Total
Predicted Class	1	Ture Positives (TP)	False Positives (FP)	P
	0	False Negatives (FN)	True Negatives (TN)	N
		P'	N'	P + N

- **True Positives (TP)** : These refers to the positive tuples that were correctly labeled by the classifier. It is the case when both actual class and predicted class of data point is 1.
- **True Negatives (TN)** : These are the negative tuples that were correctly labeled by the classifier. It is the case when both actual class and predicted class of data point is 0.
- **False Positives (FP)** : These are the negative tuples that were incorrectly labeled as positive. It is the case when actual class of data point is 0 and predicted class of data point is 1.
- **False Negatives (FN)** : These are the positive tuples that were mislabeled as negative. It is the case when actual class of data point is 1 and predicted class of data point is 0.
- Here, TP and TN tell us when the classifier is getting things right, while FP and FN tell us when the classifier is getting things wrong. Now let's understand various evaluation measures.

1. **Accuracy (Recognition rate)**
   - The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.
   - In other words, Accuracy is the ratio of the number of correct predictions and the total number of predictions. The formula for Accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{P + N}$$

- Accuracy is useful when the target class is well balanced but is not a good choice for the unbalanced classes.
- Whereas other measures, such as sensitivity (or recall), specificity, precision, F, and Fβ, are better suited to the class imbalance problem, where the main class of interest is rare.
- In reality, data is always imbalanced for example Spam email, credit card fraud, and medical diagnosis.
- Hence, if we want to do a better model evaluation and have a full picture of the model evaluation, other metrics such as recall and precision should also be considered.

2. Error/Misclassification rate

- Error rate for a classifier, M, is simply 1 – accuracy(M), where accuracy(M) is the accuracy of the model M.
- We can also write this as :

$$\text{Error rate} = \frac{FP + FN}{P + N}$$

3. Recall/Sensitivity

- Recall is a measure of completeness i.e., what percentage of positive tuples are labeled as such.
- Sensitivity refers to True Positive (recognition) rate which is the proportion of positive tuples that are correctly identified.
- Recall and Sensitivity are similar

$$\text{Recall/Sensitivity} = \frac{TP}{TP + FN} = \frac{TP}{P}$$

4. Specificity

Specificity is the true negative rate which is the proportion of negative tuples that are correctly identified.

$$\text{Specificity} = \frac{TN}{N}$$

5. Precision

Precision is a measure of exactness i.e., what percentage of tuples labeled as positive are actually such.

$$\text{Precision} = \frac{TP}{TP + FP}$$

6. F-Score

- Precision and Recall are combined into a single measure to form F measures (also known as the F1 score or F-score).

- F1 Score is the harmonic mean of precision and recall. It is maximum when Precision is equal to Recall.

$$F = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F\beta = \frac{(1 + \beta 2) * \text{Precision} * \text{Recall}}{\beta 2\ \text{Precision} + \text{Recall}}$$

7. AUC-ROC

- The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes.
- And, The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'.
- In a ROC curve, the X-axis value shows False Positive Rate/Recall and Y-axis shows True Positive Rate/Sensitivity. Higher the value of X means higher the number of False Positives (FP) than True Negatives (TN), while a higher Y-axis value indicates a higher number of TP than FN. So, the choice of the threshold depends on the ability to balance between FP and FN.
- From the graph shown in Fig. 4.16.1, the greater the AUC, the better is the performance of the model at different threshold points between positive and negative classes.
- This simply means that when AUC is equal to 1, the classifier is able to perfectly distinguish between all Positive and Negative class points.
- When AUC is equal to 0, the classifier would be predicting all Negatives as Positives and vice versa. When AUC is 0.5, the classifier is not able to distinguish between the Positive and Negative classes.

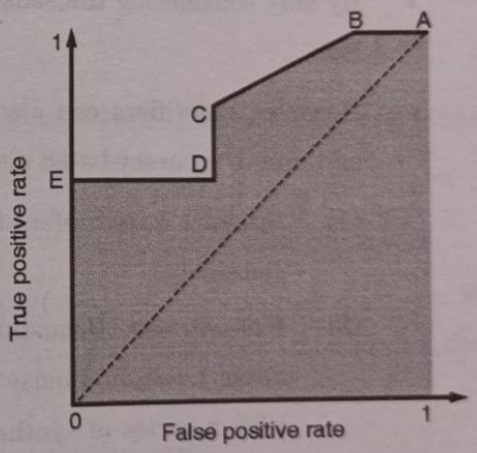


Fig. 4.15.1 : ROC Curve

8. Logarithmic Loss (Log Loss)

- It is also called Logistic regression loss or cross-entropy loss.
- It basically defined on probability estimates and measures the performance of a classification model where the input is a probability value between 0 and 1.
- It can be understood more clearly by differentiating it with accuracy.
- As we know that accuracy is the count of predictions (predicted value = actual value) in our model whereas Log Loss is the amount of uncertainty of our prediction based on how much it varies from the actual label.
- With the help of Log Loss value, we can have more accurate view of the performance of our model.

- Let's calculate these metrics for some reasonable real-world numbers. If we have 100,000 patients, of which 200 (20%) actually have cancer, we might see the following test results:

	Test Positive	Test Negative	Total
Patient Diseased	160	40	200
Patient Healthy	29940	69860	99800
Total	30100	69900	100000

For this data :

$$\text{Sensitivity} = TP/(TP + FN) = 160 / (160 + 40) = 80.0\%$$

$$\text{Specificity} = TN/(TN + FP) = 69,860 / (69,860 + 29,940)$$

$$= 70.0\%$$

- In other words, our test will correctly identify 80% of people with the disease, but 30% of healthy people will incorrectly test positive.
- By only considering the sensitivity (or accuracy) of the test, potentially important information is lost.
- However, classifiers can also be compared with respect to the following additional aspects, in addition to accuracy-based measures :

(1) **Speed :** Speed refers to the computational costs involved in creating and using the given classifier.

(2) **Robustness :** Robustness is the ability of the classifier to make correct predictions when the dataset contains noisy data or data with missing values. Robustness is typically assessed with a series of synthetic data sets representing increasing degrees of noise and missing values.

(3) **Scalability :** Scalability refers to the ability to construct the classifier efficiently given large amounts of data. Scalability is typically assessed with a series of data sets of increasing size.

(4) **Interpretability :** Interpretability refers to the level of understanding and insight that is provided by the classifier. Interpretability is subjective and therefore it is difficult to assess.

4.16 CROSS-VALIDATION IN MACHINE LEARNING

GQ. What is cross validation in machine learning ?
GQ. Explain methods for cross-validation

- Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. **It is a technique how a statistical model generalises to an independent dataset.**

- In **machine learning**, we need to test the stability of the model. It means based only on the training dataset, we cannot fit our model on the training dataset.
- For this purpose, we test our model on the sample which is not part of the training dataset. And then we deploy the model on that sample.
- This complete process comes under cross-validation.
- The basic steps of cross-validation are :
  - (i) Reserve a subset of the dataset as a validation set.
  - (ii) Provide the training to the model using the training dataset.
  - (iii) Evaluate model performance using the validation set.
- If the model performs well with the validation set, perform the further step, else check for the issues.

4.16.1 Methods used for Cross-Validation

The common methods used for cross-validation are :

(1) Validation set approach
(2) Leave-P-out cross-validation
(3) Leave one out cross- validation
(4) K-fold cross-validation
(5) Stratified K-fold cross-validation

Among these, K-fold cross-validation is easy to understand, and the output is less biased than other methods.

4.16.2 K-Fold Cross-Validation

- In each set (fold) training and the test would be performed precisely once during this entire process. It helps us to avoid overfitting. When a model is trained using all of the data in a single short and give the best performance accuracy.
- This K-fold cross-validation helps us to build the model which is a generalized one.
- To achieve this K-fold cross validation, we have to split the data set into three sets, training, testing and validation, with the challenge of the volume of the data.
- Here test and train data set will support building model and hyper parameter assessment.
- The model is validated multiple times based on the value assigned as a parameter and which is called K and it should be an **INTEGER**.
- The dataset X is divided randomly into K equal-sized parts, X_i, i = 1, 2, ..., K.
- To generate each pair, we keep one of the K parts out as validation set and combine the remaining (K – 1) parts to form the training set.
- Doing this K times, each time leaving out another one of the K parts out, we get K pairs :

$$V_1 = X_1, T_1 = X_2 \cup X_3 \cup ... \cup X_K$$

$$V_2 = X_2, T_2 = X_1 \cup X_3 \cup \ldots \cup X_K$$

$$\vdots$$

$$V_K = X_K, T_K = X_1 \cup X_2 \cup \ldots \cup X_{K-1}$$

- We come across two problems with this. First, to keep the training set large, we allow validation sets that are small.
- Second, the training sets overlap considerably namely, any two training sets share (K – 2) parts.
- K is typically 10 or 30. As K increases, the percentage of training instances increases and we get more robust estimators, but the validation set becomes smaller.
- Also, there is the cost of training the classifier K times, which increases as K is increased.
- As N increases, K can be smaller, if N is small, K should be large to allow large large enough training sets. (N is the number of instances).
- One extreme case of K-fold cross-validation is **leave-one-out** where given a dataset of N instances, only one instance is left out as the validation set (instance) and training uses (N – 1) instances.
- We then get N separate parts by leaving out a different instance at each iteration. This is typically used in applications such as medical diagnosis.

4.16.3 Life Cycle of K-fold Cross-Validation

- Let us have a generalised K-value. If K = 5, it means, we are splitting the given dataset into 5 folds and running the Train and Test.
- During each run, one fold is considered for testing and the rest will be for training and moving on with iterations, the below pictorial representation gives an idea of the flow of the fold-defined size

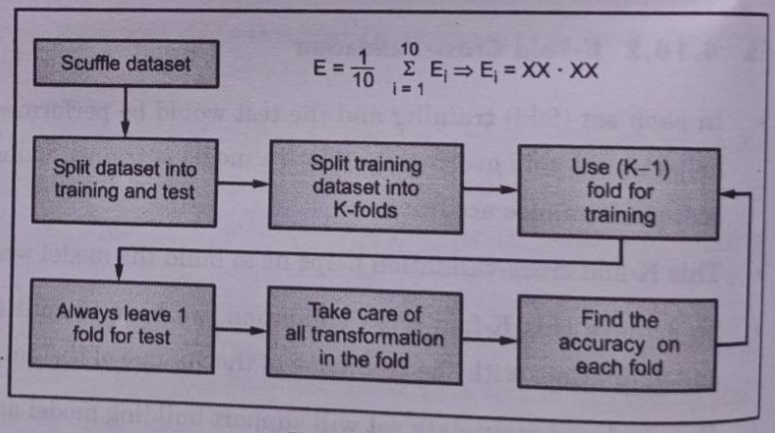


Fig. 4.16.1

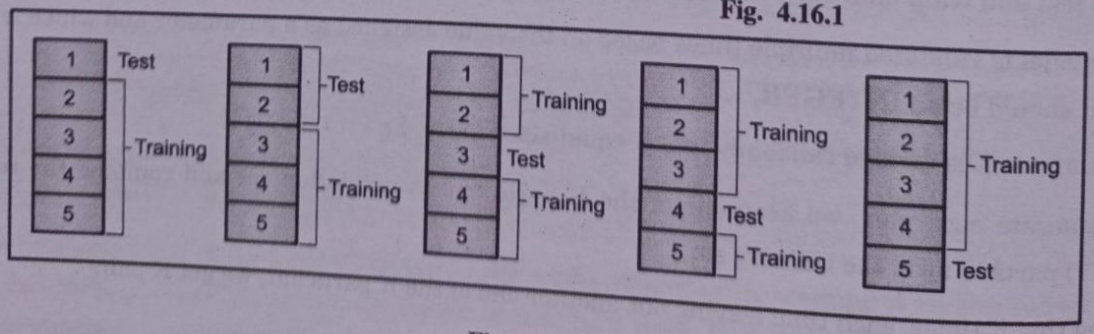


Fig. 4.16.2

- Here, each data-points is used, once in the hold-out set and K – 1 in Training. So during the full iteration at least once, one fold will be used for testing and the rest for training. In the above set, 5-Testing 20 Training.
- In each iteration, we will get an accuracy score and have to sum them and find the mean.
- Here we can understand how the data is spread in a way of consistency and will make a conclusion whether to go for the production with this model (or) NOT.

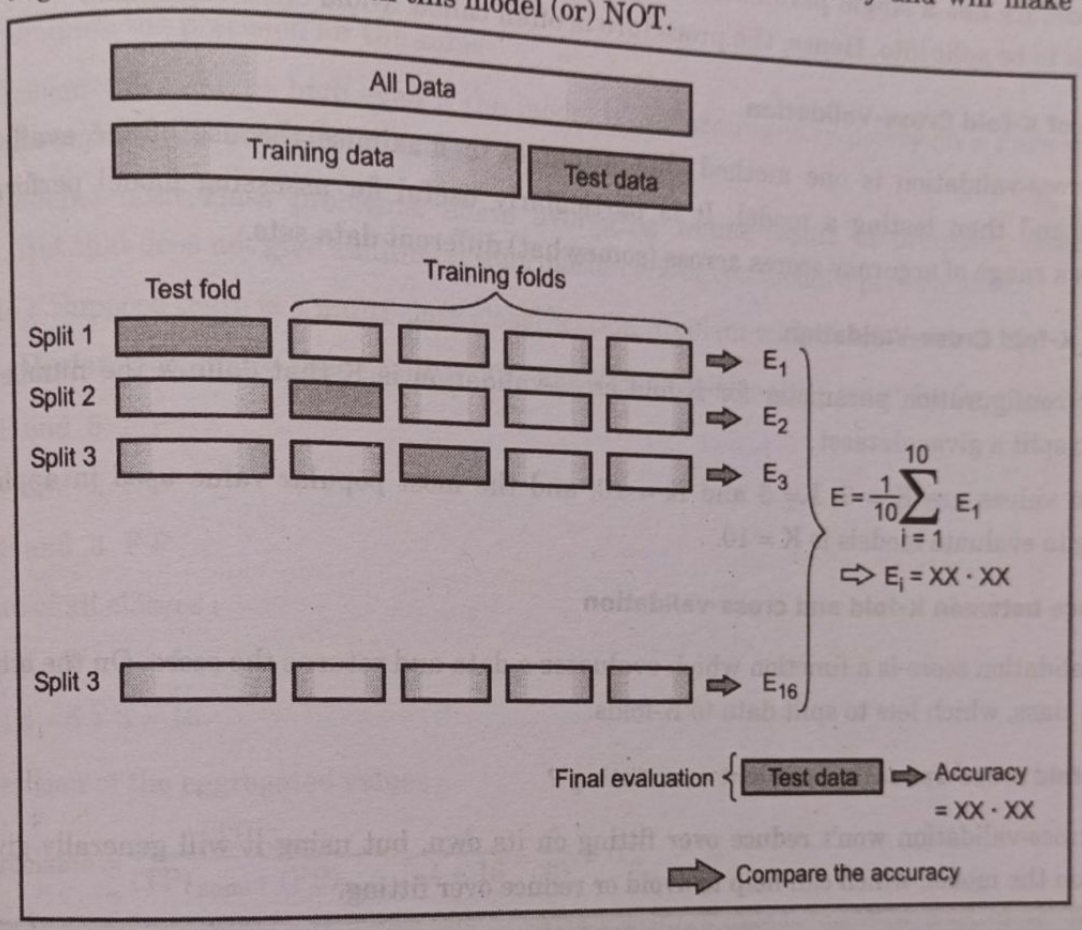


Fig. 4.16.3

4.16.4 Thumb Rules Associated with K-Fold

We discuss a few thumb-rules while dealing with K-fold.

(i) K should be always ≥ 2 and equal to number of records.

If 2, then just 2 iterations

If K = No of records in the dataset, then 1 for testing and n-for training.

(ii) The optimised value for the K is 10 and used with the data of good size, (i.e. commonly used).

(iii) If the K-value is large, then this will lead to less variance across the training set and limit the model currency, difference across the iterations.

(iv) The number of folds is generally inversely proportional to the size of the data set, which means, if the dataset size is too small, the number of folds can increase.

(v) Larger values of K eventually increase the running time of the cross-validation process.

4.16.5 Some Remarks

(1) Short note on K-cross Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called K that refers to the number of groups that a given data sample is to be split into. Hence, the procedure is often called K-fold cross-validation.

(2) Purpose of K-fold Cross-Validation

K-fold cross-validation is one method that attempts to maximise the use of the available data for training and then testing a model. It is particularly useful for assessing model performance, as it provides a range of accuracy scores across (somewhat) different data sets.

(3) Folds in K-fold Cross-Validation

The key configuration parameter for K-fold cross-validation is K that defines the number of folds in which to split a given dataset.

Common values are K = 3, K= 5 and K = 10, and the most popular value used in applied machine learning to evaluate models is K = 10.

(4) Difference between K-fold and cross-validation

Cross-validation score is a function which evaluates a data and returns the score. On the other hand, K-fold is a class, which lets to split data to K-folds.

(5) Does K-fold cross-validation prevent over fitting ?

K-fold cross-validation won't reduce over fitting on its own, but using it will generally gives a better insight on the model, which can help to avoid or reduce over fitting.

4.17 MICRO-AVERAGE PRECISION, RECALL, F-SCORE

Micro-precision measures the precision of the aggregated contributions of all classes.

It is micro-averaged precision.

Precision = 1 $\Rightarrow$ the model's predictions are perfect, all samples classified as the positive class are truly positive.

4.17.1 Emphasis on Common Classes

Micro-averaging puts more emphasis on the common classes in the data set. This is preferred behaviour for multi-label classification labels.

Precision is a metric used in binary classification problems.

Precision is defined as :

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive + False positive}}$$

Where **True positive** is when actual positive is predicted positive, and

False positive is when actual negative is predicted positive.

Micro-Averaging

- Micro-averaging is used when a problem has more than 2 labels that can be true,
- Micro-averaging is performed by first calculating the sum of all true positives and false positives, over all the classes.
- Then we compute the precision for the sums
- Micro-precision valve can be high even if the model is performing very poorly on a **rare** class, since it gives more weight to the **common classes.**
- For single-label multi-class problems, micro averaging would result in precision, almost equal to accuracy. But that does not give additional information about the model's performance.

Example : Suppose there is a multi-class classification problem with 3 classes (A. B, C).

First we calculate how many true positive (TP) and False positive (FP), we have for each class :

A : 2 T P and 8 F P

B : 1 T P and 5 F P

C : 1 T P and 3 F P

Aggregate of all classes :

$(TP)_{sum} : 2 + 1 + 1 = 4$

$(FP)_{sum} : 8 + 5 + 3 = 16$

Now, precision of the aggregated values :

$$\text{Micro-precision} = \frac{(TP)_{sum}}{(TP)_{sum} + (FP)_{sum}} = \frac{4}{4 + 16} = \frac{4}{20} = 0.2$$

4.18 WHAT IS RECALL ?

The recall is the ratio between the number of **positive samples** correctly classified as **positive** to the total number of **positive** samples.

The recall measures the model's ability to detect **positive samples.**

The higher the recall, the more positive samples detected.

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive + False negative}}$$

$$= \frac{TP}{TP + FN}$$

TP = True positive, FN = False negative

Remarks :

(i) Recall is independent of the number of negative sample classifications.

(ii) If the model classifies all positive samples as positive, then Recall will be 1.

(iii) Recall of a machine learning will be low when the value of :

TP + FN (denominator) > TP (Numerator)

(iv) Recall will be high when value of : TP (Numerator) > TP + FN (denominator).

Example : Recall of four different cases where each has the same recall as 0.667 but differs in the classification of negative samples.

Negative	Positive	Negative	Positive	Negative	Positive	Negative	Positive
X	✓	X	✓	✓	✓	✓	✓
X	✓	✓	✓	✓	✓	✓	✓
X	X	X	X	X	X	✓	X
← A →		← B →		← C →		← D →	

☞ **Here :**

(i) Case A has two negative samples classified as negative.

(ii) Case B has two negative sample classified as negative.

(iii) Case C has only one negative samples classified as negative.

(iv) Case D does not classify any negative sample as negative.

Since recall is independent of how the negative samples are classified in the model, hence we can neglect negative samples and only calculate all samples that are classified as positive.

Negative	Positive	Negative	Positive	Negative	Positive	Negative	Positive
–	✓	–	✓	–	✓	–	✓
–	✓	–	✓	–	✓	–	✓
–	x	–	X	–	X	–	X

Here there are two positive samples classified as positive while only 1 negative sample is correctly classified as negative.

$$\therefore \text{ Recall} = \frac{TP}{TP + FN} = \frac{2}{2+1} = \frac{2}{3} = 0.667$$

4.19 MICRO F1-SCORE

- Micro F-1 score (it is short form of micro-averaged F1 score) is used to assess the quality of multi-label binary problems.
- It measures the F1-score of the aggregated contributions of all classes.
- To select a model based on a balance between precision and recall, assessing F1-score is helpful.
- Micro F1-score = 1 is the best value (**perfect micro-precision** and micro-recall), and 0 is the worst value.
- Observe that precision and recall have the same relative contribution to F-1 score.

4.19.1 Emphasis on Common Labels

Micro-averaging will put more emphasis on the common labels in the dataset, because it gives each sample the Same importance.

This is the preferred behaviours for multi-class classification problems.

Labels that are very rare in the dataset, may not be influencing the overall F1-score if the model is performing well in the other more common genres.

Definition of Micro F1 :

Micro F1-score is defined as the harmonic mean of the precision and recall :

$$\text{Micro F1-score} = 2\left[\frac{(\text{Micro precision}) \cdot (\text{Micro-recall})}{(\text{micro precision}) + (\text{micro-recall})}\right]$$

4.19.2 Micro-averaging

Micro-averaging is used when a problem has 2 or more labels that can be true.

Micro-averaging F1-score is performed by first calculating the sum of all true positives, false positives, and false negatives over all the labels.

Then we compute micro-precision and micro-recall from the sums.

Finally we calculate the harmonic mean to get the micro F1-score.

4.19.3 Difference between Precision and Recall in Machine Learning

Sr. No.	Precision	Recall
1.	It helps to measure the ability to classify positive samples in the model.	It helps to measure how many positive samples were correctly classified by the ML model.
2.	While calculating the precision of a model, we have to consider both positive as well as negative samples that are classified.	While calculating the Recall of a model, we need consider all positive samples while all negative samples will be neglected.
3.	When the model classifies most of the positive samples correctly as well as many false positive samples, then the model is said to be a high recall and low precision model.	When a model classifies a sample as positive, but it can only classify a few positive samples, then the model is said to be high accuracy, high precision and low recall model.
4.	The precision of a machine learning model is dependent on both the negative and positive samples.	Recall of a machine learning model is dependent on positive samples and independent of negative samples.
5.	In precision we consider all positive samples that are classified as positive either correctly or incorrectly.	The recall cares about correctly classifying all positive samples. It does not consider if any negative sample is classified as positive.

4.19.4 The Need of Precision and Recall in Machine Learning Models

The use of precision and recall varies according to the type of problem being solved :

1. If there is a need of classifying all positive as well as negative samples as positive whether they are classified correctly or incorrectly, then we use precision.
2. But, if the aim is to detect only all positive samples, then use Recall. Here, we need not bother how negative samples are correctly or incorrectly classified the samples.

4.20 MACRO-AVERAGE, PRECISION, RECALL AND F-SCORE

4.20.1 Macro-Average Precision

- We calculate the precision for each class separately in an one vs All way. Then we take average of all precision values.
- For example, for 3 classes; a, b, c. We calculate p_a, p_b, p_c and **macro average** will be $\frac{p_a + p_b + p_c}{3}$.

4.20.2 Weighted Precision

- It is similar to macro precision, except that we take number of instances for each class into consideration as well.
- These acts as weights.
- For example, for 3 classes a, b, c; if number of instances are A, B, C, respectively, then the weighted precision is :

$$\frac{AP_a + Bp_b + Cp_c}{A + B + C}$$

4.20.3 Macro-recall Precision

- Macro-recall measures the average recall per class. (It is macro-averaged recall).
- Macro-recall = 1 implies that the model's
- Predictions are perfect, all truly positive samples were predicted as the positive class.
- And all classes are treated equally.
- Macro-recall will be low for models that only perform well on the common classes but performing poorly on the rare classes.
- Recall is a metric used in binary classification problem.

  Recall is defined as

$$\text{Recall} = \frac{\text{Truly positive}}{\text{Truly positive + False negative}} = \frac{TP}{TP + FN}$$

Where truly positive is when actual positive is predicted positive as
False negative is when actual positive is predicted negative.

4.20.4 The Macro-averaged F1-Score

- The macro F1-score is computed using the arithmetic mean of all the per-class F1 scores.
- This method treats all classes equally regardless of their support values.
- Thus macro-averaging score is to be used when all classes need to be treated equally to evaluate the overall performance of the classifier with respect to the most frequent class-labels.
- The macro-average of F-score is the harmonic mean.

$$\text{F1-Score} = 2\left[\frac{(\text{Precision}) \times (\text{Recall})}{\text{Precision} + \text{Recall}}\right]$$

What is a good F-measure score :

F1	Interpretation
70.9	Very good
0.8 – 0.9	Good
0.5 – 0.8	O. K.
< 0.5	Not good

- Thus, F1-score range from 0 to 1, where 1 is a perfect score indicating that the model predicts each observation correctly.
- A good F1-score is dependent on the data you are working with and the use case.

Chapter Ends ...

❑❑❑

UNIT V

CHAPTER 5

Unsupervised Learning

Syllabus

K-Means, K-medoids, Hierarchical, and Density-based Clustering, Spectral Clustering. Outlier analysis: introduction of isolation factor, local outlier factor.

Evaluation metrics and score: elbow method, extrinsic and intrinsic methods

5.1 UNSUPERVISED LEARNING : K MEANS CLUSTERING

GQ. Explain clustering in unsupervised learning.

- In unsupervised learning the most important task is the Clustering. Clustering is used to store data points in to related groups. In clustering advance knowledge is not present about the group definitions.

Definition : "Clustering is a process of partitioning a set of data in a set of meaningful sub-classes, called as clusters".

- In clustering we group the "similar" objects in one cluster and "dissimilar" objects in another cluster.

K- means Clustering

- To solve the well known 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.

UQ. Describe the essential steps of K-means algorithm for clustering analysis. **(Ref. - May 15, 5 Marks)**

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.

- The k-means algorithm does not guarantee the most optimal solution corresponding to global minimum objective function, although it can be proved that the process will always terminate.
- Initial random selection of cluster centers affects the performance of the algorithm. The k-means algorithm is applied for a number of times to reduce this effect.
- Let's assume that n sample data points $x_1, x_2, ..., x_n$ of the same class are present, and we know that the data points belongs to k clusters, $k < n$.
- Let m_i represents the mean of the data points in cluster i. x can be stored in cluster i, if $|| x - m_i ||$ is the minimum of all the k distances.
- The k-means procedure is shown below:
- Select initial values for the means $m_1, m_2, ..., m_k$
- Until no data point moves from one group to another

Use the calculated means to group the data points into clusters

For i from 1 to k

Mean of all of the samples for cluster i is used to replace m_i with the

end_for

end_until

- The K-means algorithm is implemented in three steps.
- Iterate until stable (= no data point move group)

1. Determine the centroid coordinate
2. Determine the distance of each data point to the centroid
3. Group the data points based on minimum distance.

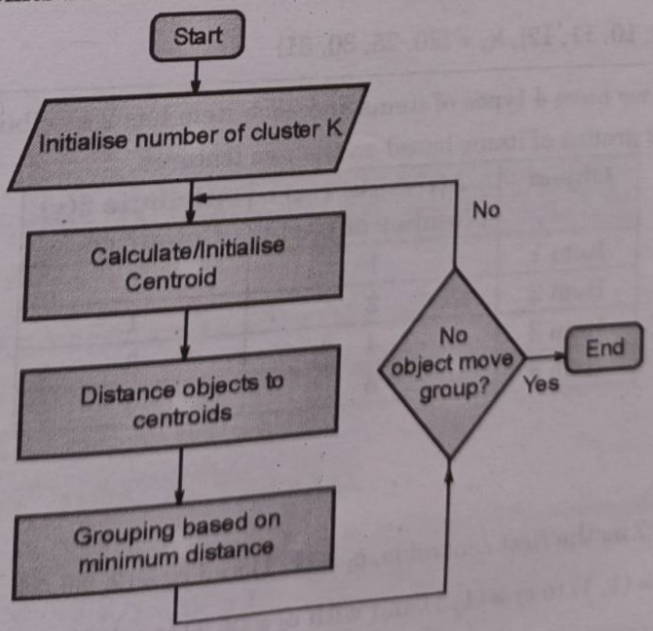


Fig. 5.1.1

5.1.1 Examples on K-means Clustering

Ex. 5.1.1 : Given { 2, 4, 10, 12, 3, 20, 30, 11, 25}. Assume number of clusters i.e. K = 2

☑ **Soln. :**

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.

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

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

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

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

Final clusters

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

Ex. 5.1.2 : Given {10, 4, 2, 12, 3, 20, 30, 11, 25, 31} Assume number of clusters i.e. K = 2

☑ **Soln. :**

Randomly assign alternative values to each cluster

$K_1 = (10, 2, 3, 30, 25\}, \quad k_2 = \{4, 12, 20, 11, 31\}$ $m_1 = 14, \quad m_2 = 15.6$

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

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

Ex. 5.1.3 : 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

☑ **Soln. :**

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

Ex. 5.1.4 : 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

☑ **Soln. :**

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

$$D^0 = \begin{bmatrix} 0 & 5 & 8.48 & 3.61 & 7.07 & 7.21 & 8.06 & 2.24 \\ 3.61 & 4.24 & 5 & 0 & 3.61 & 4.12 & 7.21 & 1.4 \\ 8.06 & 3.16 & 7.28 & 7.21 & 6.71 & 5.39 & 0 & 7.62 \end{bmatrix} \begin{matrix} c_1 = (2, 10) & \text{group 1} \\ c_2 = (5, 8) & \text{group 2} \\ c_3 = (1, 2) & \text{group 3} \end{matrix}$$

From the above object centroid distance matrix we can see,

- Data point 1 has minimum distance for group1, so we cluster data point 1 in group 1.
- Data point 2 has minimum distance for group3, so we cluster data point 2 in 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

$$D^1 = \begin{bmatrix} 0 & 5 & 8.48 & 3.61 & 7.07 & 7.21 & 8.06 & 2.24 \\ 5.66 & 4.12 & 2.83 & 2.24 & 1.41 & 2 & 6.40 & 3.16 \\ 6.52 & 1.58 & 6.25 & 5.7 & 5.7 & 4.52 & 1.58 & 6.04 \end{bmatrix} \begin{matrix} c_1 = (2, 10) & \text{group 1} \\ c_2 = (6, 6) & \text{group 2} \\ c_3 = (1.5, 3.5) & \text{group 3} \end{matrix}$$

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

$$C_1 = (2 + 4/2, 10 + 9/2) = (3, 9.5)$$

$$C_2 = (8 + 5 + 7 + 6/4, 4 + 8 + 5 + 4/4) = (6.5, 5.25)$$

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

$$D^2 = \begin{bmatrix} 1.12 & 2.35 & 7.43 & 2.5 & 6.02 & 6.26 & 7.76 & 1.12 \\ 6.54 & 4.51 & 1.95 & 3.13 & 0.56 & 1.35 & 6.38 & 7.68 \\ 6.52 & 1.58 & 6.52 & 5.7 & 5.7 & 4.52 & 1.58 & 6.04 \end{bmatrix} \begin{matrix} c_1 = (3, 9.5) & \text{group 1} \\ c_2 = (6.5, 5.25) & \text{group 2} \\ c_3 = (1.5, 3.5) & \text{group 3} \end{matrix}$$

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

$C_1 = (2 + 5 + 4/3, 10 + 9 + 8/3) = (3.67, 9)$

$C_2 = (8 + 7 + 6/3, 4 + 5 + 4/3) = (7, 4.33)$

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

$$D^2 = \begin{bmatrix} 1.95 & 4.33 & 6.61 & 1.66 & 5.2 & 5.52 & 7.49 & 0.33 \\ 6.01 & 5.04 & 1.05 & 4.17 & 0.67 & 1.05 & 6.44 & 5.55 \\ 6.52 & 1.58 & 6.52 & 5.7 & 5.7 & 4.52 & 1.58 & 6.04 \end{bmatrix} \begin{matrix} c_1 = (3.67, 9) & \text{group 1} \\ c_2 = (7, 4.33) & \text{group 2} \\ c_3 = (1.5, 3.5) & \text{group 3} \end{matrix}$$

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

UEx. 5.1.5 **Ref. - May 15, May 16, 10 Marks**

Apply K-means algorithm on given data for k = 3. Use $c_1(2)$, $c_2(16)$ and $c_3(38)$ as initial cluster centres.

Data : 2, 4, 6, 3, 31, 12, 15, 16, 38, 35, 14, 21, 23, 25, 30

☑ **Soln. :**

$c_1 = 2, \quad c_2 = 16, \quad c_3 = 38$

The numbers which are close to mean are grouped into respective clusters.

$k_1 = (2, 4, 6, 3\}, \quad k_2 = \{12, 15, 16, 14, 21, 23, 25\}, \quad k_3 = \{31, 35, 30\}$

Again calculate new mean for new cluster group.

$c_1 = 3.75, \quad c_2 = 18, \quad c_3 = 32$

New clusters

$k_1 = (2, 4, 6, 3\}, \quad k_2 = \{12, 15, 16, 14, 21, 23, 25\}, \quad k_3 = \{31, 35, 30\} \quad c_1 = 3.75, \quad c_2 = 18, \quad c_3 = 32$

Clusters remains unchanged

Final clusters

$K_1 = (2, 4, 6, 3\}, \quad k_2 = \{12, 15, 16, 14, 21, 23, 25\}, \quad k_3 = \{31, 35, 30\}$

5.2 K-MEDOIDS

GQ. Explain the concept of k-medoids.

- The K-medoids problem is a clustering problem similar to K-means.
- The K-medoids algorithms are partitioned. i.e. breaking the dataset up into groups and attempt to minimize the distance between the points in a cluster and a point which is the center of that cluster.
- K-medoids chooses actual data points as centers. The center of the cluster need not be one of the input data points (it is the average between the points in the cluster).

5.2.1 Definiation of Medoid

The medoid of a cluster is defined as the object in the cluster whose average dissimilarity to all the objects in the cluster is minimal, that is, it is a most centrally located point in the cluster.

In general, the k-medoids problem is hard to solve exactly. Hence, many heuristic solutions exist.

5.2.2 Partitioning Around Medoids (PAM)

PAM uses a greedy search which may not find the optimum solution, but it is faster than exhaustive search.

It works as follows :

(1) (BUILD) Initialise : greedily select k of the n data points an the medoids to minimize the cost.

(2) Associate each data point to the closest mediod.

(3) (SWAP) while the cost of the configuration decreases.

(i) For each mediod m, and for each non-medoid data point o :

(a) Consider the swap of m and o, and compute the cost change.

(b) If the cost change is the current best, remember this m and o combination.

(ii) Perform the best swap for m_{best} and o_{best}, if it decreases the cost function.

Otherwise the algorithm terminates.

5.2.3 Run-time of PAM Algorithm

The run-time complexity of the original PAM algorithm per iteration of (3) is $o(k(n-k)^2)$

This run- time can be reduced to $o(n^2)$. This is achieved by splitting the cost change into three parts such that computation can be shared (it is fast PAM).

Other Algorithm

Algorithms other than PAM have also been suggested and is known as "Alternating" heuristic, as it alternates between two optimisation steps.

1. Select initial medoids randomly.

2. Iterate while the cost decreases :
   (i) In each cluster, make the point that minimises the sum of distances within the cluster-medoid.
   (ii) Reassign each point to the cluster defined by the closest medoid determined in the previous step.

5.2.4 Comparison between K-means and k-Medoids Problem

Sr. No.	K mean	k-medoids
1.	k-means is a clustering problem	k-medoids is also clustering problem.
2.	k-mean algorithm partitions the dataset to minisise the distance between the points in the cluster.	k-medoids algorithm also partitions the dataset to minimize the distance between the points in the cluster.
3.	k-means In k-means, centre of a cluster is not necessarily one of the input data points	k-medoids chooses actual data points as centers and allows for accessibility of the input data points.
4.	k-means generally require **Euclidean distance** for efficient solutions	k-medoids can be used with arbitrary dissimilarity measures.
5.	k-means uses Euclidean distance for efficient solution . Hence the effect of noise and outliers cannot be avoided.	k-medoids minimize a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances, it is more robust to noise and outliers.

5.3 UNSUPERVISED LEARNING : HIERARCHICAL CLUSTERING

GQ. Explain the concept of hierarchical clustering.

5.3.1 Hierarchical Clustering

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.

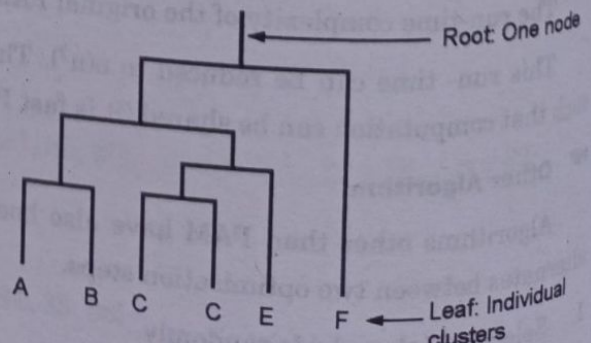


Fig. 5.3.1 : Dendogram

- In agglomerative, clustering is represented as a dendogram as in Fig. 5.3.1 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 i.e., 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 i.e., 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

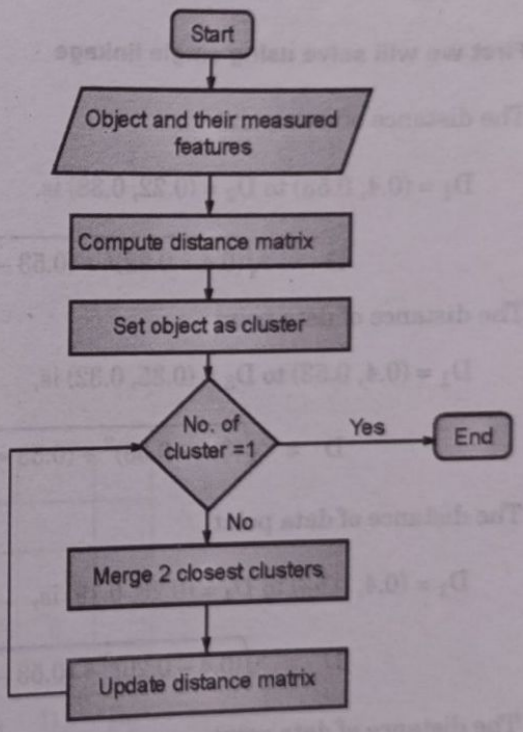


Fig. 5.3.2

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

5.3.2 Examples on Hierarchical Clustering

Ex. 5.3.1 : The table 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

☑ **Soln. :**

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	(D_2, D_5)	(D_3, D_6)	D_4
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

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	(D_2, D_5, D_3, D_6)	D_4
D_1	0		
(D_2, D_5, D_3, D_6)	0.22	0	
D_4	0.37	0.15	0

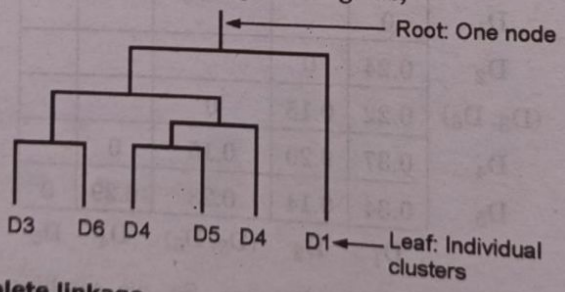
0.15 is smallest. (D_2, D_5, D_3, D_6) and D_4 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

	D_1	$(D_2, D_5, D_3, D_6, D_4)$
D_1	0	
$(D_2, D_5, D_3, D_6, D_4)$	0.22	**0**

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	D_2	D_3	D_4	D_5	D_6
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

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)

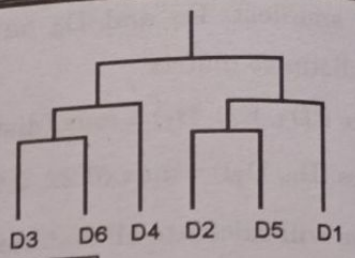
0.34 is smallest. (D_2, D_5) and D_1 have smallest distance so, we combine these two in one cluster and recalculate distance matrix.

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	D_2	D_3	D_4	D_5	D_6
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

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	D_2	(D_3, D_6)	D_4	D_5
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

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	(D_2, D_5)	(D_3, D_6)	D_4
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_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	(D_2, D_5)	(D_3, D_6, D_4)
D_1	0		
(D_2, D_5)	0.24	0	
(D_3, D_6, D_4)	0.27	0.26	0

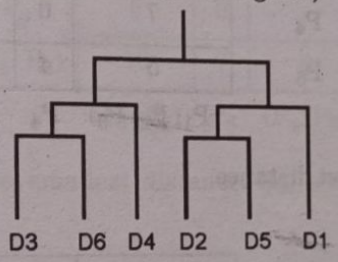
0.24 is smallest. (D_2, D_5) and D_1 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

	(D_2, D_5, D_1)	(D_3, D_6, D_4)
(D_2, D_5, D_1)	0	0
(D_3, D_6, D_4)	0.26	0

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,



Ex. 5.3.2 : Apply single linkage, complete linkage and average linkage on the following distance matrix and draw dendogram.

☑ **Soln. :**

First we will solve using single linkage

Distance matrix

	P_1	P_1	P_3	P_4	P_5
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

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

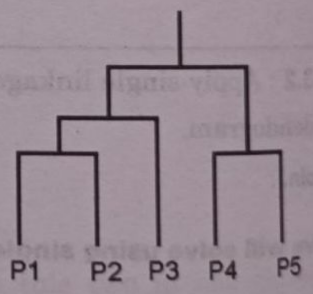
4 is smallest. P_4 and P_5 have smallest distance.

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)

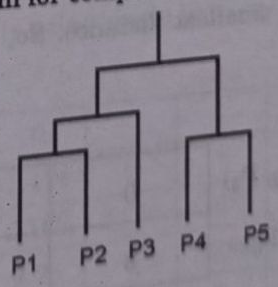
6 is smallest. (P_1, P_2) and P_3 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

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)

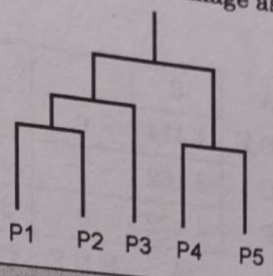
4.5 is smallest. (P_1, P_2) and P_3 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

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,



UEx. 5.3.3 **Ref - May 16, 10 Marks**

Apply Agglomerative clustering algorithm on given data and draw dendrogram. 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

☑ **Soln. :**

Distance matrix

A	0					
B	1.414	0				
C	3.162	2.828	0			
D	4.123	1	2.236	0		
E	2.236	1	2.236	2	0	
F	4.472	4.242	2	3	3.6	0
	A	B	C	D	E	F

1 is smallest. B, D and B, E have smallest distance. We can select anyone. So, we combine B, D in one cluster and recalculate distance matrix using single linkage.

Distance matrix

A	0				
B,D	1.414	0			
C	3.162	2.236	0		
E	2.26	1	2.236	0	
F	4.472	3	2	3.6	0
	A	B,D	C	E	F

1 is smallest. B, D and E have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

A	0			
B,D,E	1.414	0		
C	3.162	1	0	
F	4.472	3	2	0
	A	B,D,E	C	F

1 is smallest. B, D, E and C are combined together.

Distance matrix

A	0		
B,D,E,C	1.414	0	
F	4.472	2	0
	A	B,D,E,C	F

In the questions three clusters are asked with their allocated points. Three clusters are A, (B, D, E, C) and F.

UEx. 5.3.4 Ref - May 17, 10 Marks

For the given set of points identify clusters using complete link and average link using Agglomerative clustering.

	A	B
P_1	1	1
P_2	1.5	1.5
P_3	5	5
P_4	3	4
P_5	4	4
P_6	3	3.5

☑ **Soln. :**

First we will solve using complete linkage

Distance matrix

p1	0					
P2	0.707	0				
P3	5.656	4.949	0			
P4	3.605	2.915	2.236	0		
P5	4.242	3.535	1.414	1	0	
P6	5.201	2.5	1.802	0.5	1.118	0
	P1	P2	P3	P4	P5	P6

0.5 is smallest. P4 and P6 have smallest distance. We can select anyone. So, we combine this in one cluster and recalculate distance matrix using complete linkage.

Distance matrix

P1	0				
P2	0.707	0			
P3	5.656	4.949	0		
P4,P6	5.201	2.5	1.802	0	
P5	4.242	3.535	1.414	1.118	0
	P1	P2	P3	P4,P6	P5

0.707 is smallest. P1 and P2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

P1,P2	0			
P3	5.656	0		
P4,P6	5.201	2.236	0	
P5	4.242	1.414	1.118	0
	P1,P2	P3	P4,P6	P5

1.118 is smallest. P4, P6 and P5 are combined together.

Distance matrix

P1,P2	0		
P3	5.656	0	
P4,P5,P6	5.201	2.236	0
	P1,P2	P3	P4,P5,P6

2.236 is smallest. P4, P5, P6 and P3 are combined together.

P1,P2	0	
P3,P4,P5,P6	5.656	0
	P1,P2	P3,P4,P5,P6

Next we will combine all clusters in a single cluster.

Now we will solve using average linkage

Distance matrix

P1	0					
P2	0.707	0				
P3	5.656	4.949	0			
P4	3.605	2.915	2.236	0		
P5	4.242	3.535	1.414	1	0	
P6	5.201	2.5	1.802	0.5	1.118	0
	P1	P2	P3	P4	P5	P6

0.5 is smallest. P4 and P6 have smallest distance. We can select anyone .So, we combine this in one cluster and recalculate distance matrix using complete linkage.

Distance matrix

P1	0				
P2	0.707	0			
P3	5.656	4.949	0		
P4,P6	4.403	2.707	2.019	0	
P5	4.242	3.535	1.414	1.059	0
	P1	P2	P3	P4,P6	P5

0.707 is smallest. P1 and P2 have smallest distance. So, we combine this two in one cluster and recalculate distance matrix.

Distance matrix

P1,P2	0			
P3	5.302	0		
P4,P6	3.55	2.019	0	
P5	3.888	1.414	1.059	0
	P1, P2	P3	P4, P6	P5

1.059 is smallest. P4, P6 and P5 are combined together.

Distance matrix

P1,P2	0		
P3	5.302	0	
P4,P5,P6	3.66	1.817	0
	P1,P2	P3	P4,P5,P6

1.817 is smallest. P4,P5,P6 and P3 are combined together.

P1,P2	0	
P3,P4,P5,P6	4.07	0
	P1,P2	P3,P4,P5,P6

Next we will combine all clusters in a single cluster.

5.4 DENSITY-BASED CLUSTERING

GQ. What is D.B.C ? Explain its working.

- Density-based clustering refers to a method that is based on local cluster criterion such as density connected points. Density-based clustering is an unsupervised learning methology used in model building and machine learning algorithms
- The data points in the region separated by two clusters of low density are considered as noise.

- The surroundings of a radius ∈ of a given object are known as ∈-neighbourhood of the object.
- If the ∈-neighbourhood of the object contains at least at least a minimum numbers, Minpts of objects, then it is called a **core object.**.

5.4.1 Background of D-B Clustering

- There are two different parameters to calculate the density-based clustering.

Eps : It is considered as the maximum radius of the neighbourhood

Minpts (i) : Minpts refer to the minimum number of points in an examples neighborhood of that point.

NEps (i) : {K belongs to D and dist (i, k) ≤ Eps}.

Directly density reachable :

A point i is considered as the directly density reachable from a point k with respect to Eps, Minpts if i belongs to NEps(k).

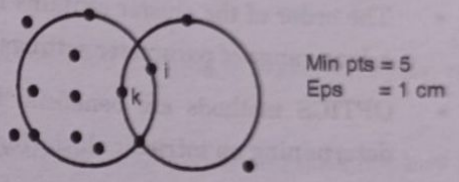


Fig. 5.4.1

Core point condition :

NEps (k) ≥ minimum points

5.4.2 Density Reachable

A point denoted by i is a density reachable from a point j with respect to Eps, Minpts if there is a sequence chain of a point i_1, ..., i_n, p_n = i such that i_1 + 1 is directly density reachable from i_1.

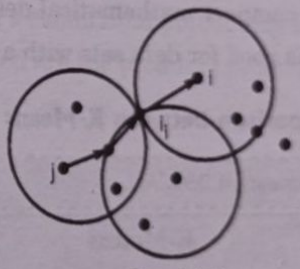


Fig. 5.4.1

5.4.3 Working of Density-Based Clustering

1. Let a set of objects be denoted by D', we can say that an object i is directly density reachable from the object j only if it is located within the ∈-neighborhood of j, and j is a core object.
2. An object i is density reachable from the object j with respect O ∈ and Minpts in a given set of objects, D' only, if there is a sequence of object chains point i_1, ..., i_n, i_1 = j, p_n = i such that i_i + 1 is directly density reachable from i_i with respect to ∈ and Minpts.
3. An object i is density connected object j with respect to ∈ and Minpts in a give set of objects, D', only if there is an object O belonging to D such that both points i and j are density reachable from O with respect to ∈ and Minpts.

Major Features of Density-Based Clustering

The primary features of density-based clustering are :

(i) It is a scan method,

(ii) It requires density parameters as a termination condition,

(iii) It is used to manage noise in data clusters.

(iv) Density-based clustering is used to identify clusters of arbitrary size.

5.4.4 Density-Based Clustering Methods

(1) DBSCAN

- DBSCAN stands for density-based spatial clustering of applications with noise.
- It depends on a density-based notion of cluster. It also identifies clusters of arbitrary size in the spatial database with outliers.

(2) OPTICS

- OPTICS stands for ordering points to identify the clustering structure. It gives a significant order of database with respect to its density-based clustering structure.
- The order of the cluster contains information equivalent to the density-based clustering related to a long range of parameter settings.
- OPTICS methods are beneficial for both automatic and interactive cluster analysis, including determining an intrinsic clustering structure.

(3) DENCLUE

It enables a compact mathematical description of arbitrarily shaped cluster in high dimension state of data, and it is good for data sets with a huge amount of noise.

5.4.5 Comparison between K-Means and DBSCAN

GQ. Compare kimean of DBSCAN.

Sr. No.	K-Means	DBSCAN
1.	k-means generally cluster all the objects.	DBSCAN discards objects that it defines as nose.
2.	k-means needs a prototype-based concept of a cluster.	DBSCAN needs a density-based concept.
3.	k-means has difficulty with non-globular clusters and clusters of multiple sizes.	DBSCAN is used to handle clusters of multiple sizes and structures and is not powerfully influenced by noise or outliers.
4.	k-means can be used for data that has a definite centroid, including a mean or median.	DBSCAN needs its definition of density, which further depends on the traditional Euclidean concept of density, which is significant for data.
5.	k-means can be used to sparse, high dimensional data, including file data.	DBSCAN generally implements poorly for such information. It is because Euclidean definition of density does not operate well for high dimensional data.
6.	The basic k-means algorithm is similar to a statistical clustering approach. It considers all clusters which come from spherical Gaussian distributions with several means but the equal covariance matrix.	DBSCAN creates no assumption about the distribution of the record.

Use of density-based clustering

The density-based clustering tool works by detecting areas where they are separated by areas that are empty or sparse.

Points that are not part of a cluster are labeled as **Noise**.

5.5 APPLICATIONS OF MACHINE LEARNING

GQ. Mention various applications of machine learning.

UQ. Write short note on : Machine learning applications. **(Ref. - May 16, May 17, 10 Marks)**

(1) Learning Associations

- A supermarket chain-one an example of retail application of machine learning is basket analysis, which is finding associations between products bought by customers :
- If people who buy P typically also buy Q and if there is a customer who buys Q and does not buy P, he or she is a potential P customer. Once we identify such customers, we can target them for cross-selling.
- In finding an association rule, we are interested in learning a conditional probability of the form P (Q/P) where Q is the product we would like to condition on P, which are the product / products which we know that customer has already purchased.

  P (Milk / Bread) = 0.7

- It implies that 70% of customers who buy bread also buy milk

(2) Classification

- A credit is an amount of money loaned by a financial institution.
- It is important for the bank to be able to predict in advance the risk associated with a loan. Which is the probability that the customer will default and not pay the whole amount back?
- In credit scoring, the bank calculates the risk given the amount of credit and the information about the customer. (Income, savings, collaterals, profession, age, past financial history). The aim is to infer a general rule from this data, coding the association between a customer's attributes and his risk.
- Machine Learning system fits a model to the past data to be able to calculate the risk for a new application and then decides to accept or refuse it accordingly.

  If income > Q_1 and savings > Q_2

  Then low – risk ELES high – risk
- Other classification examples are Optical character recognition, face recognition, medical diagnosis, speech recognition and biometric.

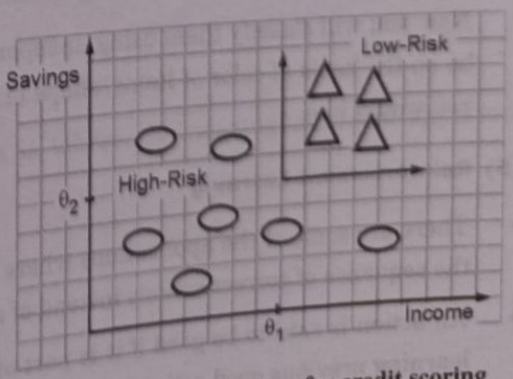


Fig. 5.5.1 : Classification for credit scoring

(3) Regression

- Suppose we want to design a system that can predict the price of a flat.
- Let's take the inputs as the area of the flat, location and purchase year and other information that affects the rate of flat.
- The output is the price of the flat. The applications where output is numeric are regression problems.
- Let X represents flat features and Y is the price of flat. We can collect training data by surveying past purchased transactions and the Machine Learning algorithm fits a function to this data to learn Y as a function of X for the suitable values of W and W_0.

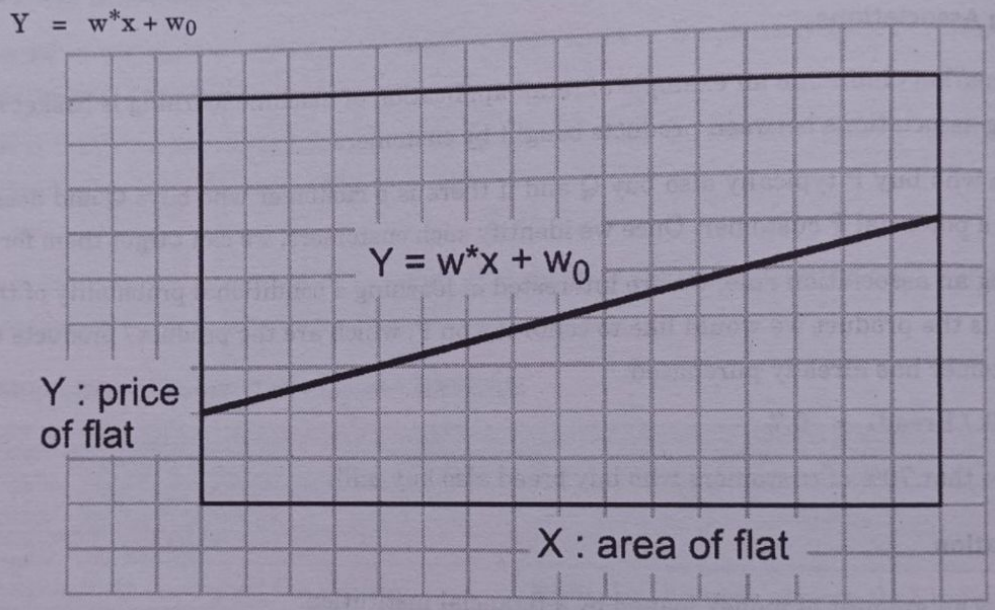


Fig. 5.5.2 : Regression for prediction of price of flat

(4) Unsupervised Learning

- One of the important unsupervised learning problem is clustering. In clustering dataset is partitioned in to meaningful sub classes known as clusters. For example, suppose you want to decorate your home using given items.
- Now you will classify them using unsupervised learning (no prior knowledge) and this classification can be on the basis of color of items, shape of items, material used for items, type of items or whatever way you would like.

(5) Reinforcement Learning

- There are some of the applications where output of system is a sequence of actions. In such applications the sequence of correct actions instead of single action is important in order to reach goal.
- An action is said to be good if it is part of good policy. Machine learning program generates a policy by learning previous good action sequences. Such methods are called reinforcement methods.

- A good example of reinforcement learning is chess playing. In artificial intelligence and machine learning, one of the most important research area is game playing.
- Games can be are easily described but at the same time, they are quite difficult to play well.
- Let's take a example of chess that has limited number of rules, but the game is very difficult because for each state there can be large number of possible moves.
- Another application of reinforcement learning is robot navigation. The robot can move in all possible directions at any point of time.
- The algorithm should reach goal state from an initial state by learning the correct sequence of actions after conducting number of trial runs.
- When the system has unreliable and partial sensory information, it makes reinforcement learning complex. Let's take an example of robot with incomplete camera information. Here robot does not know its exact location.

5.6 GRAPH BASED CLUSTERING

GQ. Explain graph based clustering and its algorithm.

- Graph clustering is an important subject, and deals with clustering with graphs.
- The data of a clustering problem can be represented as a graph where each element to be clustered is represented as a node and the distance between two elements is modeled by a certain weight on the edge linking the nodes.
- Thus in graph clustering, elements within a cluster are connected to each other but have no connection to elements outside that cluster.

5.6.1 Graph Clustering Algorithm

- The HCS (Highly Connected Subgraphs) clustering algorithm (also known as the HCS algorithm, and other names such as highly connected clusters /components /Kernels) is an algorithm based on graph connectivity for cluster analysis.
- It works by representing the similarity data in a **similarity graph**, and then finding all the highly connected subgraphs.
- It does not make any prior assumptions on the number of clusters.
- The HCS algorithm gives a clustering solution, which is inherently meaningful in the application domain.

5.6.2 Method of Graph-based Clustering

(1) **Transform the data into a graph representation :**

(i) Vertices are the data points to be clustered

(ii) Edges are weighted and based on similarity between data points.

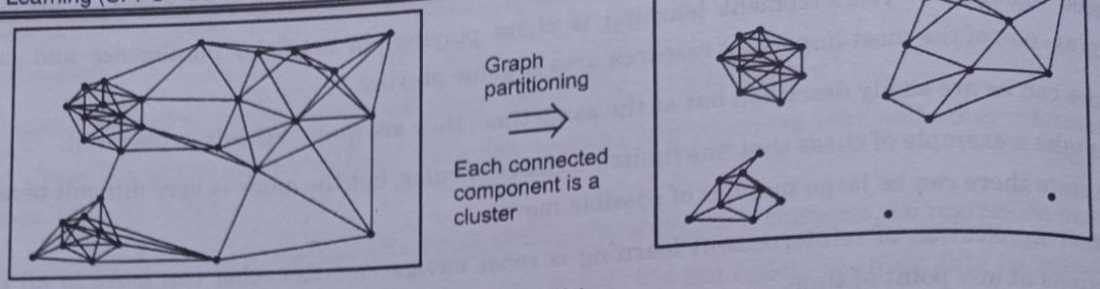


Fig. 5.6.1

(2) Clustering as graph partitioning

Two things are required :

(i) An objective function to determine what would be the best way to 'cut' the edges of a graph.

(ii) An algorithm to find the optimal partition (optimal according to objective function).

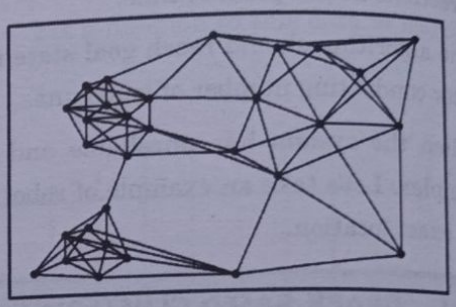


Fig. 5.6.2

(3) Objective function for partitioning

Suppose we want to partition the set of vertices V into two sets : V_1 and V_2. One possible objective function is to minimise graph cut :

Cut $(V_1, V_2) \sum_{i \in V_1, j \in V_2} W_{ij}$; W_{ij} is weight of the edge between nodes i and j

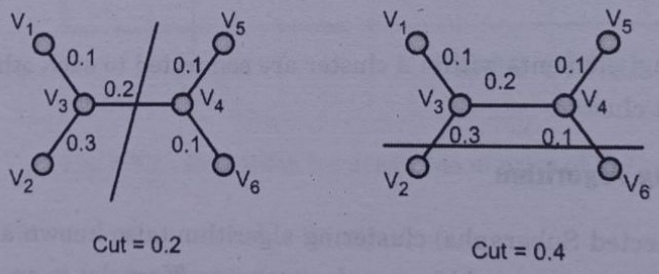


Fig. 5.6.3

(4) Objective function for partitioning limitation of minimising graph cut

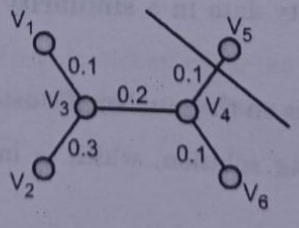


Fig. 5.6.4

The optimal solution might be to split up a single node from the rest of the graph!

Not a desirable solution,

(5) Objective function for partitioning

Our aim is not to only to have 'minimise the graph', but also look for "balanced" clusters.

$$\text{Ratio Cut } (V_1, V_2) = \frac{\text{Cut } (V_1, V_2)}{|\ V_1\ |} + \frac{\text{Cut } (V_1, V_2)}{|\ V_2\ |}$$

$$\text{Normalised Cut } (V_1, V_2) = \frac{\text{Cut } (V_1, V_2)}{\sum_{i \in V_1} d_i} + \frac{\text{Cut } (V_1, V_2)}{\sum_{j \in V_2} d_j}$$

Where $d_i = \sum_j W_{ij}$

V_1 and V_2 are the set of nodes in partitions 1 and 2;

$|\ V_1\ |$ is the number of nodes in partition V_1

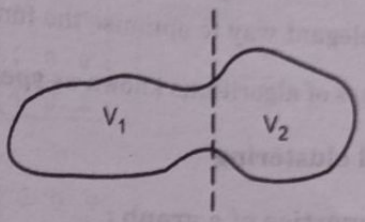
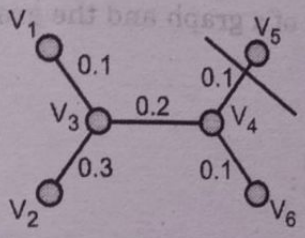


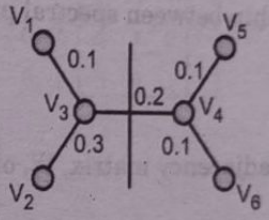
Fig. 5.6.5

(a) **Example**



Cut = 0.1

Fig. 5.6.6



Cut = 0.2

Fig. 5.6.7

Ratio cut $= \frac{0.1}{1} + \frac{0.1}{5} = 0.12$ **Ratio cut** $= \frac{0.2}{3} + \frac{0.2}{3} = 0.13$

Normalised cut

$= \frac{0.1}{0.1} + \frac{0.1}{1.5} = 1.07$

Normalised cut

$\frac{0.2}{1} + \frac{0.2}{0.6} = 0.53$

(b) **Example :** If graph is unweighted (or has to same edge weight).

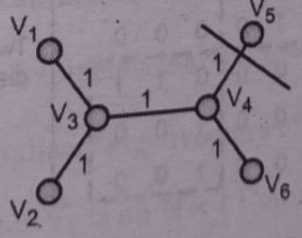


Fig. 5.6.8

Cut = 1

Ratio cut $= \frac{1}{1} + \frac{1}{5} = 1.2$

Normalised cut

$= \frac{1}{1} + \frac{1}{9} = 1.11$

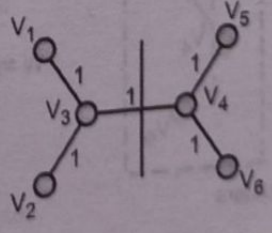


Fig. 5.6.9

Cut = 2

Ratio cut $= \frac{1}{3} + \frac{1}{3} = 0.67$

Normalised cut

$\frac{1}{5} + \frac{1}{5} = 0.2$

(6) Algorithm for graph partitioning

Method of minimising the objective function :

(i) We can use a heuristic (greedy) approach to do this.

(ii) An elegant way to optimise the function is by using ideas from spectral graph theory. This leads to a class of algorithms known as **spectral clustering**.

(7) Spectral clustering

Spectral properties of a graph :

(i) We find eigenvalues and eigenvectors of the adjacency matrix. They can be used to represent a graph.

(ii) There exists a relationship between spectral properties of a graph and the graph partitioning problem.

Method

(i) Start with a similarity/adjacency matrix, W, of a graph :

(ii) Define a diagonal matrix D

$$D_{ij} = \begin{cases} \sum_{k=1}^{n} W_{ik} & , \text{ if } i = j \\ 0 & , \text{ otherwise} \end{cases}$$

If W is a binary 0 – 1 matrix, then D_{ii} represents the degree of node i.

☞ **Preliminaries**

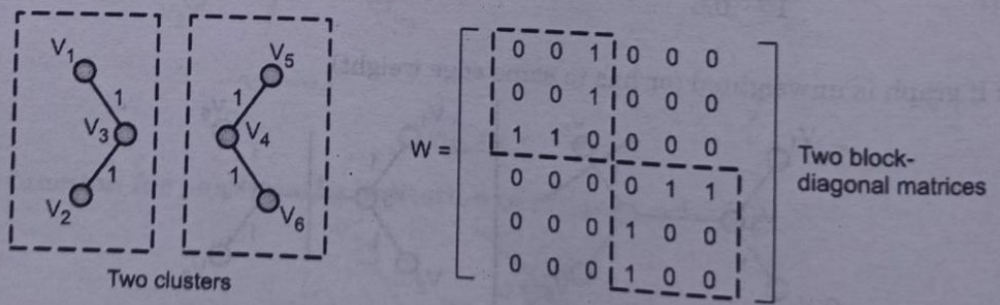


Fig. 5.6.10

(8) Graph Laplacian matrix

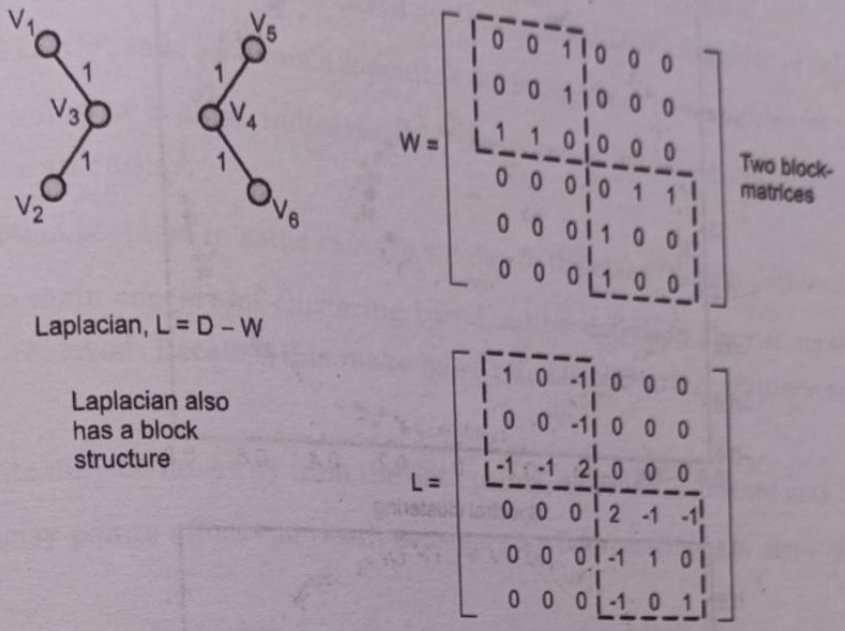


Fig. 5.6.11

(9) Properties of graph Laplacian

(i) L = (D – W) is a symmetric matrix

(ii) L is a positive semi-definite matrix

It means all eigenvalues are non-negative i.e. ≥ 0.

(10) Spectral clustering

Consider a data-set with N data points :

(i) Construct an N × N similarity matrix W.

(ii) Compute the N × N Laplacian matrix, L = D – W

(iii) Compute the K "Smallest" eigenvectors of L :

(a) Each eigenvector V_1 is an N × 1 Column vector.

(b) Create a matrix V containing eigen vectors $V_1, V_2, ..., V_k$ as column (one may exclude the first eigenvector)

(iv) Cluster the rows of V using K-means or other clustering algorithms into K-clusters.

☞ **Example**

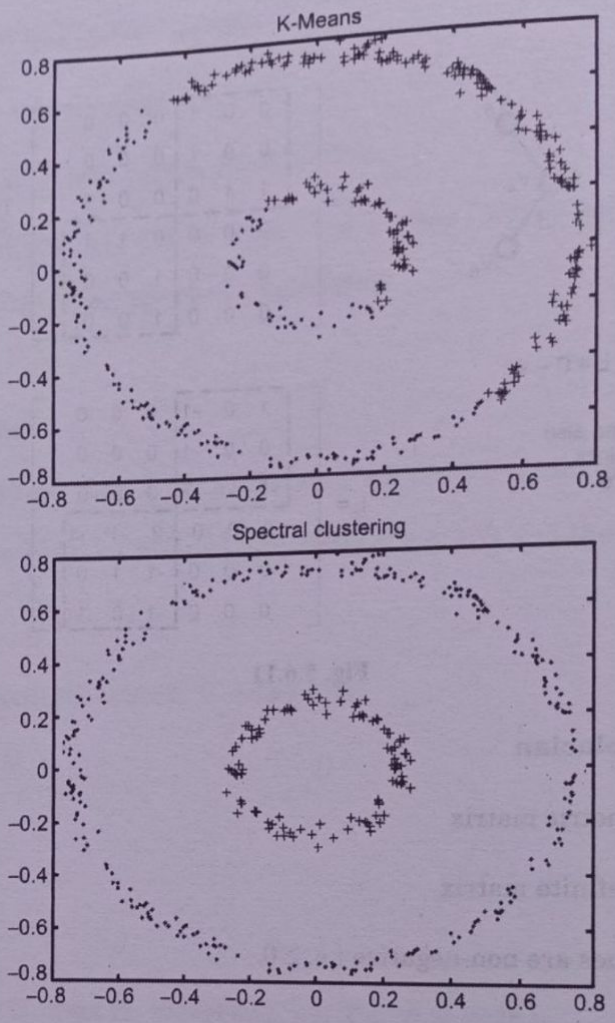


Fig. 5.6.12

☞ **Remark**

Spectral properties of a graph (i.e. eigen-values and eigenvectors) contain information about clustering structure.

5.7 OUTLIER ANALYSIS

GQ. What is outlier analysis.

- Outlier analysis is a fundamental issue in data mining, it is used to detect and remove anomalous objects from data-mining.
- The approach to detect outlier includes three methods. They are (i) clustering, (ii) pruning and (iii) computing outlier score.

(i) For clustering k-means algorithm is used. It partitions the dataset into given number of clusters.

(ii) In pruning points which are closed to centroid of each cluster and pruned. Pruning is based on distance measure.

(iii) For unpruned points, local distance based outlier factor (LDOF) measure is calculated.

- A measure called LDOF, tells how much a point is deviating from its neoighbours.
- The high LDOF value of a point indicates that the point is deviating more from its neighbours and probably it may be an outlier.
- The outlier detection problem in some cases is similar to the classification problem.
- For example, the main concern of clustering based outlier detection algorithms is to find clusters and outliers is to be removed. Because this make more reliable clustering. Outliers are generally regarded as noise.
- Some noisy points may be far away from the data points, whereas the others may be close.
- The far away noisy points affect the result considerably because they are more different from the data points.
- Hence it is desirable to remove the outliers, which are far away from all other points ion cluster.
- The identification of an outlier is affected by various factors, many of them are practical application.
- For example, fraud or criminal deception, will always be a costly problem for many profit organizations. Data mining can minimize some of these losses by making use of massive collections of customer data.
- Using web log files, it becomes possible to recognise fraudulent behaviour, changes in behaviour of customers or faults in systems.
- The typical fault detection can discover exceptions in the amount of money spent, type of item purchased, time and location.
- Another example is a computer security intrusion detection system. It finds outlier patterns as a possible intrusion attempts.
- Intrusion detection corresponds to a suite of techniques that are used to identify attacks against computers and network infrastructures.

5.8 ISOLATION FACATORS

GQ. What are isolation factors ?

- Any data point/observation that deviates significantly from other observations is called an Anomaly/outlier.
- Anomaly detection finds its application in various domains like network intrusion detection, sudden rise/drop in sales etc.

- Isolation factors (IF), similar to Random Forests (or Random Factors) are built on decision trees. It is an unsupervised model, since there are no pre-defined labels.
- In an isolation, randomly sub-sampled data is processed in a tree-structure based on randomly selected features.
- The samples which end up in shorter distances indicate anomalies, as it is easier to separate them from other observations.

How do Isolation factor / Forests work ?

The algorithm starts with the training of the data.

1. When given a dataset, a random subsample of the data is selected.
2. Branching starts by selecting a random feature (from the set of all N features) first. And then branching is done on a random threshold; (i.e. any value in the range of minimum and maximum values of the selected feature).
3. If the value of a data point is less than the selected threshold, it goes to the left branch otherwise to the right. And thus a node is split into left and right branches.
4. This process from step 2 is continued recursively till each data point is completely isolated.

5.8.1 Limitations of Isolation Factor

Isolations are computationally efficient and are very effective in Anomaly detection.

Despite its advantages, there are a few limitations as mentioned below :

1. The final anomaly score depends on the contamination parameter, provided while training the model, contamination has occurred.

   It indicates that we have an idea of what percentage of the data is anomalous before hand to get a better prediction.

2. Also the model suffers from a bias due to the way the branching takes place.

5.9 LOCAL OUTLIER FACTOR

GQ. Explain local outlier factor. Mention its advantages and disadvantages.

Local Outlier Factor (LOF) is an algorithm used for unsupervised outlier detection. It produces an anomaly score that represents data points which are outliers in the data set.

It does this by measuring the local density deviation of a given data point with respect to the data points near it.

Working of LOF

- Local density is determined by estimating distances between data points that are neighbours (k-nearest neighbours).
- So for each data point, local density can be calculated.
- By comparing this we can check which data points have similar densities and which have a lesser density than its neighbours.
- The ones with lesser densities are considered as outliers.
- Firstly, k-distances are distances between points that are calculated for each point to determine their k-nearest neighbours.
- The second closest point is said to be the second nearest neighbour to the point.
- The distance is used to calculate the reachability distance. It is defined as the maximum of the distance between two points and the k-distance of that point.
- Here is an image which represents reachability distance of a point to various neighbours.

Fig. 5.9.1

- For points inside the circle the k-distance is considered and for points outside the cluster, the distance between points is considered.
- The reachability distance to all of the k-nearest neighbours of a point are calculated to determine the Local Reachability Density (LRD) of that point.
- The local reachability density is calculated by taking the inverse of the sum of all the reachability distances of all the k-nearest neighbouring points.
- So, if the density of the neighbours and the points are almost equal, we say they are quite similar.
- If the density of the neighbours is lesser than the density of the point, the point is inlier, i.e. inside the cluster; and if the density is of the neighbours is move, then the point is outlier.

5.9.1 Advantages of Local Outlier Factor

(i) Sometimes it is difficult to determine outliers. A point that it is at a small distance from a very dence cluster might be considered as an outlier but a point that is at a farthest distance from a wider spread cluster may be considered as an inlier.

With LOR, outliers in local areas are determined, so this does not persist.

(iii) The method in LOF can be applied in many other fields to solve problems of detecting outliers like geographic data, video streams etc.

(iii) The LOF can be used to implement a different dissimilarity function as well. And it is found to outperform many other algorithms of anomaly detection.

5.9.2 Disadvantages of Local Outlier Factor

(i) It is not always the same LOF score that determines whether a point is an outlier or not. It might vary for different data sets.

(ii) In higher dimensions, the LOF algorithm detection accuracy gets effected.

(iii) As LOF score can be any number, it might be a little inconvenient to understand the distinguishing of inliers and outliers based on it.

5.10 EVALUATION METRICS AND SCORE

GQ. Explain different evaluation metrics and score.

To evaluate the models, we consider different kinds of merits. The choice of metric depends on the type of model and the implementation plan of the model.

The following metrics will tell us in evaluating the model's accuracy

Metrics contents

1. Confusion matrix	2. F1 Score
3. Gain and Lift charts	4. Kolmogorov Smirnov charts
5. AUC – RUC	6. Log – Loss
7. Gini coefficient	8. Concordant – Discordant ratio
9. Root mean squared Error	10. Cross – Validation

1. Confusion matrix

A confusion matrix is an $N \times N$ matrix, where N is the number of predicted classes.

We note that the following points

(i) **Accuracy :** The proportion of the total number of predictions that were correct.

(ii) **Positive predictive value or precision :** The proportion of positive cases that were correctly identified.

(iii) **Negative predictive value :** The proportion of negative cases that were correctly identified.

(iv) **Sensitivity or recall :** The proportion of actual positive cases which are correctly identifies.

(v) **Specificity :** The properties of actual negative cases which are correctly identified.

In general, we are concerned with one of the above defined metric. For example, in a pharmaceutical company, concern will be of minimal wrong positive diagnosis. Here it is a concern of high specificities.

2. **F1 Score**

F1 score is the harmonic mean of precision and recall values for a classification problem.

Formula is : $F1 = 2\left[\frac{(\text{Precison}) \cdot (\text{recall})}{(\text{Precison}) + (\text{recall})}\right]$

We have already discussed F1 score in detail.

3. **Gain and Lift charts**

Gain and lift chart are concerned to check the rank ordering of the probabilities. We mention the steps to build a Lift/gain chart.

- **Step 1 :** Calculate probability for each observations
- **Step 2 :** Rank these probabilities in decreasing order.
- **Step 3 :** Build deciles with each group having almost 10% of the observations.
- **Step 4 :** Calculate the response rate at each deciles for good (Responders), Bad (Non-responders) and total.

Lift/Gain charts are widely used in campaign targeting problem.

4. **Kolmogorov Smirnov Chart :**

K-S chart measures performance of classification models. K-S is a measure of the degree of separation between the positive and negative distribution.

5. **Area under the ROC curve (AUC-ROC).**

- The biggest advantage of using ROC-curve is that it is independent of change in proportion of responders.
- The ROC curve is the plot between sensitivity and (1-specificity). (1- specificity) is know as false positive rate and sensitivity is also known as True positive rate.
- For a model which gives class as output, will be represented as a single point in ROC plot.

6. **Log Loss**

- AUC – ROC considers only the order of probabilities and it does not take into account the model's capability to predict higher probabilities for samples more likely to be positive.

  So, we calculate log loss.
- Log loss is the negative average of the log of corrected predicted probabilities for each instance.

7. **Gini coefficient**

Gini is nothing but ratio between area between the ROC curve and the diagonal line and the area of the above triangle.

Gini = 2 (AUC) – 1

Gini above 60% is a good model

8. Concordant – Discordant Ratio

It is primarily used to access the model's predictive power. This metric is useful for any classification predictions.

9. Root mean squared error (RMSE)

It is used in regression problems. It assumes that errors are unbiased and follow a normal distribution.

RMSE metric is given by

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (\text{predicted}_i - \text{Actual}_i)^2}{N}}$$

Where N is total number of observation.

10. Root mean squared Logarithmic Error

Here, we take log of the predictions and actual values.

RMSLE is used when we don't want to penalise huge differences in the predicted and the actual valves when both predicted and true valves are huge numbers.

11. R-squared/Adjusted R-Squared

In RMSE metric, we do not have a benchmark to compare.

The formula for R-squared is :

$$R^2 = 1 - \frac{\text{MSE (model)}}{\text{MSE (base line)}}$$

12. Cross-Validation

It is one of the most important concepts in any type of data modelling. It says : Try to leave a sample on which you do not train the model and test the model on this sample before finalising the model.

5.11 ELBOW METHOD

- In cluster analysis, the **elbow method** is a **heuristic** used in determining the number of clusters in a data set.
- In this method, explained variation as a function of the number of clusters is plotted and picking the **elbow of the curve** as the number of clusters to use.

5.11.1 Intuition Works

- Using the "elbow" as a cutoff point is a common heuristic in mathematical optimisation. It chooses a point where diminishing returns are no longer worth the additional cost.

- The intuition is that increasing the number of clusters will improve the **fit** since there are more parameters to use.
- In practice, this may not be a sharp elbow and as a heuristic method, such an 'elbow' cannot be correctly identified.
- To overcome this the quantity called 'elbow strength' was introduced.

5.11.2 Measures of Variation

- There are various measures of "explained variation", used in the elbow method.
- Commonly, variation is quantified by **variance**, and the ratio used is the ratio of between-group variance to the total variance.
- One can also use the ratio of between-group variance to within-group -variance.

5.11.3 Elbow – Method Calculation

1. Compute clustering algorithm (e.g k-means clustering) for different values of k.
2. For each k, calculate the total within – cluster sum of square (WSS).
3. Plot the curve of WSS according to the number of clusters k.

5.11.4 Working of Elbow – Method

- Elbow method does not always work well, if data is not very much clustered. Dataset need not have a clear elbow.
- We can have a fairly smooth curve, and it is unclear what is the beast valve of k to choose.

5.12 EXTRINSIC AND INTRINSIC METHOD

GQ. Explain and compare extrinsic and intrinsic method.

Extrinsic motivation

Extrinsic motivation refers to the behaviour of individuals to perform tasks and learn new skills because of external **rewards or avoidance of punishment**.

Examples of extrinsic motivation could include :

(i) Reading a book to prepare for a test

(ii) Exercising to lose weight

(iii) Cleaning your home to prepare for visitors coming over.

5.12.1 Intrinsic Motivation

When one is intrinsically motivated, the behaviour is motivated by the internal desire to do something for its own sake for example, one's personal enjoyment of an activity, or one's desire to learn a skill because one is eager to learn.

Examples of intrinsic motivation include :

(i) Reading a book because you enjoy the storytelling.

(ii) Exercising because you want to relieve stress.

(iii) Cleaning your study- room because it helps you feel will-organized

5.12.2 The difference between Intrinsic and Extrinsic Motivation

Intrinsic motivation comes from within, while extrinsic motivation arises from external factors.

When one is intrinsically motivated, one engages in an activity because one enjoys it and gets personal satisfaction from doing it.

Table 5.12.1

Sr. No.	Intrinsic	Extrinsic
1.	Participating in a sport because it is fun and you enjoy it.	Participating in a sport in order to win a reward or get physically fit.
2.	Learning a new language because you like experiencing new things	Learning a new language because your job requires it.
3.	Spending time with some one because you enjoy their company.	Spending time with some one because they can further your social standing .
4.	Cleaning because you enjoy a tidy space.	Cleaning to avoid making your partner angry.
5.	Playing cards because you enjoy the challenge	Playing cards to win money
6.	Exercising because you enjoy physically challenging your body	Exercising because you want to lose weight or fit into an adult.
7.	Volunteering because it makes you feel content and fulfilled	Volunteering in order to meet a school or work requirement.
8.	Going for a run because you find it relaxing or are trying to beat a personal record	Going for a run to increase your chances at winning a competition.
9.	Painting because it makes you feel calm and happy	Painting so you can sell your art to make money.
10.	Taking on more responsibility at work because you enjoy being challenged and feeling accomplishment	Taking on more responsibility at work in order to receive a raise or promotion.

5.12.3 Which is better : Extrinsic or Intrinsic Motivation

- Actually both motivations are effective. Most people agree with the idea that extrinsic rewards should be used less in order to minimise the **over justification effects**.
- It does not mean that extrinsic motivation always presents negative outcomes. In fact, it can be extremely beneficial in some situations, the situation where someone needs to complete a task that they find unpleasant.
- Excessive rewards may create problems, but when used appropriately, extrinsic motivating factors can be a useful tool.
- There are several factors that can help to promote intrinsic motivation.
- These factors are :

(i) **Curiosity :** Fostered curiosity pushed people to explore and learn for the sole pleasure of learning and mastering.

(ii) **Challenge :** Being challenged helps people to work at optimal levels continuously, while staying consistent in work towards meaningful goals.

(iii) **Recognition :** People have an innate desire to be appreciated, so when efforts are recognized and appreciated by others, satisfaction becomes a reward in and of itself.

(iv) **Cooperation :** Cooperating with others satisfies the need to belong. It also present the reward of satisfaction, because cooperation involves helping others and working towards a shared goal.

- While intrinsic motivation is often seen as ideal due to its sustainability and the inherent nature of its rewards , both extrinsic and intrinsic motivation are influential in driving behavior .
- In order to understand how these can be best utilized, it is important to understand their key differences and the optimal times to employ each method.

Chapter Ends ...

❑❑❑

UNIT VI

CHAPTER 6

Introduction to Neural Networks

Syllabus

Artificial Neural Networks: Single Layer Neural Network, Multilayer Perceptron, Back Propagation Learning, Functional Link Artificial Neural Network, and Radial Basis Function Network, Activation functions,

Introduction to Recurrent Neural Networks and Convolutional Neural Networks

6.1 INTRODUCTION

The human brain is a highly complex and amazing processor. The most basic element of the human brain is a specific type of cell, known as Neuron, which does not regenerate. The power of human brain (mind) comes from the number of Neurons and their multiple interconnections. A brain has the ability to **Learn** to build up its own rules through what we refer to as Experience.

A **Neural-Network** is a machine that is designed to model the way in which the brain performs a particular task. Clearly, our interest is confined to an important class of **neural networks** that perform useful computation through process of **learning.** To achieve good performance, Neural networks employ a massive interconnection of simple computing cells, referred to as "Neurons" or "Processing units".

6.2 'ARTIFICIAL NEURAL NETWORK' (ANN)

Definition : Artificial Neural Network thus, we can define an 'artificial neural network' (ANN) is a massively distributed processor made up of simple processing units which store experiential knowledge and make it available for use.

The important features of 'Artificial Neural-Network (ANN)' are :

(1) Knowledge is acquired from its environment through a learning process.

(2) Interneuron-connections store the acquired knowledge.

(3) 'Artificial Neural Networks' learn by examples. In biological processes, learning involves adjustments to the connections that exist between neurons, ANNS undergo a similar changes and do the job on computers accurately all the time.

6.2.1 Concept

(1) Neural networks are those information processing systems, which are constructed and implemented to **model the human brain**.

(2) The concept that is of primary importance for a Neural Network is the ability of the network to learn from its environment, and to **improve** its performance through learning.

6.2.2 Single-Layer Network

Let us consider a Single-Layer Network

Now, the separating line is given by,

$$b + x_1 w_1 + x_2 w_2 = 0$$

If $w_2 \neq 0$, then

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

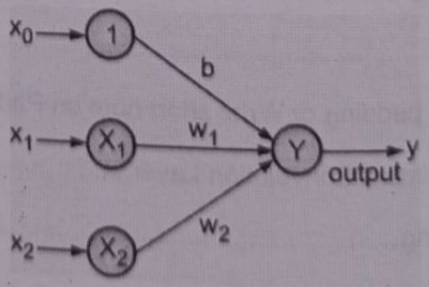


Fig. 6.2.1 : A single-layer neural-net

Thus the requirement for the positive response is : $b + x_1 w_1 + x_2 w_2 > 0$

Remarks

If the threshold value is being used, then the condition for the positive response from the output unit is

Net input received > θ (threshold value)

$$\therefore y_{in} > \theta \; ; \qquad \therefore x_1 w_1 + x_2 w_2 > \theta$$

Then the equation of the separating line will be

$$x_1 w_1 + x_2 w_2 = \theta \; ; \qquad \therefore x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}; \text{ (with } w_2 \neq 0)$$

6.3 SOLVED EXAMPLES ON ANN

Ex. 6.3.1 : For the network shown in the Fig. Ex. 6.3.1 calculate the net input to the output neuron.

Soln. :

The given neural net consists of three input neurons and one output neuron.

The inputs are : $x_1 = 0.3, \quad x_2 = 0.5, \quad x_3 = 0.6$

The weights are : $w_1 = 0.2, \quad w_2 = 0.1, \quad w_3 = -0.3$

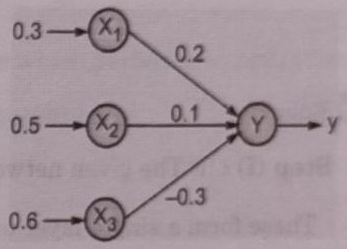


Fig. Ex. 6.3.1

The net-input is given by,

$$y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3 = (0.3)(0.2) + (0.5)(0.1) + (0.6)(-0.3)$$

$$\therefore y_{in} = 0.06 + 0.05 - 0.18 = -0.07$$

Ex. 6.3.2 : Obtain the output of the neuron Y for the net-work shown in the Fig. Ex. 6.3.2, using activation functions as : (i) binary sigmoidal and (ii) bipolar sigmoidal.

Soln. :

Step (I) :

(i) The given network has three input neurons with bias and one output neuron.

These form a single – layer network.

(ii) The inputs are : $x_1 = 0.8$; $x_2 = 0.6$; $x_3 = 0.4$

and weights are $w_1 = 0.1$; $w_2 = 0.3$; $w_3 = -0.2$

and bias is b = 0.35 (its input is always 1)

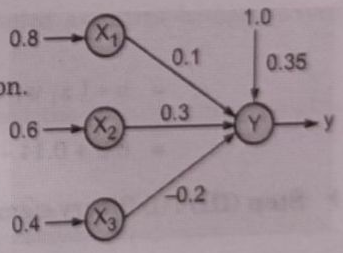


Fig. Ex. 6.3.2

Step (II) : Now the net input to the output neuron is $y_{in} = b + \sum_{i=1}^{3} x_i w_i$;

(∵ only 3 input neurons are given)

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3 = 0.35 + (0.8)(0.1) + (0.6)(0.3) + (0.4)(-0.2)$$

$$= 0.35 + 0.08 + 0.18 - 0.08 = 0.53$$

Step (III) : (i) Binary sigmoidal activation function given by,

$$y = f(y_{in}) = \left(\frac{1}{1 + e^{-y_{in}}}\right) = \frac{1}{1 + e^{-0.53}} = 0.625$$

(ii) bipolar sigmoidal activation functions is given by,

$$y = f(y_{in}) = \left(\frac{2}{1+e^{-y_{in}}}\right) - 1 = \left(\frac{2}{1+e^{-0.53}}\right) - 1 = 0.259$$

UEx. 6.3.3 (Ref. - Q. 1(d), June 14, 5 Marks)

Calculate the output of the neuron Y for the net given below. Use binary and bipolar sigmoidal activation functions :

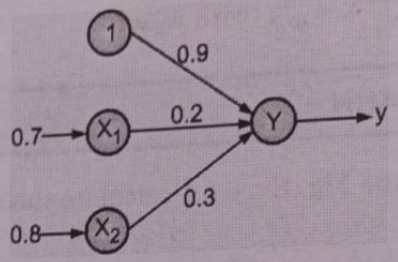


Fig. Ex. 6.3.3

☑ **Soln. :**

▶ **Step (I) :** (i) The given network has two input neurons with bias and one output neuron.

These form a single-layer network.

(ii) The inputs are : $x_1 = 0.7$; $x_2 = 0.8$

and weights are : $w_1 = 0.2$; $w_2 = 0.3$ and bias = $b = 0.9$ (its input is always 1)

▶ **Step (II) :** Now the net input to the output neuron is $y_{in} = b + \sum_{i=1}^{2} x_i w_i$;

(∵ only 2 input neurons are mentioned)

$$= b + (x_1 w_1 + x_2 w_2) = 0,9 + [(0.7)(0.2) + (0.8)(0.3)]$$

$$= 0.9 + 0.14 + 0.24 = 1.28 \quad \text{...(i)}$$

▶ **Step (III) :** (i) Binary sigmoidal function is given by

$$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}}$$

$$= \frac{1}{1+e^{-1.28}} = \frac{1}{1+0.278} = 0.7824$$

(ii) Bipolar sigmoidal activation function is,

$$y = f(y_{in}) = \left[\frac{2}{1+e^{-y_{in}}}\right] - 1$$

$$= \left[\frac{2}{1+e^{-1.28}}\right] - 1$$

$$= \mathbf{0.5649}$$

6.3.1 Single Layer Feed Forward Network

(1) This type of network comprises two layers, namely the **input layer** and the **output layer**.

(2) The **input layer neurons** receive' the input signals,

(3) The **output layer neurons** receive the output signals.

(4) The synaptic links carrying the weights connect every input neuron to the output neuron but not conversely.

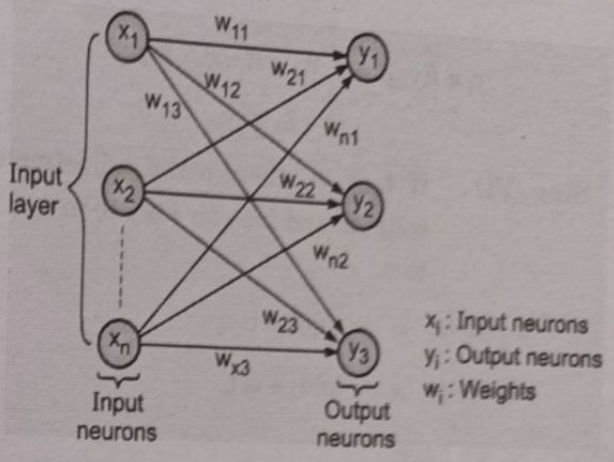


Fig. 6.3.1 : Single layer feed forward network

(5) This network is called as feed forward in type or acyclic in nature.

(6) The network is termed as single layer since the alone output layer, alone which performs computation.

(7) Even though there are two layers, it is called as single layer because of output layer.

(8) The input layer merely transmits the signals to the output layer.

We illustrate an example network as shown in Fig. 6.3.1.

6.4 MULTILAYER FEEDFORWARD NETWORK

This network is made up of multiple layers. The architecture of this class consists of one or more intermediary layers called as hidden layers, besides having an input and an output layer. The hidden layers perform intermediary computations before directing the input to the output layer. The input layer neurons are linked to the hidden layer neurons and the weights on these links are called as **input-hidden layer weights.**

Again, the hidden layer neurons are linked to the output layer neurons and the corresponding weights are referred to as **hidden-output layer weights.**

6.4.1 Perceptron Training Algorithm for Multiple output Classes

We mention below the Algorithm of perceptron training for multiple output classes :

- **Step (I) :** Initialise the weights, biases and learning rate conveniently.
- **Step (II) :** Check for stopping condition; if it is not true, perform steps 3-7.
- **Step (III) :** For each bipolar or binary training vector pair S : t, perform steps 4–6.
- **Step (IV) :** Set activation function (identity function) of each input unit i = 1 to n : xi = si
- **Step (V) :** Calculate the net input as, $yinj = bj + \sum_{i=1}^{n} xi\ wij$

Then calculate output response of each output unit j = 1 to m : To calculate the output response, apply activation formula over the net input :

$$y_j = f(y_{inj}) = \begin{cases} 1, & \text{if } y_{inj} > \theta \text{ ; } \theta \text{ is threshold valve} \\ 0, & \text{if } -\theta \le y_{inj} \le \theta \\ -1, & \text{if } y_{inj} < -\theta \end{cases}$$

- **Step (VI) :** If $t_j \neq y_j$, then make adjustment in weights and bias for j = 1 to m and i = 1 to n.

i.e. $w_{ij}\,(new) = w_{ij}(old) + \alpha\, t_j\, x_i$

$b_j\,(new) = b_j\,(old) + \alpha\, t_j$

Otherwise we have

$w_{ij}\,(new) = w_{ij}\,(old)$

$b_j\,(new) = b_j\,(old)$

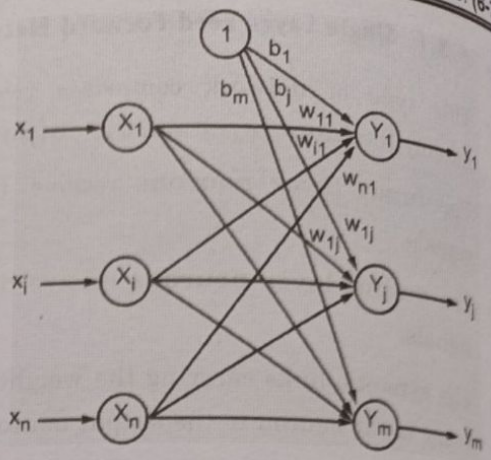


Fig. 6.4.1 : Network architecture for perceptron network for several output classes

- **Step (VII) :** If there is no change in weights then stop the training process, otherwise begin from step 3.

We exhibit the architecture for the above algorithm :

6.5 PROGRAM FOR PERCEPTRON TRAINING ALGORITHM

➢ **Program 6.5.1 :** Write a program for solving linearly separable problem using Perceptron Model.

☞ **Source code**

```
%Perceptron for AND function
clear;
clc;
x=[1 1 -1 -1;1 -1 1 -1];
t=[1 -1 -1 -1];
w=[0 0];
b=0;
alpha=input('Enter Learning rate=');
theta=input('Enter Threshold value=');
con=1;
epoch=0;
while con
  con=0;
    for i=1:4
      yin=b+x(1,i)*w(1)+x(2,i)*w(2);
        if yin>theta
        y=1;
        end
          if yin <=theta & yin>=-theta
```

```
            y=0;
            end
            if yin<-theta
            y=-1;
            end
            if y-t(i)
            con=1;
              for j=1:2
                w(j)=w(j)+alpha*t(i)*x(j,i);
              end
            b=b+alpha*t(i);
            end
      end
epoch=epoch+1;
end
disp('Perceptron for AND function');
disp(' Final Weight matrix');
disp(w);
disp('Final Bias');
disp(b);
save 'result.mat','w','b';
```

Output

```
Enter Learning rate= 0.5
Enter Threshold value=1
Perceptron for AND function
 Final Weight matrix
   1.5000    1.5000

Final Bias
  -1.5000
```

6.5.1 Perceptron Network Testing Algorithm

(I) For efficient performance of the network, it is to be trained with more data.

We mention the testing algorithm :

- **Step (I) :** The initial weights to be used are to be taken from the final weights obtained during training.
- **Step (II) :** Perform steps 3-4 for each input vector X.
- **Step (III) :** Set activations formula of the input unit.

- **Step (IV) :**Obtain the response of output unit : $y_{in} = \sum_{i=1}^{n} x_i w_i$

and $y=f(y_{in}) = \begin{cases} 1 & \text{if } y_{inj} > \theta \quad \theta \text{ is threshold} \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{inj} < -\theta \end{cases}$

With these steps the performance of the algorithm can be tested.

➢ **Program 6.5.2 :** Test the above trained perceptron network (Using MATLAB)

```
%This is the recall program for perceptron_AND
%Trained weight vector and bias value is saved in file 'result.mat' which is loaded here using load command
load result.mat;
x=input('Enter the test pattern');
yin=b+x(1)*w(1)+x(2)*w(2);
 if yin>theta
   y=1;
 end
 if yin <=theta & yin>=-theta
   y=0;
 end
 if yin<-theta
  y=-1;
 end
 disp('output is');
 y
```

☞ **Output**

```
Enter the test pattern
[-1 1]
output is
y =   -1
```

6.6 BACK-PROPAGATION NETWORK (BPN)

(i) Back-propagation network algorithm is applied to multilayer feed-forward networks consisting of processing elements.

(ii) The networks connected to back-propagation learning algorithm are also called as **back-propagation network (BPN).**

(iii) This algorithm provides a procedure for changing the weight in back-propagation network (BPN) to the given input pattern correctly.

(iv) The basic concept for this weight update is that where the error is propagated back to the hidden unit.

(v) In BPN, weights are calculated during the learning period of the network.

(vi) To update weights, the error must be calculated. There is no direct information of the error at the hidden layer. So we have to develop other techniques to calculate an error at the hidden layer, and this will cause minimisation of the output error.

(vii) Backpropagation is a systematic method of training multilayer artificial neural networks.

It is developed on high mathematical foundation and it has very good application potential.

(viii) Backpropagation learning rule is applicable on any feed forward network architecture.

(ix) Slow rate of convergence and local minima problems are its weaknesses.

(x) The multilayer feedforward (MLFF) network with back propagation (BP) learning is also called as 'multilayer perception' because of its similarity to perceptron networks with more than one layer.

(xi) **We carry out BPN as follows :**

(a) The feed forward of the input training pattern

(b) The back-propagation of the error and

(c) Updation of weights.

6.6.1 Architecture

A back-propagation neural network (BPN) consist of an input layer, hidden layer and an output layer. The neurons at the hidden and output layers have biases. The bias terms also act as weights. The outputs obtained could be either binary (0, 1) or bipolar (– 1, 1). Refer Fig. 6.6.1.

We Exhibit the Architecture of BPN

The terminologies used in the algorithm are as follows :

x = input training vector$(x_1, x_2, ..., x_n)$

t = target output vector

$(t_1, t_2, ... t_k, ..., t_m)$

α = learning rate parameter ;

v_{oj} = bias on j^{th} hidden unit,

w_{ok} = bias on k^{th} output unit ;

Z_j = hidden unit j.

x_j = input unit j, (since identity activation function is used for input layer, the input and output signals are same).

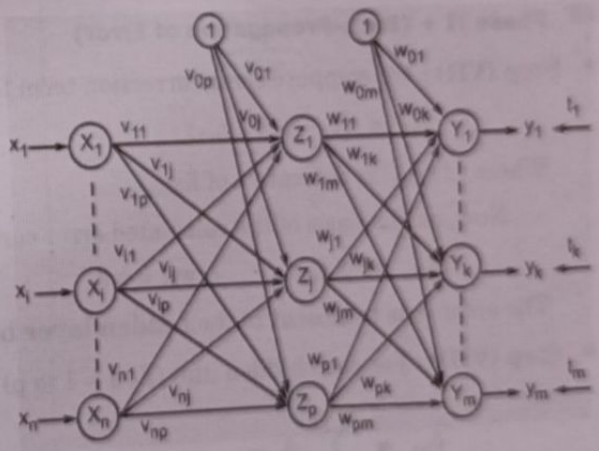


Fig. 6.6.1 : Architecture of a back-propagation network

The net input to Z_j is, $Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$ and the output is, $Z_j = f(Z_{inj})$

y_k = output unit k.

The net input to y_k is, $y_{ink} = w_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$ and the output is $y_k = f(y_{ink})$

δ_k = error correction weight adjustment for w_{jk}, which is back-propagated to the hidden units that feed into unit y_k. and

δ_j = error connection weight adjustment for v_{ij} and to the hidden unit Z_j.

6.6.2 Algorithm (Training)

We mention the error-back-propagation learning algorithm :

- **Step (I)** : Small random values for weights and learning rate,
- **Step (II)** : Carry on the steps 3-10 when stopping condition fails.
- **Step (III)** : Carry on steps 3-9 for each training pair.

Phase (I) : Feed-forward phase

- **Step (IV)** : Each input receives input signal xi and sends to hidden unit for i = 1 to n.
- **Step (V)** : Calculate net input

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

(To calculate output, we apply activation function Z_{inj})(binary or bipolar sigmoidal activation function): $Z_j = f(Z_{inj})$ and send the output signal to the input of output layer units.

- **Step (VI) :** For each output y_k (k = 1 to m), calculate the net input :

$$y_{ink} = w_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$$

and we apply activation function to evaluate output signal : $y_k = f(y_{ink})$

Phase II + (Back-Propagation of Error)

- **Step (VII) :** We compute error correction term for each output unit y_k (K = 1 to m)

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

Where $f'(y_{ink})$ is derivative of $f(y_{ink})$.

Now, on the basis of the calculated error correction term, update the change in weights and bias :

$$\Delta w_{jk} = \alpha \delta_k z_j \; ; \quad \Delta w_{ok} = \alpha \delta_k$$

The error δ_k is to be sent to the **hidden layer backwards.**

- **Step (VIII) :** For each hidden unit (Z_j, J = 1 to p) ;

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

and to calculate the error term : $\delta_j = \delta_{inj} \cdot f'(Z_{inj})$

Now, we update the change in weights and bias ;

$$\Delta V_{ij} = \alpha \delta_j x_i \; ; \quad \Delta V_{oj} = \alpha \delta_j$$

Phase (III) : Weight and Bias Updation

- **Step (IX) :** Each output unit (y_k, k = 1 to m) updates the bias and weights :

w_{jk} (new) = w_{jk} (old) + Δw_{jk} ; w_{ok} (new) = w_{ok} (old) + Δw_{ok}

Each hidden unit (Z_j, j = 1 to p) update its bias and weights :

V_{ij} (new) = V_{ij} (old) + ΔV_{ij} and v_{oj} (new) = V_{oj} (old) + ΔV_{oj}

- **Step (X) :** Check for stopping condition.

6.6.3 Flow-chart for Back-Propagation Network Training

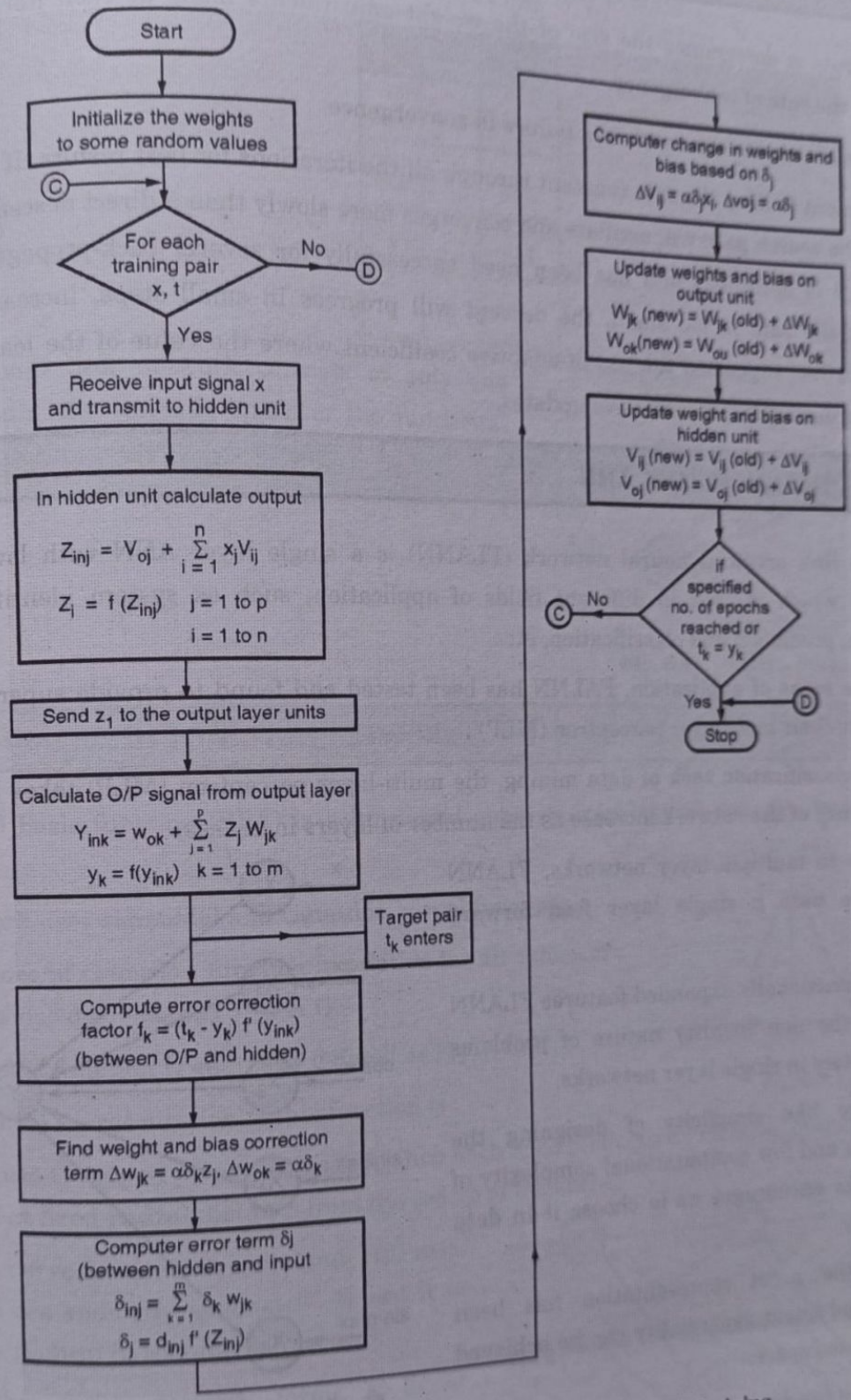


Fig. 6.6.2 : Flow-chart for Back-Propagation Network Training

6.7 LEARNING FACTORS OF BACK PROPAGATION NETWORK

(i) Learning rate α determines the size of the weight adjustments made at each iteration and hence influences the rate of convergence.

(ii) Poor choice of the rate can result in a failure in convergence.

It is convenient to keep the rate constant through all the iterations for best results. If the learning rate α is too large, the search path will oscillate and converges more slowly than a direct descent. The range of α from 10^{-3} to 0.9 is optimistic and has been used successfully for several back-propagation algorithmic experiments. If the rate is too small, the descent will progress in small steps, increasing the time to converge. Jacobs has suggested the use of adaptive coefficient where the value of the learning rate is the function of error derivative on successive updates.

6.8 FUNCTIONAL LINK ANN

- Functional link artificial neural network (FLANN) is a single layer ANN with low computational complexity which is used in different fields of application, such as system identification, pattern recognition, prediction and classification, etc.
- In all these areas of application, FALNN has been tested and found to provide superior performance compared to their multilayer perceptron (MLP) ;
- In solving classification task of data mining, the multi-layer perceptron (MLP) takes longer time and the complexity of the network increase as the number of layers increases.
- In contrast to multiple layer networks, FLANN architecture uses a single layer feed forward network.
- Using the functionally expanded features FLANN overcomes the non-linearity nature of problems and encounters in single layer networks.
- The feature like simplicity of designing the architecture and low computational complexity of the networks encourages us to choose it in data mining task.
- In short, the input representation has been enhanced and linear separability can be achieved in the extended space.

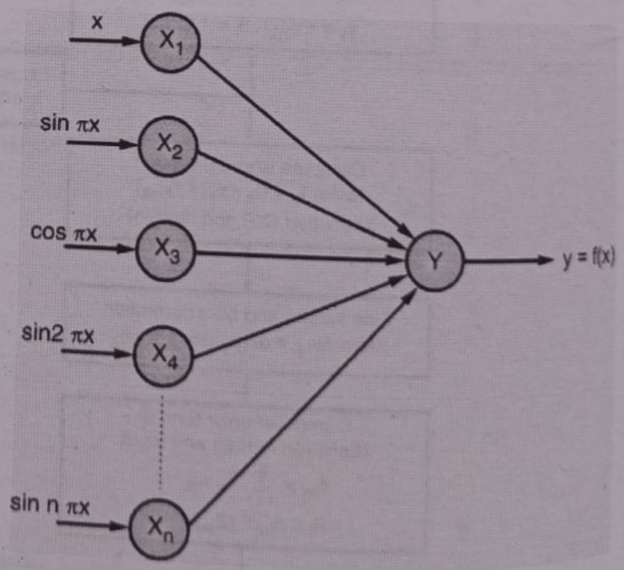


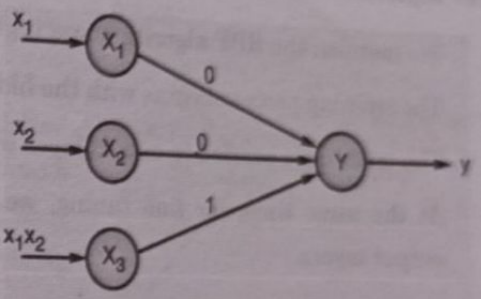
Fig. 6.8.1 : Functional link network

- We prepare a functional link model networks for learning continuous functions.
- Using (here) orthogonal basis functions such as sin πx, Cos πx, sin2 πx, Cos2 πx, etc., we can have higher order input terms.

- The most common example of linear non-separability is XOR – problem.
- The FLN can help in solving this problem. The input are

x_1	x_2	$x_1 x_2$
−1	−1	1
−1	1	−1
1	−1	−1
1	1	1

The functional link network consists of only one layer. As a result, the learning speed of the functional link is faster than that of BPN.



X OR - problem

Fig. 6.8.2 : XOR – Problem

6.9 RADIAL BASIS FUNCTION NETWORK

- The radial basis function (RBF) is a functional approximation neural network. It was developed by Powell.
- The network uses sigmoidal and Gaussion kernel functions.
- The response of Gaussion function is positive for all values of y and response decreases as $|y| \to O$.
- The Gaussion function is generally defined as $F(y) = e^{-y^2}$.

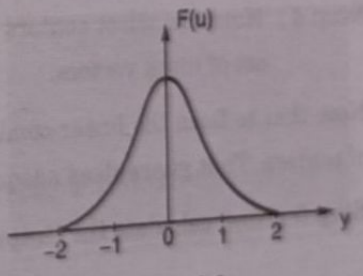


Fig. 6.9.1

- The graphical representation of this function is :
- When we use Gaussion potential function then each node produces an identical output for those inputs which are at fixed **radial** distance from the centre of the kernel.
- Since the curve is symmetrical about F(y) axis, the input are radically symmetric. Hence it is named as 'Radical basis function network'
- Hence, the entire network forms a linear combination of the **nonlinear basis function.**
- We draw architectural design of RBF.

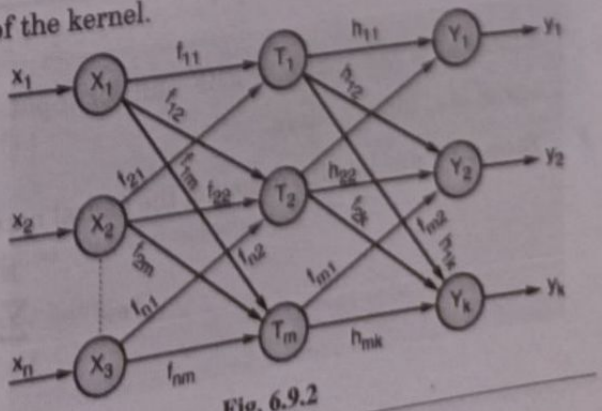


Fig. 6.9.2

- For convenience, we have chosen two layers to draw the architecture design.
- The output nodes form a **linear combination** of the basis function. This is computed by means of RBF nodes or hidden layer nodes.
- The important point is when the input falls within a small localized region of the input space formed by input stimulus in the hidden layer, it produces a significant non-zero response : This way linearity is secured.
- This network is also called as localized receptive field network.

RBF Algorithm

- We mention the RBF algorithm and it given all details of the calculations involved in the process.
- The training process begins with the hidden layer using unsupervised learning algorithm.
- The training is carried to output layer with a supervised learning algorithm.
- At the same time, for fine tuning, we apply supervised learning algorithm to both the hidden and output layers.

RBF Algorithm (Training)

- **Step 1 :** Weights of small random values to be set.
- **Step 2 :** Each input unit xi (x_i for all i = 1 to n) receives input signals and it transmits to the next hidden layer.
- **Step 3 :** To calculate the radial base function.
- **Step 4 :** Now, we select centers for the radial basis function. The centers are to be selected from the set of input vectors.

  Note that to form the linear combination of the non-linear basis, we have to choose a sufficient number of centers. That guarantees adequate sampling of the input vector space.
- **Step 5 :** Now, calculate the Output from the hidden layer unit using the formula.

$$f_j(x_i) = \frac{\exp\left[-\sum_{i=1}^{r} (x_{ji} - \bar{x}_{ji})^2\right]}{\sigma_i^2}$$

where $\bar{x}_{ji}$ is the centre of RBF unit for input variables ; σ_i the width of i^{th} RBF unit ; x_{ji} is the j^{th} variable of input pattern.

- **Step 6 :** Calculate the output of the neural network :

$$y_{net} = \sum_{i=1}^{k} h_{im} f_j(x_i) + h_o ;$$

k is the number of hidden layer nodes of RBF function. y_{net} is the output value of m^{th} node in the output layer ; w_o is the biasing term at the n^{th} output node.

Step 7 : We calculate the error ; if no error, then we stop the process ; other continue process by changing the weights.

6.10 ACTIVATION FUNCTION

UQ. Explain common activation functions used in neural network. **(Ref. - Q. 1(b), May 2011, 5 Marks)**

(1) A model of the behavior of a neuron can be presented as shown in Fig. 6.10.1. Here $x_1, x_2, ..., x_n$ are the n-inputs to the artificial neuron. $w_1, w_2, ..., w_n$ are the weights attached to the input links.

(2) **Biological** neuron receives all inputs through the dendrites, sums them and produces an output. If the sum is greater than a threshold value. The input signals are passed on to the cell body through the synapse which may accelerate or retard an arriving signal.

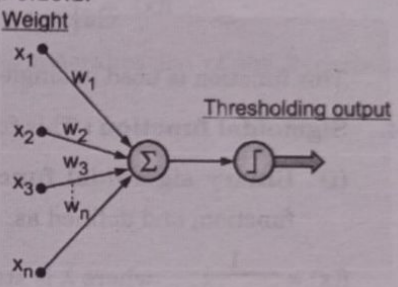


Fig. 6.10.1 : Simple model of an artificial neuron

(3) Here weights model the acceleration or retardation of the input signals. In short, weights are multiplicative factors of the inputs.

(4) The total input (say I) received by the soma (body) of the artificial neuron is

$$I = w_1 x_1 + w_2 x_2 + ... + w_n x_n \quad ...(i)$$

$$= \sum_{i=1}^{n} w_i x_i$$

(5) This sum is passed on to a non-linear filter ϕ called Activation function, or Transfer function, or Squash Function which releases the outputs.

$$\text{i.e. } y = f(I) \quad ...(ii)$$

6.10.1 Activation Functions

UQ. What are the important properties of activation function used in neural networks ? **(Ref. - Q. 1(c), May 2016, 5 Marks)**

UQ. List the different activation functions used in neural network. **(Ref. - Q. 1(d), May 2017, 5 Marks)**

UQ. With mathematical list four different activation functions used in neurons. **(Ref. - Q. 1(e), May 2019, 5 Marks)**

The obtain exact output, the activation function is applied over the net input of an ANN.

There are several activation functions. Here we consider a few :

1. **Linear function :** It is defined as

$$f(x) = x \quad \text{for all } x$$

Here input = output

2. **Bipolar Step function :** The function is defined as :

$$f(x) = \begin{cases} 1, & \text{if } x \geq \theta \\ -1, & \text{if } x < \theta \end{cases}$$

Where θ is threshold value. The function is also used in single-layer nets to convert the net input to an bipolar output (+ 1, – 1).

3. **Binary step function :** The function is defined as :

$$f(x) = \begin{cases} 1, & \text{if } x \geq \theta \\ 0, & \text{if } x < \theta \end{cases}$$

This function is used in single-layer nets to convert the net input to an output that is binary (0 or 1).

4. **Sigmoidal function :** This function is used in back-propagation nets. They are of two types :

(i) Binary sigmoidal function : It is also called as unipolar sigmoid function or a logistic sigmoid function, and defined as

$f(x) = \dfrac{1}{1+e^{-\lambda x}}$, where λ is steepness parameter

The derivative of f(x) is :

$$f'(x) = \frac{-1}{[1+e^{-\lambda x}]^2} \cdot (-\lambda e^{-\lambda x}) = \frac{\lambda (e^{-\lambda x},)}{[1+e^{-\lambda x}]^2} = \frac{\lambda [1+e^{-\lambda x} - 1]}{[1+e^{-\lambda x}]^2}$$

$$= \lambda \left\{ \frac{1}{(1+e^{-\lambda x})} - \frac{1}{(1+e^{-\lambda x})^2} \right\} = \lambda \left[\frac{1}{(1+e^{-\lambda x})} \left\{ 1 - \frac{1}{(1+e^{-\lambda x})} \right\} \right]$$

$$= \lambda [f(x)(1 - f(x))] = \lambda f(x)[1 - f(x)]$$

Range of this sigmoid function is 0 to 1 [$\because$ denominator is always greater than or equal to 1]

(ii) Bipolar sigmoid function :

The function is given by

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1 = \frac{2 - 1 - e^{-\lambda x}}{1+e^{-\lambda x}} = \frac{1 - e^{-\lambda x}}{1+e^{-\lambda x}}$$

where again, λ is steepress parameter and the range is between – 1 and + 1.

Derivative of f(x) is :

$$f'(x) = \frac{[1+e^{-\lambda x}][\lambda e^{-\lambda x}] - (1 - e^{-\lambda x})(-\lambda e^{-\lambda x})}{(1+e^{-\lambda x})^2}$$

$$= \frac{\lambda e^{-\lambda x}[1 + e^{-\lambda x} + 1 - e^{-\lambda x}]}{(1+e^{-\lambda x})^2} = \frac{2\lambda e^{-\lambda x}}{(1+e^{-\lambda x})^2} = \frac{\lambda}{2}\left[\frac{4 e^{-\lambda x}}{(1+e^{-\lambda x})^2} \right]$$

$$= \frac{\lambda}{2}\left[\frac{(1+e^{-\lambda x})^2 - (1-e^{-\lambda x})^2}{(1+e^{-\lambda x})^2} \right] = \frac{\lambda}{2}\left[1 - \left(\frac{1-e^{-\lambda x}}{1+e^{-\lambda x}} \right)^2 \right] = \frac{\lambda}{2}[1 - f(x)^2]$$

$$= \frac{\lambda}{2}[(1 + f(x))(1 - f(x))] \qquad [\because\ a^2 - b^2 = (a+b)(a-b)]$$

☞ **Remark**

(1) Here, $f(x) = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}} = \tanh\left(\frac{\lambda x}{2}\right)$

$$\therefore \quad f'(x) = \frac{\lambda}{2}\operatorname{sech}^2\left(\frac{x}{2}\right)$$

$$= \frac{\lambda}{2}\left[1-\tanh^2\left(\frac{x}{2}\right)\right] = \frac{\lambda}{2}\left[\left(1+\tanh\frac{x}{2}\right)\left(1-\tanh\frac{x}{2}\right)\right]$$

$$f'(x) = \frac{\lambda}{2}\left[(1+f(x))\,(1-f(x))\right]$$

(2) Sigmoid functions are so-called because their graphs are "S-shaped".

(i) Simple sigmoids defined to be odd, are monotone functions of one variable,

(ii) Hyperbolic sigmoids are subset of simple sigmoids and natural generalisation of the hyperbolic tangent function.

(3) **Ramp function**

$$\text{It is defined as } f(x) = \begin{cases} 1, & \text{if } x > 1 \\ x, & 0 \le x \le 1 \\ 0, & x < 0 \end{cases}$$

Graphs of activation functions

(1) Linear function

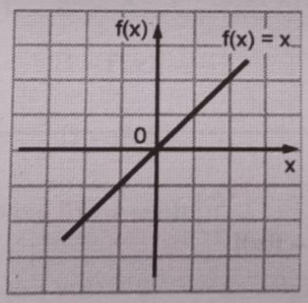


Fig. 6.10.2

(2)

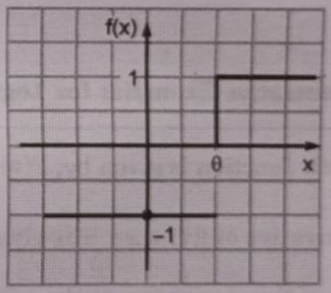


Fig. 6.10.3

(3) **Binary-step function**

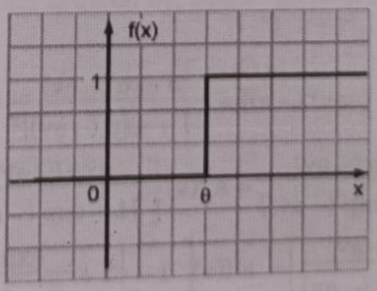


Fig. 6.10.4

(4)

(i) Binary sigmoidal function

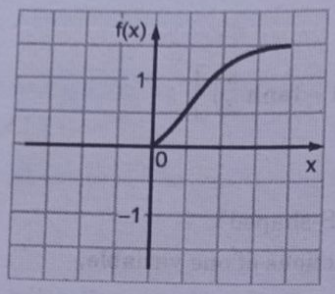


Fig. 6.10.5(i)

(ii) Bipolar sigmoidal function

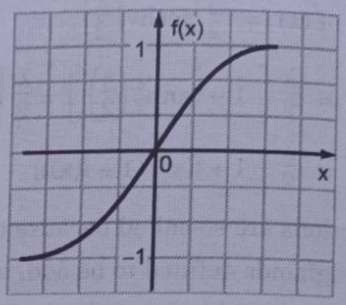


Fig. 6.10.5(ii)

(5) Ramp function

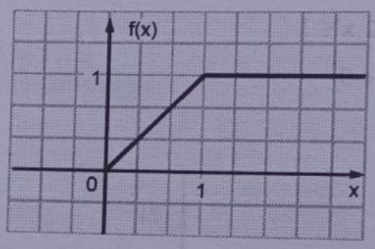


Fig. 6.10.6

6.10.2 Illustrative Examples for Logistic Function

Ex. 6.10.1 : Logistic function is given by, $F(v) = \frac{1}{1 + \exp(-av)}$

Show that the derivative of f(v) w.r.t. v is given by, $\frac{df}{dv} = af(v)\,[1 - f(v)]$

What is the value of this derivative at the origin ?

Soln. :

Step (I) : We have, $f(v) = \frac{1}{1 + \exp(-av)}$

Different w.r.t. v ; we get

$$f(v) = \frac{-1}{[1 + \exp(-av)]^2}\,[\exp(-av)(-a)] = \frac{a \exp(-av)}{[1 + \exp(-av)]^2}$$

$$= \frac{a}{[1 + \exp(-av)]}\,\frac{\exp(-av)}{[1 + \exp(-av)]} = a\,f(v)\left[\frac{\exp(-av)}{[1 + \exp(-av)]}\right]$$

$$= a\,f(v)\left[1 - 1 - \frac{\exp(-av)}{1 + \exp(-av)}\right] = a\,f(v)\left[1 - \left\{1 + \frac{\exp(-av)}{1 + \exp(-av)}\right\}\right]$$

$$= a\,f(v)\left[1 - \left\{\frac{1 + \exp(-av) - \exp(-av)}{1 + \exp(-av)}\right\}\right]$$

$$= a\,f(v)\left[1 - \left\{\frac{1}{1 + \exp(-av)}\right\}\right]$$

$\therefore f'(v) = a f(v)[1 - f(v)]$

Step (II) : As $V \to 0$; $\exp(-av) = \exp(0) = 1$

$$\therefore f(v) = \frac{1}{1+1} = \frac{1}{2}$$

$\therefore f'(v)$ at $v = 0$ is,

$$= a \cdot \frac{1}{2}\left[1 - \frac{1}{2}\right] = \frac{9}{4}$$

Ex. 6.10.2 : An odd sigmoid function is given by $f(v) = \frac{1 - \exp(-av)}{1 + \exp(-av)} = \tan h\left(\frac{av}{2}\right)$.

Show that the derivative of f(v) w.r.t. V is given by, $\frac{df}{dv} = \frac{a}{2}[1 - f^2(v)]$.

What is the valve of this derivative at the origin ? Suppose that the slope parameter a is made infinitely large. What is the resulting form of f(v).

☑ **Soln. :**

Step (I) : For convenience we take, $f(v) = \tan h\left(\frac{av}{2}\right)$

Differentiating w.r.t. V;

$$\frac{df}{dv} = \operatorname{sech}^2\left(\frac{av}{2}\right) \cdot \frac{a}{2} = \frac{a}{2}\left[1 - \tan h^2\left(\frac{av}{2}\right)\right]$$

$$\left[\because \operatorname{sech}^2 x = 1 - \tanh^2 x\right]$$

$$= \frac{a}{2}[1 - f^2(v)]$$

$$\therefore \frac{df}{dv} = \frac{a}{2}[1 - f^2(v)] \quad \ldots(i)$$

Step (II) : We have $f(v) = \tanh\left(\frac{av}{2}\right)$

At $v = 0$, $f(0) = \tan h(0) = 0$. $\therefore$ At origin, from Equation (i), $\frac{df}{dv} = \frac{a}{2}[1 - 0] = \frac{a}{2}$

As $a \to \infty$, $\tan h(\infty) = 1$. $\therefore$ As $a \to \infty$, $f(v) \to 1$.

Ex. 6.10.3 : The algebraic sigmoid function is given by, $f(v) = \frac{v}{\sqrt{1+v^2}}$

Show that the derivative of f(v) w.r.t. v is given by, $\frac{df}{dv} = \frac{\phi^3(v)}{v^3}$

What is the value of this derivative at origin ?

☑ **Soln. :**

Step (I) :

Differentiating $f(v) = \frac{v}{\sqrt{1+v^2}}$; we get

$$f'(v) = \left[\frac{\sqrt{1+v^2}\cdot 1 - v\cdot\frac{1\cdot 2v}{2\sqrt{1+v^2}}}{(1+v^2)}\right] = \left[\frac{\sqrt{1+v^2} - \frac{v^2}{\sqrt{1+v^2}}}{(1+v^2)}\right] = \left[\frac{(1+v^2)-v^2}{(1+v^2)^{3/2}}\right]$$

$$\therefore\ f'(v) = \frac{1}{(1+v^2)^{3/2}} \quad ...(i)$$

$$\therefore\ f'(v) = \frac{1}{v^3}\left[\frac{v^3}{(1+v^2)^{3/2}}\right] = \frac{1}{v^3}\left[\frac{v}{(1+v^2)}\right]^3$$

$$= \frac{1}{v^3}[f(v)]^3 = \frac{f^3(v)}{v^3} \quad ...(ii)$$

▶ **Step (II) :** As $v \to 0$ $f'(v) = \frac{1}{(1+v^2)^{3/2}} \to 1$.

$\therefore$ At origin, $f'(v) = 1$.

6.11 RECURRENT NEURAL NETWORK (RNN)

- Recurrent neural network is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence. This makes it to show temporal dynamic behaviour.
- It uses sequential data or time series data. These data are commonly used for ordinal or temporal problems, such as languages translation, natural language processing, speech recognition and image captioning, they are incorporated into popular applications such as voice search and Google translation.
- Like CNNs, recurrent neural networks utilise training data by learning. They are distinguished by their 'memory' as they take information from prior inputs to influence the current input and output.
- While traditional deep neural networks assume that inputs and outputs are independent of each other. The output of recurrent neural networks depends on the prior elements within the sequence, while future events would be also helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

6.12 CONVOLUTIONAL NETWORKS

GQ. Explain convolutional networks.

Convolutional network is a **special class** of **multilayer perceptrons,** designed to recognise two-dimensional shapes with a high degree of accuracy to skewing, scaling and to translation and other forms of distortion.

This task is achieved by a supervised manner which includes the following forms of constraints :

(1) Each neuron extracts local feature from the previous layer. Convolution layers apply a convolution operation to the input passing the result to the next layer. A convolution converts all the pixels in it's receptive field into a single value. The final output of the convolutional layer is a vector.

- Convolutional layers are the major building blocks used in convolution neural networks.

- In convolution layer, a neuron is only connected to a local area of input.
- A convolution layer contains units whose receptive fields cover a patch of receptive layer.
- The convolution layer is the core building block of a convolution network.

(2) Each computational layer is composed of feature maps in the form of a plane in which each individual neuron share the same synaptic weights. It has the following beneficial effects :

(i) Shift invariance through the use of **convolution** with a **kernel** of small size (and using sigmoid functions)

(ii) Reduction in the number of free parameters.

(3) **Subsampling :** Each convolutional layer performs local averaging and subsampling. It reduces the sensivity of the feature maps distortion.

- In convolutional network, all weights in all layers are learned through training. The network learns to extract it's own features automatically.
- In deep learning, a convolutional neural network (CNN) is an artificial neural network, applied to analyse visual imagery. They are also called as shift invariant or space invariant artificial neural networks (SIANN). Convolutional neural networks are equivariant and not invariant to translation.
- They have applications in image and video recognition, image segmentation, image classification, medical image analysis, image and video recognition, natural language processing brain-computer interfaces, and financial time series.
- CNNs are regularised versions of multilayer perceptronsy which are fully connected layers, i.e. each neuron in one layer is connected to all neurons in the next layer.

These days, deep learning is the threshold of many advanced technological applications, from self-driving cars to art and music. Deep learning enables the computer to build complex concepts from simpler concepts. Very soon we shall see that deep learning will be extended to applications that enable machines to thinks on their own.

6.13 CONVOLUTION NEURAL NETWORK (CNN) ARCHITECTURE

GQ. Explain CNN architecture.

- A **Convolution Neural Network** (CNN) is a feed-forward neural network (FNN). So far CNNs have established extraordinary performance in image search services, voice recognition and natural language processing (NLP).
- We have already seen that a regular multilayer perceptron gives good result for small images. But it breaks down for larger images.
- For example, if the first layer has 1000 neurons, then there will be 10 million connections.
- CNNs solve this problem using partially connected layers. CNN has fewer parameters than a deep neural network (DNN), hence it requires less training data.

In addition, CNN can detect any particular feature anywhere on the image, but a DNN can detect it only in that particular location.

- Since images have generally repetitive features, CNNs can generalise much better than DNNs for image processing tasks such as classification.
- A CNNs architecture has prior knowledge of how pixels are organised.
- Lower layers identify features in small areas of the images, while higher layers combine features of lower-layer into larger features. In case of DNNs, this doesn't work.
- Thus, in short, CNN is a class of neural networks that specialises in processing data that has a grid-like topology, such as an image.
- Each neuron works in it's own receptive field and is connected to other neurons in a way that they cover entire visual field.
- A CNN design begins with feature extraction and finishes with classification.

☞ **DNN Neural network**

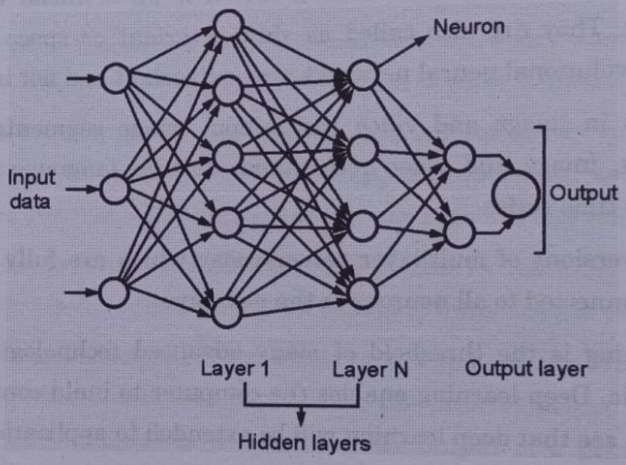


Fig. 6.13.1

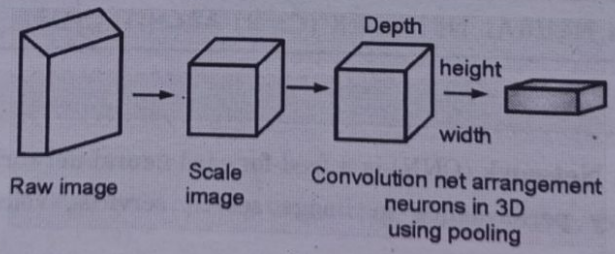


Fig. 6.13.2

In Fig. 6.13.1, there is DNN neural network. On the right, a convolution net arranges it's neurons in 3-dimensions, as seen in one of the layers.

6.14 CONVOLUTIONAL NETWORKS

GQ. Explain convolutional networks.

Convolutional network is a **special class** of **multilayer perceptrons,** designed to recognise two-dimensional shapes with a high degree of accuracy to skewing, scaling and to translation and other forms of distortion.

This task is achieved by a supervised manner which includes the following forms of constraints :

(1) Each neuron extracts local feature from the previous layer. Convolution layers apply a convolution operation to the input passing the result to the next layer. A convolution converts all the pixels in it's receptive field into a single value. The final output of the convolutional layer is a vector.

- Convolutional layers are the major building blocks used in convolution neural networks.
- In convolution layer, a neuron is only connected to a local area of input.
- A convolution layer contains units whose receptive fields cover a patch of receptive layer.
- The convolution layer is the core building block of a convolution network.

(2) Each computational layer is composed of feature maps in the form of a plane in which each individual neuron share the same synaptic weights. It has the following beneficial effects :

(i) Shift invariance through the use of **convolution** with a **kernel** of small size (and using sigmoid functions)

(ii) Reduction in the number of free parameters.

(3) **Subsampling :** Each convolutional layer performs local averaging and subsampling. It reduces the sensivity of the feature maps distortion.

- In convolutional network, all weights in all layers are learned through training. The network learns to extract it's own features automatically.
- In deep learning, a convolutional neural network (CNN) is an artificial neural network, applied to analyse visual imagery. They are also called as shift invariant or space invariant artificial neural networks (SIANN). Convolutional neural networks are equivariant and not invariant to translation.
- They have applications in image and video recognition, image segmentation, image classification, medical image analysis, image and video recognition, natural language processing brain-computer interfaces, and financial time series.
- CNNs are regularised versions of multilayer perceptronsy which are fully connected layers, i.e. each neuron in one layer is connected to all neurons in the next layer.

These days, deep learning is the threshold of many advanced technological applications, from self-driving cars to art and music. Deep learning enables the computer to build complex concepts from simpler concepts. Very soon we shall see that deep learning will be extended to applications that enable machines to thinks on their own.

6.15 CONVOLUTION NEURAL NETWORK (CNN) ARCHITECTURE

GQ. Explain CNN architecture.

- A **Convolution Neural Network** (CNN) is a feed-forward neural network (FNN). So far CNNs have established extraordinary performance in image search services, voice recognition and natural language processing (NLP).
- We have already seen that a regular multilayer perceptron gives good result for small images. But it breaks down for larger images.
- For example, if the first layer has 1000 neurons, then there will be 10 million connections.
- CNNs solve this problem using partially connected layers. CNN has fewer parameters than a deep neural network (DNN), hence it requires less training data.
- In addition, CNN can detect any particular feature anywhere on the image, but a DNN can detect it only in that particular location.
- Since images have generally repetitive features, CNNs can generalise much better than DNNs for image processing tasks such as classification.
- A CNNs architecture has prior knowledge of how pixels are organised.
- Lower layers identify features in small areas of the images, while higher layers combine features of lower-layer into larger features. In case of DNNs, this doesn't work.
- Thus, in short, CNN is a class of neural networks that specialises in processing data that has a grid-like topology, such as an image.
- Each neuron works in it's own receptive field and is connected to other neurons in a way that they cover entire visual field.

A CNN design begins with feature extraction and finishes with classification.

☞ **DNN Neural network**

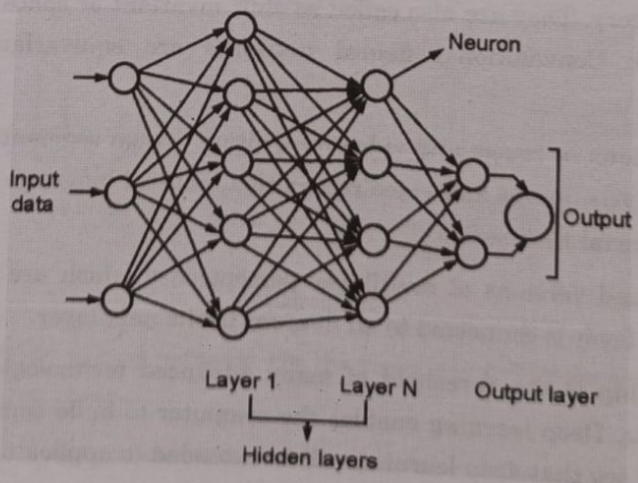


Fig. 6.15.1

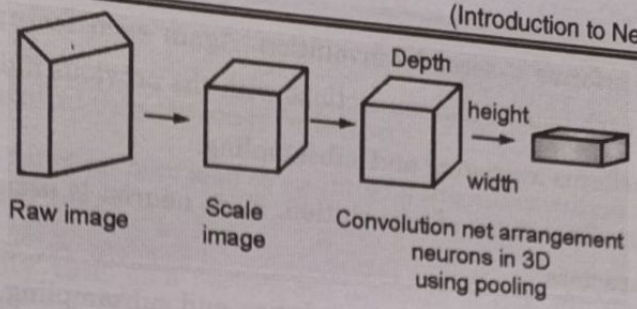


Fig. 6.15.2

In Fig. 6.15.1, there is DNN neural network. On the right, a convolution net arranges it's neurons in 3-dimensions, as seen in one of the layers.

6.15.1 Definition

Convolution is a mathematical operation. In 'convolution neural network' convolution is employed in the network.

6.15.2 Architecture

- A conventional neural network consists of an input layer, hidden layers and output layer. (as shown in the figure). The middle layers are called as hidden layers because their inputs and outputs are governed by activation function and convolution.
- In CNN, the input is a tensor with a shape : (Number of inputs) × (input height) × (input width) × (input channels).
- After passing through a convolutional layer, the image becomes a features map, also called an activation map, with shape :

  (number of inputs) × (feature map height) × (feature map width) × (feature map channels).
- The input is convolved in convolutional layers and the result is forwarded to the next layer. Each convolutional neuron processes data only for it's receptive field.
- Fully connected feed forward neural networks architecture is impractical for larger inputs. It requires a very high number of neurons. For example, a fully connected layer of size 100 × 100 has 10,000 weights for each neuron in the second layer. Instead using convolution a 5 × 5 filing regions, only 25 learnable parameters are required. Also convolutional neural networks are ideal for data with a a grid-like topology since separate features are taken into account during convolution.

6.15.3 Functions of Hidden Layers

GQ. Explain function of Hidden layers.

1. The first hidden layer performs convolution. Each feature map in the hidden layer consists of neurons and each neuron is assigned a receptive field.
2. The second hidden layer performs averaging and subsampling. Each layer consists of feature maps and the neurons of each feature map has a receptive field. It has a trainable bias, trainable coefficient and sigmoid function. They control the operating point of the neuron.

3. The next hidden layer performs a second convolution. Again each feature map in this hidden layer consists of neurons. And each neuron has connections with the previous hidden layer.
4. The next hidden layer performs averaging and subsampling.
5. The output layer performs final stage of convolution. Each neuron is assigned a receptive field and is assigned the possible characters.

The layers in the network alternate between convolution and subsampling, and we get a "bipyramidal" effect. Thus at each layer i.e. either subsampling or convolutional layer, the number of feature maps is increased while the space-resolution is reduced. The weight sharing reduces the number of free parameters in the network compared to the synaptic connections in multilayer perceptron. Also by the use of weight sharing, implementation of convolutional network in parallel form is possible.

6.15.4 Design of Multilayer Perceptron

GQ. Explain in detail Multilayer Perceptron.

- Using convolutional network, a multilayer perceptron of manageable size can learn a complex, high-dimensional, nonlinear mapping.
- Through the training set synaptic weights and bias levels can be learned by simple back propagation algorithm.
- Back propagation algorithm is the key-stone algorithm for the multilayer perceptrons. The partial derivatives of the cost function with respect to the free parameters of the network are determined by back-propagating of the error signals of the output neurons, hence the algorithm is called as Back-Propagation Algorithm.
- Updating the synaptic weights and biases of multilayer perceptron and deriving all partial derivatives of the cost function with respect to free parameters are the base factors for evaluating the power of the algorithm.

Details involved in the **design of a multilayer perceptron** are as follows :

(1) All the neurons in the network are **nonlinear**. And nonlinearity is achieved by using a sigmoid function, and they are
   (i) The nonsymmetric logistic function, and
   (ii) The antisymmetric hyperbolic tangent function.

(2) Each neuron has it's own hyperplane in decision space, and the combination of hyperplanes formed by all the neurons in the network is interatively adjusted by a supervised learning process. And this patterns from different classes are separated with the few classification errors.

(3) For the pattern classification, the random back-propagation algorithm is used to perform the training.

(4) In nonlinear regression, the output range of the multilayer perceptron should be sufficiently larger.

6.15.5 Performance Measure

(i) Algorithm is based on **minimising the cost function,**

(ii) But minimising the cost function may lead to optimising the intermediate quantity, and this may not be the aim of the system.

Hence 'reward-to-volatility' ratio as a performace measure of risk-adjusted return is more appreciable than Eav.

6.15.6 Input Layer

The input layer passes the data directly to the first hidden layer. Here the data is multiplied by the first hidden layers weights. Then the input layer passes the data through the activation function then it passes on.

6.15.7 Pooling Layers

GQ. Explain Pooling layers and its different types.

- Pooling layers are used to reduce the dimensions of the feature maps. It reduces the number of parameters to learn and the amount of computation performed in the network.
- The pooling layer summarises the features present in a region of the feature map generated by a convolutional layer.
- A pooling layers is another building block of a CNN, when processing multichannel input-data, the pooling layer pools each input channel separately.
- Pooling layer reduce the dimensions of the data by combining the output of neuron-clusters. The pooling layer is used to reduce spatial dimensions, but not depth, on a convolutional neural network.
- Pooling layer operates on each feature map independently.
- Pooling is basically 'downscaling' the image obtained from the previous layers. It can be compared to shrinking an image to reduce the pixel density.

6.15.8 Average Pooling

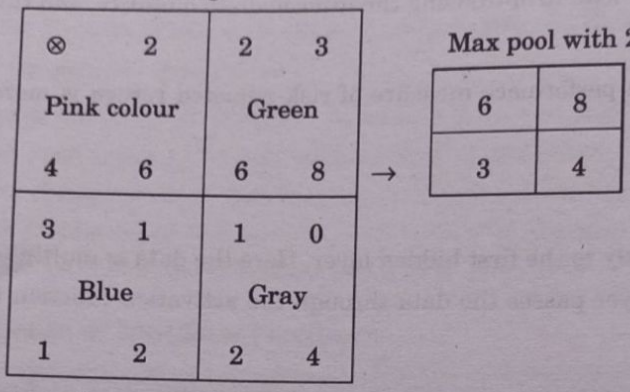
There are two types of widely used pooling in CNN layer.

1. Max pooling	2. Average Pooling

1. **Max-pooling**.

- The popular kind of pooling is **Max-pooling**. Suppose we want to pool by a ratio of 2. It implies that the height and width of your image will be half of it's original value.
- So we have to compress every 4 pixels (a 2×2 grid) and map it to a new single pixel with **loosing** the **important** data from the **missing** pixels.

- Max pooling is done by taking the largest value of these 4 pixels. Thus one new pixel represents 4 old pixels by using the largest value of these 4 pixels. This is repeated for every group of 4 pixels throughout the whole image.



Here the largest pixels are 6,8,3 and 4.

- Here the quality of an image is reduced to avoid computational load on the system. By reducing the quality of the image, we can increase the 'depth' of the layer, to have more features in the reduced image.
- Reducing the image size helps the convolution layer after the pooling layer to look for 'higher level features'. It means that the convolution layer looks at the picture as a whole.

2. Average pooling

- **Average pooling** is different from **Max Pooling**. Average pooling retains 'less important' information about the elements of a block, or pool.
- But Max pooling chooses the maximum value and discards less important values (as shown in the Fig. 6.15.3).

But 'less important' is also sometimes useful in a variety of situations.

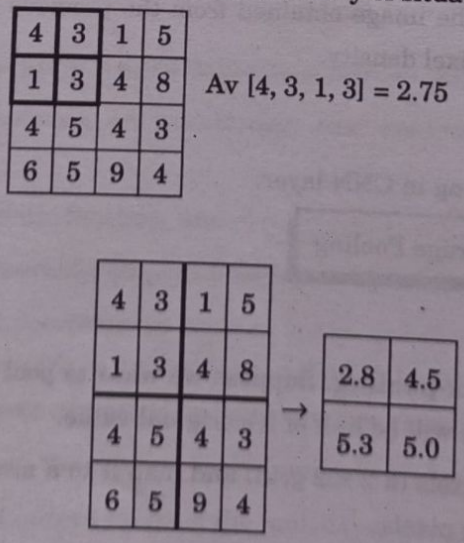


Fig. 6.15.3

6.15.9 Padding

GQ. Discuss padding or Write short note on Padding.

- In Convolutional Neural Networks (CNN), padding is a term that refers to the a number of pixels added to an image when it is being processed by the kernel of a CNN.
- For example, if padding in CNN is kept zero, then every pixel value that is added will be of value zero.
- Padding is simply a process of adding layers of zeros to our input images so as to avoid the problems mentioned above.
- CNNs are used extensively for tackling problems occurring in image processing and predictive modelling or classification tasks. The main application of CNN is to analyse image data.
- Now, every image in any dataset is a matrix of it's pixel values. When working with simple CNN for an image, we get output reduced in size and that is loss of data. And that hinders to obtain a proper result according to our requirements.
- When we do not want the shape or our outputs to reduce in size, the addition of more layers in the data can help to obtain a proper result and that addition can be done by padding.
- Now we study padding with it's importance and how to use it with CNN models. We also see different methods of padding and how they can be implemented.

6.15.10 Problem with Simple Convolution Layer

- A simple CNN of size $(n \times n)$ with $(f \times f)$ filter/kernel size gives result for output image as $(n - f + 1) \times (n - f + 1)$
- For example in any convolution operation with a (8×8) image and (3×3) filter the output image size will be (6×6). Thus the output of the layers is shrunk in comparison to the input. Again, the filters we are using may not focus on the corners every time when it moves on the pixels e.g.

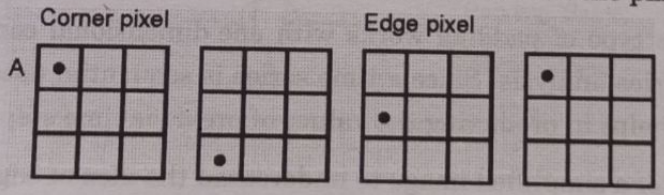


Fig. 6.15.4

- The above image is an example of the movement of a filter of size (3×3) on an image of size (6×6), corner pixel A is coming under the filter in only one movement. This shown that pixel A is misinterpreted.
- This causes loss of information available in the corners and also the output from the layers is reduced and this reduced information may create confusion for next layer. This problem of model can be solved by padding layer.

- The convolution layers reduce the size of the output. So when we want to save the information presented in the corners, we can use padding layers where padding helps by adding extra rows and columns on the outer dimension of the images. So the size of the **input data** will remain similar to the output data.
- Padding basically extends the area of an image in which convolutional neural network processes. The kernel/filter which moves across the image scans each pixels and converts the image into a smaller image.
- Padding is added to the outer frame of the image to allow for more space for the filter to cover in the image. This creates a more accurate analysis of images.

6.15.11 Types of Padding

GQ. What are types of Padding.

We come across three types of padding :

(1) Same padding
(2) Valid Padding
(3) Causal padding

(1) **Same padding :** In this type of padding, the padding layers have zero values in the outer frame of the images or data so the filter we are using can cover the edge of the matrix and it carries the required inference.

(2) **Valid padding :** This type of padding is called as no padding. Here we don't apply any padding, but the input gets, fully covered by the filter and so the every pixel of the image is valid.

Valid padding is to be used with Max-pooling layers. Here we use every pixel or point value while learning the model as valid pixel. It works on the validation of pixel and not on the size of the input.

(3) **Causal padding :** This type of padding works with one-dimensional convolution layers, we can use them majorly in time series analysis. Since a time series is sequential data, it helps in adding zeros at the start of data, and it helps in predicting the values of previous time steps.

We consider an example of a simplified image to understand the idea of edge detection.

Here, a 6×6 matrix convolved with a 3×3 matrix, output is 4×4 matrix. recall that if $n \times n$ image convolved with $f \times f$ kernel or filter then output image is of size.

$$(n - f + 1) \times (n - f + 1)$$

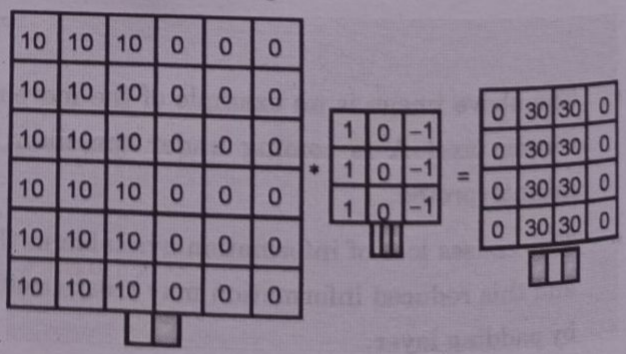


Fig. 6.15.5 : A 6×6 image convolved with 3×3 filter

As we have already seen, there are two problems with convolution :

1. After multiple convolution operation, our original image becomes small which we don't want to happen.
2. Corner features of any image are not used much in the output

Here padding preserves the size of the original image :

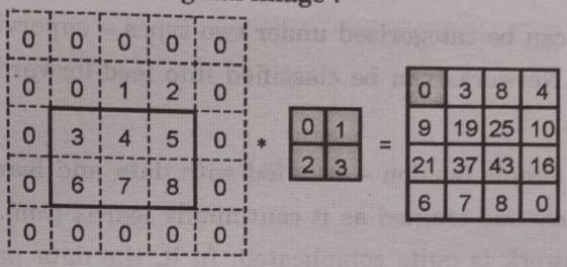


Fig. 6.15.6 : Padded image convolved with 2 * 2 kernel

Hence, if a $(n \times n)$ matrix convolved with an $(f \times f)$ matrix with padding P then the size of the output image becomes.

$(n + 2P - f + 1) \times (n + 2P - f + 1)$; here $P = 1$

Remark

Zero padding is introduced to make the shapes match as required, equally on every side of the input map.

Valid means no padding.

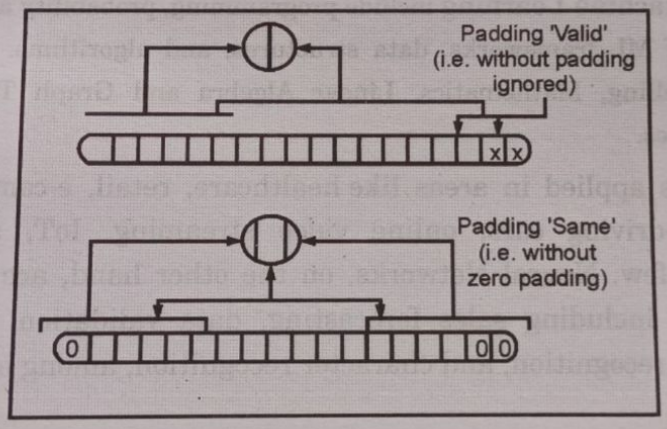


Fig. 6.15.7 : Same versus valid padding with CNN

6.16 MACHINE LEARNING VS NEURAL NETWORK

1. Machine Learning uses advanced algorithms that parse data, learns from it, and use those learnings to discover meaningful patterns of interest. Whereas a Neural Network consists of an assortment of algorithms used in Machine Learning for data modelling using graphs of neurons.
2. While a Machine Learning model makes decisions according to what it has learned from the data, a Neural Network arranges algorithms in a fashion that it can make accurate decisions by itself. Thus,

although Machine Learning models can learn from data, in the initial stages, they may require some human intervention.

Neural networks do not require human intervention as the nested layers within pass the data through hierarchies of various concepts, which eventually makes them capable of learning through their own errors.

3. Machine learning models can be categorized under two types – supervised and unsupervised learning models. However, Neural Networks can be classified into feed-forward, recurrent, convolutional, and modular Neural Networks.
4. An ML model works in a simple fashion – it is fed with data and learns from it. With time, the ML model becomes more mature and trained as it continually learns from the data. On the contrary, the structure of a Neural Network is quite complicated. In it, the data passes through several layers of interconnected nodes, wherein each node classifies the characteristics and information of the previous layer before passing the results on to other nodes in subsequent layers.
5. Since Machine Learning models are adaptive, they are continually evolving by learning through new sample data and experiences. Thus, the models can identify the patterns in the data. Here, data is the only input layer. However, even in a simple Neural Network model, there are multiple layers.

   The first layer is the input layer, followed by a hidden layer, and then finally an output layer. Each layer contains one or more neurons. By increasing the number of hidden layers within a Neural Network model, you can increase its computational and problem-solving abilities.
6. Skills required for Machine Learning include programming, probability and statistics, Big Data and Hadoop, knowledge of ML frameworks, data structures, and algorithms. Neural networks demand skills like data modelling, Mathematics, Linear Algebra and Graph Theory, programming, and probability and statistics.
7. Machine Learning is applied in areas like healthcare, retail, e-commerce (recommendation engines), BFSI, self-driving cars, online video streaming, IoT, and transportation and logistics, to name a few. Neural Networks, on the other hand, are used to solve numerous business challenges, including sales forecasting, data validation, customer research, risk management, speech recognition, and character recognition, among other things.

☞ **Conclusion**

These are some of the major differences between Machine Learning and Neural Networks. Neural Networks are essentially a part of Deep Learning, which in turn is a subset of Machine Learning. So, Neural Networks are nothing but a highly advanced application of Machine Learning that is now finding applications in many fields of interest.

Chapter Ends ...

□□□