CHAPTER 1

Introduction to Machine Learning

Learning Objectives

At the end of this chapter, you will be able to:

- Give a brief overview of machine learning (ML)
- Describe the learning paradigms used in ML
- Explain the important steps in ML, including data acquisition, feature engineering, model selection, model learning, model validation, model explanation, representation, and search and explanation

1.1 Evolution of Machine Learning

Machine learning (ML) is the process of learning a model that can be used in prediction based on data. Prediction involves assigning a data item to one of the classes or associating the data item with a number. The former activity is classification while the latter is regression. ML is an important and state-of-the-art topic. It gained prominence because of the improved processing speed and storage space of computers and the availability of large data sets for experimentation. Deep learning (DL) is an offshoot of ML. In fact, perceptron was the earliest popular ML tool and it forms the basic building block of various DL architectures, including multi-layer perceptron networks, convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

In the early days of artificial intelligence (AI), it was opined that mathematical logic was the ideal vehicle for building AI systems. Some of the initial contributions in this area like the General Problem Solver (GPS), Automatic Theorem Proving (ATP), rule-based systems and programming languages like Prolog and Lisp (lambda calculus–based) were all outcomes of this view. Various problem solving and game playing solutions also had this flavour. During the twentieth century, a majority of prominent AI researchers were of the view that *logic is AI and AI is logic.* Most of the reasoning systems were developed based on this view. Further, the role of artificial neural networks in solving complex real-world AI problems was under-appreciated.

However, this view was challenged in the early twenty-first century and the current view is that *AI is deep learning and deep learning is AI.* The advent of efficient graphics processing units (GPUs), platforms like TensorFlow and PyTorch along with the demonstrated success stories of convolutional neural networks, gated recurrent units and generative models based on neural networks have impacted every aspect of science and engineering activities across the globe.

So, ML along with DL has become a state-of-the-art subject. Artificial neural networks form the backbone of DL.

A high-level view of AI is shown in Fig. 1.1. The tasks related to conventional AI are listed separately. Here, ML may be viewed as dealing with more than just pattern recognition (PR) tasks. Classification and clustering are the typical tasks of a PR system. However, ML deals with regression problems also. Data mining is the efficient organization of data in the form of a database.

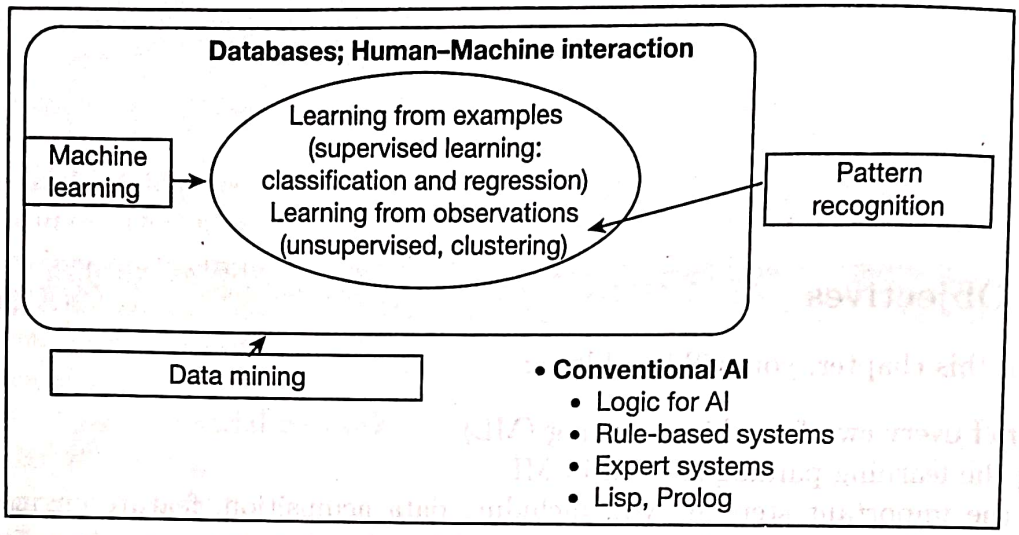


FIG. 1.1 A high-level view of AI

The typical background topics of AI are shown in Fig. 1.2.

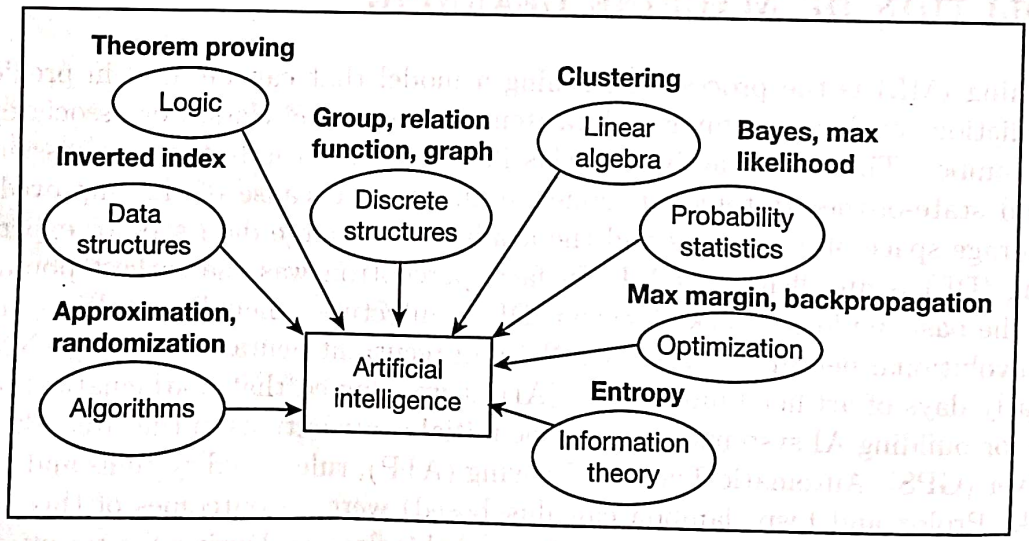


FIG. 1.2 Background topics of AI

Note that data structures and algorithms are basic to both conventional and current AI systems. Logic and discrete structures played an important role in the analysis and synthesis of conventional AI systems. The importance of other background topics may be summarized as follows:

- In ML, we deal with vectors and vector spaces and these topics are better appreciated through **linear algebra**. The data input to an ML system may be viewed as a matrix, popularly called the data matrix. If there are n data items, each represented as an l-dimensional vector, then the

corresponding data matrix A is of size $n \times l$. Linear algebra is useful in analysing the weights associated with the edges in a neural network. Matrix multiplication and eigen analysis are important in initializing the weights of the neural network and in weight updates. It can also help in weight normalization. The whole activity of clustering may be viewed as data matrix factorization.

- The role of **probability and statistics** need not be explained as ML is, in fact, statistical ML. These topics help in estimating the distributions underlying the data. Further, they play a crucial role in analysis and inference in ML.
- **Optimization** (along with calculus) is essential in training neural networks where gradients and their computations are important. Gradient descent–based optimization is an essential ingredient of any DL system.
- **Information theoretic concepts** like entropy, mutual information and Kullback–Leibler divergence are essential to understand topics such as decision tree classifiers, feature selection and deep neural networks.

We will provide details of all these background topics in their respective chapters.

1.2 Paradigms for ML

There are different ways or paradigms for ML, such as learning by rote, learning by deduction, learning by abduction, learning by induction and reinforcement learning. We shall look at each of these in detail.

1.2.1 Learning by Rote

This involves memorization in an effective manner. It is a form of learning that is popular in elementary schools where the alphabet and numbers are memorized. Memorizing simple addition and multiplication tables are also examples of rote learning. In the case of data caching, we store computed values so that we do not have to recompute them later. Caching is implemented by search engines and it may be viewed as another popular scheme of rote learning. When computation is more expensive than recall, this strategy can save a significant amount of time. Chess masters spend a lot of time memorizing the great games of the past. It is this rote learning that teaches them how to 'think' in chess. Various board positions and their potential to reach the winning configuration are exploited in games like chess and checkers.

1.2.2 Learning by Deduction

Deductive learning deals with the exploitation of deductions made earlier. This type of learning is based on reasoning that is truth preserving. Given A, and *if* A *then* B ($A \rightarrow B$), we can deduce B. We can use B along with *if* B *then* C ($B \rightarrow C$) to deduce C. Note that whenever A and $A \rightarrow B$ are $True$, then B is $True$, ratifying the truth preserving nature of learning by deduction. Consider the following statements:

1. It is raining.
2. If it rains, the roads get wet.
3. If a road is wet, it is slippery.

From (1) and (2), we can infer using deduction that (4) the roads are wet. This deduction can then be used with (3) to deduce or learn that (5) the roads are slippery. Here, if statements (1), (2) and (3) are *True*, then statements (4) and (5) are automatically *True*.

A digital computer is primarily a deductive engine and is ideally suited for this form of learning. Deductive learning is applied in well-defined domains like game playing, including in chess.

1.2.3 Learning by Abduction

Here, we infer A from B and $(A \rightarrow B)$. Notice that this is not truth preserving like in deduction as both B and $(A \rightarrow B)$ can be *True* and A can be *False*. Consider the following inference:

1. An aeroplane is a flying object (*aeroplane* $\rightarrow$ *flying object*).
2. A is a flying object.

From (1) and (2), we infer using abduction that A is an aeroplane. This kind of reasoning may lead to incorrect conclusions. For example, A could be a bird or a kite.

1.2.4 Learning by Induction

This is the most popular and effective form of ML. Here, learning is achieved with the help of examples or observations. It may be categorized as follows:

- **Learning from Examples:** Here, it is assumed that a collection of labelled examples are provided and the ML system uses these examples to make a prediction on a new data pattern. In supervised classification or learning from examples, we deal with two ML problems: classification and regression.
  1. **Classification:** Consider the handwritten digits shown in Fig. 1.3. Here, each row has 15 examples of each of the digits. The problem is to learn an ML model using such data to classify a new data pattern. This is also called **supervised learning** as the model is learnt with the help of such exemplar data. It may be provided by an expert in several practical situations. For example, a medical doctor may provide examples of normal patients and patients infected by COVID-19 based on some test results. In the case of handwritten digits, we have 10 class labels, one class label corresponding to each of the digits from 0 to 9. In classification, we would like to assign an appropriate class label from these labels to a new pattern.
  2. **Regression:** Contrary to classification, there are several prediction applications where the labels come from a possibly infinite set. For example, the share value of a stock could be a positive real number. The stock may have different values at a particular time and each of these values is a real number. This is a typical regression or curve fitting problem. The practical need here is to predict the share value of a stock at a future time instance based on past data in the form of examples.
- **Learning from Observations:** Observations are also instances like examples but they are different because observations need not be labelled. In this case, we cluster or group the observations into a smaller number of groups. Such grouping is performed with the help of a clustering algorithm that assigns similar patterns to the same group/cluster. Each cluster

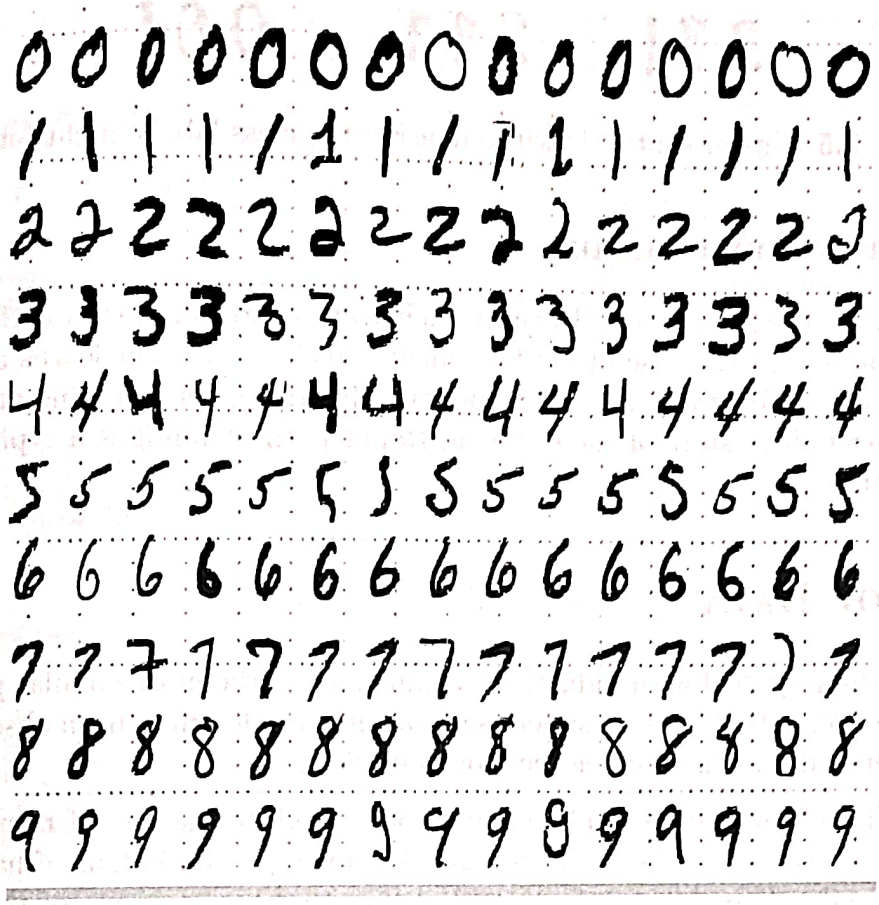


FIG. 1.3 Examples of handwritten digits labelled 0 to 9

could be represented by its centroid or mean. Let $x_1, x_2, \ldots, x_p$ be p elements of a cluster. Then the centroid of the cluster is defined by

$$\frac{1}{p}\sum_{i=1}^{p} x_i$$

Let us consider the handwritten digit data set of 3 classes: 0, 1 and 3. By using the class labels and clustering patterns of each class separately, we obtain 3 clusters that give us the 9 centroids shown in Fig. 1.4.

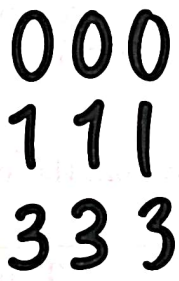


FIG. 1.4 Cluster centroids using the class labels in clustering

However, when we cluster the entire data of digits 0, 1 and 3 into 9 clusters, we obtain the centroids shown in Fig. 1.5. So, the clusters and their representatives could differ based on how we exploit the class labels.

331 311 001

FIG. 1.5 Cluster centroids without using the class labels in clustering

1.2.5 Reinforcement Learning

In supervised learning, the ML model is learnt in such a way as to maximize a performance measure like prediction accuracy. In the case of reinforcement learning, an agent learns an optimal policy to optimize some reward function. The learnt policy helps the agent in taking an action based on the current configuration or state of the problem. Robot path planning is a typical application of reinforcement learning.

1.3 TYPES OF DATA

In this book, we primarily deal with inductive learning as it is the most popular paradigm for ML. It is important to observe that in both supervised learning and learning from observations, we deal with data. In general, data can be categorical or numerical.

- **Categorical:** This type of data can be nominal or ordinal. In the case of **nominal data**, there is no order among the elements of the domain. For example, for colour of hair, the domain is {brown, black, red}. This data is of categorical type and the elements of the domain are not ordered. On the contrary, in **ordinal data**, there is an order among the values of the domain. For example, the domain of variable *employee number* could be $\{1, 2, \ldots, 1011\}$ if there are 1011 employees in an organization. Here, ordering among the elements of the domain is observed, indicating that senior employees have smaller employee numbers compared to junior employees; the most senior employee will have employee number 1.
- **Numerical:** In the case of numerical data, the domain of values of the data type could be a set/subset of integers or a set/subset of real numbers. For example, in Table 1.1, a subset of the features used by the Wisconsin Breast Cancer data is shown. The domain of *Diagnosis*, the class label, is a binary set with values *Malignant* and *Benign*. The domain of *ID Number* is a subset of integers in the range $[8670, 917897]$ and the domain of *Area_Mean* is a collection of floating point numbers (interval) in the range $[143.5, 2501]$. It is possible to have binary values in the domain for categorical or numerical data. For example, the domain of *Status* could be {Pass, Fail} and this variable is nominal; an example of a binary ordinal type is {short, tall} for humans based on their height. A very popular binary numerical type is {0, 1}; also in the

TABLE 1.1 Different types of data from the Wisconsin Breast Cancer database

Feature Number	Attribute	Type of Data	Domain
1	Diagnosis	Nominal	$\{Malignant, Benign\}$
2	ID Number	Ordinal	$[8670, 917897]$
3	Perimeter_Mean	Numerical	$[43.79, 188.5]$
4	Area_Mean	Numerical	$[143.5, 2501]$
5	Smoothness_Mean	Numerical	$[0.05263, 0.1634]$

classification context, the *class label* data can have the domain $\{-1, +1\}$ where -1 stands for the label of the negative class and $+1$ stands for the label of the positive class.

Typically, each pattern or data item is represented as a vector of feature values. For example, a data item corresponding to a patient with ID 92751 is represented by a five-dimensional vector (*Benign*, 92751, 47.92, 181, 0.05263), where each component of the vector represents the corresponding feature shown in Table 1.1. *Benign* is the value of feature 1, *Diagnosis*; similarly, the third entry 47.92 corresponds to feature 3, that is *Perimeter_Mean* and so on. Note that *Diagnosis* is a nominal feature and *ID Number* is an ordinal attribute. The remaining three features are numerical.

Here, *Diagnosis* or the class label is a dependent feature and the remaining four features are independent features. Given a collection of such data items or patterns in the form of five-dimensional vectors, the ML system learns an association or mapping between the independent features and the dependent feature.

1.4 MATCHING

Matching is an important activity in ML. It is used in both supervised learning and in learning from observations. Matching is carried out by using a **proximity measure** which can be a distance/dissimilarity measure or a similarity measure. Two data items, u and v, represented as l-dimensional vectors, match better when the distance between them is smaller or when the similarity between them is larger.

A popular distance measure is the **Euclidean distance** and a popular similarity measure is the **cosine of the angle between vectors.** The Euclidean distance is given by

$$d(u,v) = \sqrt{\sum_{i=1}^{l}(u(i) - v(i))^2}$$

The cosine similarity is given by

$$cos(u,v) = \frac{u^t v}{||u||||v||},$$

where $u^t v$ is the dot product between vectors u and v and $||u||$ is the Euclidean distance between u and the origin; it is also called the Euclidean norm.
Some of the important applications of matching in ML are in:

- **Finding the Nearest Neighbor of a Pattern:** Let x be an l-dimensional pattern vector. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be a collection of n data vectors. The nearest neighbor of x from $\mathcal{X}$, denoted by $NN_x(\mathcal{X})$, is x_j if

  $$d(x, x_j) \leq d(x, x_i),\ \forall x_i \in \mathcal{X}$$

  This is an approximate search where a pattern that best matches x is obtained. If there is a tie, that is, when both $x_p \in \mathcal{X}$ and $x_q \in \mathcal{X}$ are the nearest neighbors of x, we can break the tie arbitrarily or choose either of the two to be the nearest neighbor of x. This step is useful in classification and will be discussed in the next chapter.
- **Assigning to a Set with the Nearest Representative:** Let $C_1, C_2, \ldots, C_K$ be K sets with $x^1, x^2, \ldots, x^K$ as their respective representatives. A pattern x is assigned to C_i if

  $$d(x, x^i) \leq d(x, x^j),\ for\ j \in \{1, 2, \ldots, K\}$$

This idea is useful in clustering or learning from observations, where C_i is the i^{th} group or cluster of patterns or observations and x^i is the representative of C_i. The centroid of the data vectors in C_i is a popularly used representative, x^i, of C_i.

1.5 Stages in Machine Learning

Building a machine learning system involves a number of steps, as illustrated in Fig. 1.6. Note the emphasis on data in the form of training, validation and test data.

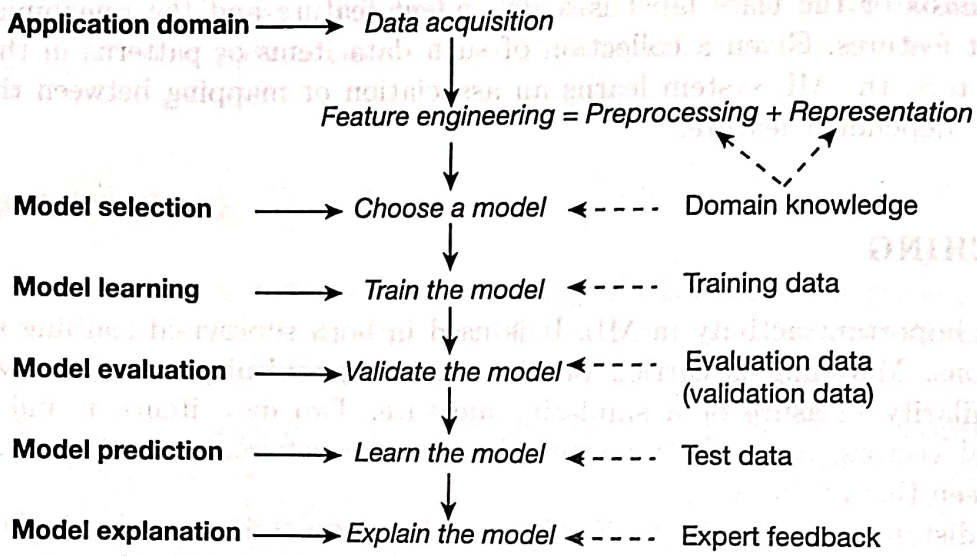


Fig. 1.6 Important steps in a practical machine learning system

Typically, the available data is split into training, validation and test data. Training data is used in model learning or training and validation data is used to tune the ML model. Test data is used to examine how the learnt model is performing. We now describe the components of an ML system.

1.5.1 Data Acquisition

This depends on the domain of the application. For example, to distinguish between *adults* and *children*, measurements of their *height* or *weight* are adequate; however, to distinguish between *normal* and *COVID-19-infected* humans, their *body temperature* and *chest congestion* may be more important than their *height* or *weight*. Typically, data collection is carried out before feature engineering.

1.5.2 Feature Engineering

This step involves a combination of data preprocessing and data representation.

Data Preprocessing

In several practical applications, the raw data available needs to be updated before it can be used by an ML model. The common problems encountered with raw data are missing values,

different ranges for different variables and the presence of outliers. We will now explain how to deal these problems.

1. **Missing Data:** It is likely that in some domains, there could be missing data. This occurs as a consequence of the inability to measure a feature value or due to unavailability or erroneous data entry. Some ML algorithms can work even when there are a reasonable number of missing data values and, in such cases, there is no need for preprocessing. However, there are a large number of other cases where the ML models cannot handle missing values. So, there is a need to examine techniques for dealing with missing data. Different schemes are used for dealing with the prediction of missing values:

- *Use the nearest neighbor:* Let x be an l-dimensional data vector that has its i^{th} component $x(i)$ missing. Let $\mathcal{X} = \{x^1, x^2, \ldots, x^n\}$ be the set of n training pattern vectors. Let $x^j \in \mathcal{X}$ be the nearest neighbor of x based on the remaining $l-1$ (excluding the i^{th}) components. Predict the value of $x(i)$ to be $x^j(i)$, that is, if the i^{th} component $x(i)$ of x is missing, use the i^{th} component of $x^j = NN_x(\mathcal{X})$ instead.
- *Use a larger neighborhood:* Use the k-nearest neighbors (KNNs) of x to predict the missing value x_i. Let the KNNs of x, using the remaining $l-1$ components, from $\mathcal{X}$ be $x1, x2, \ldots, xK$. Now the predicted value of $x(i)$ is the average of the i^{th} components of these KNNs. That is, the predicted value of $x(i)$ is

$$\frac{1}{K}\sum_{j=1}^{K} xj(i)$$

EXAMPLE 1: Consider the set data vectors

$$(1,1,1),(1,1,2),(1,1,3),(1,-,2),(1,1,-),(6,6,1)$$

There are 6 three-dimensional pattern vectors in the set. Missing values are indicated by $-$. Let us see how to predict the missing value in (1,–,2). Let us use $K = 3$ and find the 3 nearest neighbors (NNs) based on the remaining two feature values. The three NNs are $(1,1,1)$, $(1,1,2)$ and $(1,1,3)$. Note that the second feature value of all these three neighbors is 1, which is the predicted value for the missing value in(1,–,2). So, the vector becomes $(1,1,2)$.

- *Cluster the data and locate the nearest cluster:* This approach is based on clustering the training data and locating the cluster to which x belongs based on the remaining $l-1$ components. Let x with its i^{th} value missing belong to cluster C^q. Let μ^q be the centroid of C^q. Then the predicted value of $x(i)$ is μ_i^q, the i^{th} component of μ^q. We will explain clustering in detail in a later chapter; it is sufficient for now to note that a clustering algorithm can be used to group patterns in the training data into K clusters where patterns in each cluster are similar to each other and patterns belonging to different clusters are dissimilar.

EXAMPLE 2: Consider the following data matrix. It has 5 data vectors in a four-dimensional space.

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3.0 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & 3.1 & 1.5 & 0.2 \\ 5.0 & 3.6 & 1.4 & 0.2 \end{bmatrix}$$

Let us imagine that two values are missing to obtain

$$\begin{bmatrix} \text{NA} & 3.5 & 1.4 & 0.2 \\ 4.9 & 3.0 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & \text{NA} & 1.5 & 0.2 \\ 5.0 & 3.6 & 1.4 & 0.2 \end{bmatrix}$$

If we assume that this data is from the same cluster, we can compute the mean in each case and use it to predict the missing value. The mean of the available 4 values in the first column is 4.8; similarly the mean of the available 4 values in the second column is 3.325. So, using these mean values to replace the missing values in the matrix gives us the updated matrix

$$\begin{bmatrix} \mathbf{4.8} & 3.5 & 1.4 & 0.2 \\ 4.9 & 3.0 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & \mathbf{3.325} & 1.5 & 0.2 \\ 5.0 & 3.6 & 1.4 & 0.2 \end{bmatrix}$$

We can compute the **mean squared error (MSE)** with respect to the predicted values based on the deviations from the original values. The computation of MSE may be explained as follows: Given the n true (target) values to be $y_1, y_2, \ldots, y_n$ and the predicted values to be $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$, MSE is defined as

$$\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

In the above example, we have predicted two missing values based on the mean of the remaining values of the respective feature. In the first case, instead of 5.1, our estimated value is 4.8. Similarly, in the second case, for the value 3.1, our estimate is 3.325. So, the MSE here is

$$\frac{(5.1-4.8)^2 + (3.1-3.325)^2}{2} = \frac{0.09 + 0.00050625}{2} = 0.04525$$

EXAMPLE 3: Consider three clusters of points and their centroids:

a. Cluster1: {(1,1,1), (1,2,3), (1,3,2)} Centroid 1: (1,2,2)
b. Cluster2: {(3,4,3), (3,5,3), (3,3,3)} Centroid 2: (3,4,3)
c. Cluster3:{(6,6,6), (6,8,6), (6,7,6)} Centroid 3: (6,7,6)

Consider a pattern vector with a missing value given by (1, –, 2). Its nearest centroid among the three centroids based on the remaining two features is Centroid 1, (1,2,2). So, the missing value in the second location is predicted to be the second component in Centroid 1, that is, 2. So, the pattern with the missing value becomes (1,2,2).

We will now illustrate how the KNN-based and mean-based schemes work on a bigger data set. We consider 20,640 patterns of the California Housing data set. It has 8 features and the target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,000). This is a regression problem. Some values in the data set are removed to create missing values. The missing values are imputed using the KNN scheme and the mean-based scheme. Now the regressor (a function to predict the target) is used on the whole data set without missing values, on the KNN-based imputed data set

and the mean-based imputed data set. The resulting mean squared error of the predictions of the regressor is shown in Fig. 1.7.

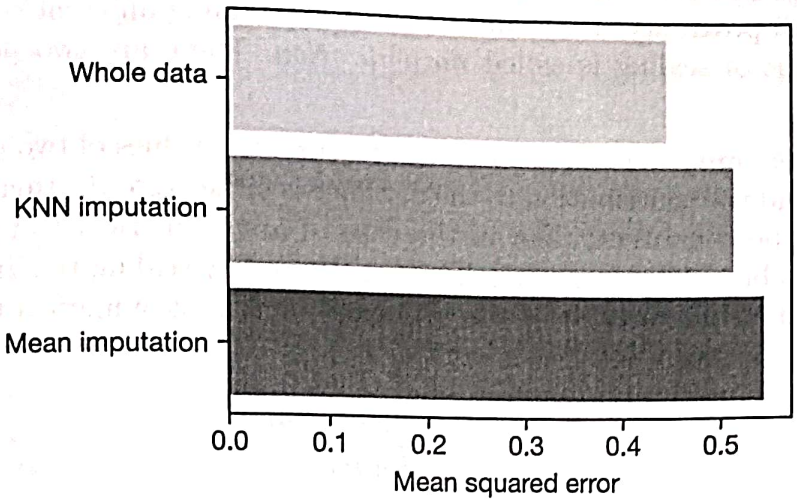


FIG. 1.7 MSE of the regressor on data imputed using KNN and mean

It is easy to observe that the regressor performs best when the whole data is available. However, when prediction is made by removing some values and guessing them, the performance of the regressor suffers; this is natural. Note that between the KNN-based and the mean-based imputations, the former made better predictions leading to smaller MSE. This is because the KNN-based scheme is more local to the respective point x and so is more focussed.

2. **Data from Different Domains:** The scales of values of different features could be very different. This would bias the matching process to depend more upon features that assume larger values, toning down the contributions of features with smaller values. So, in applications where different components of the vectors have different domain ranges, it is possible for some components to dominate in contributing to the distance between any pair of patterns. Consider for example, classification of objects into one of two classes: *adult* or *child*. Let the objects be represented by *height* in metres and *weight* in grams. Consider an adult represented by the vector $(1.6, 75000)$ and a child represented by the vector $(0.6, 5000)$, where the heights of the adult and the child in metres are 1.6 and 0.6, respectively, and the weights of the adult and the child in grams are 75000 and 5000, respectively. Assume that the domain of *height* is $[0.5, 2.5]$ and the domain of *weight* is $[2000, 200000]$ in this example. So, there is a large difference in the ranges of values of these two features.
Now the Euclidean distance between the adult and child vectors given above is

$$\sqrt{1.6 - 0.6)^2 + (75000 - 5000)^2} = \sqrt{1 + 4.9 \times 10^9} \approx 70000$$

Similarly, the cosine of the angle between the adult and child vectors is

$$\frac{0.96 + 375 \times 10^6}{\sqrt{25000000.36} \times \sqrt{5625000002.56}} \approx 1.0$$

Note that the proximity values computed between the two vectors, whether it is the Euclidean distance or the cosine of the angle between the two vectors, are dependent largely upon only

one of the two features, that is, *weight*, while the contributions of *height* are negligible. This is because of the difference in the magnitudes of the ranges of values of the two features. This example illustrates how the magnitudes/ranges of values of different features contribute differently to the overall proximity. This can be handled by scaling different components differently and such a process of scaling is called *normalization.* There are two popular normalization schemes:

- *Scaling using the range:* On any categorical feature, the values of two patterns either match or mismatch and the contribution to the distance is either zero (0) (match) or 1 (mismatch). If we want to be consistent, then in the case of any numerical feature also we want the contribution to be in the range [0,1]. This is achieved by scaling the difference by the range of the values of the feature. So, if the p^{th} component is of numerical type, its contribution to the distance between patterns x^i and x^j is

$$\frac{|x^i(p) - x^j(p)|}{Range_p},$$

where $Range_p$ is the range of the p^{th} feature values. Note that the value of this term is in the range $[0, 1]$; the value of 1 is achieved when $|x_p^i - x_p^j| = Range_p$ and it is 0 (zero) if patterns x^i and x^j have the same value for the p^{th} feature. Such a scaling will ensure that the contribution, to the distance, of either a categorical feature or a numerical feature will be in the range $[0, 1]$.
- *Standardization:* Here, the data is normalized so that it will have 0 (zero) mean and unit variance. This is motivated by the property of standard normal distribution, which is characterized by zero mean and unit variance.

EXAMPLE 4: Let there be 5 l-dimensional data vectors and let the the q^{th} components of the 5 vectors be 60, 80, 20, 100 and 40. The mean of this collection is

$$\frac{60 + 80 + 20 + 100 + 40}{5} = 60$$

We get zero-mean data by subtracting this mean from each of the 5 data items to obtain $0, 20, -40, 40, -20$ for their q^{th} components. Note that this is zero-mean data as these values add up to 0. To make the standard deviation of this data 1, we divide each of the zero-mean data values by the standard deviation of the data. Note that the variance of the zero-mean data is

$$\frac{0 + 20^2 + (-40)^2 + 40^2 + (-20)^2}{5} = 800$$

and the standard deviation is 28.284. So, the scaled data is $0, 0.707, -1.414, 1.414, -0.707$. Note that this data corresponding to the q^{th} feature value of the 5 vectors has zero mean and unit variance.

3. **Outliers in the Data:** An outlier is a data item that is either noisy or erroneous. Noisy measuring instruments or erroneous data recordings are responsible for the presence of such outliers. A common problem across various applications is the presence of outliers. A data item is usually called an outlier if it

- Assumes values that are far away from those of the average data items
- Deviates from the normally behaving data item
- Is not connected/similar to any other object in terms of its characteristics

Outliers can occur because of different reasons:

- *Noisy measurements:* The measuring instruments may malfunction and may lead to recording of noisy data. It is possible that the recorded value lies outside the domain of the data type.
- *Erroneous data entry:* Outlying data can occur at the data entry level itself. For example, it is very common for spelling mistakes to occur when names are entered. Also, it is possible to enter numbers such as salary erroneously as 2000000 instead of 200000 by typing an extra zero (0).
- *Evolving systems:* It is possible to encounter data items in sparse regions during the evolution of a system. For example, it is common to encounter isolated entities in the early days of a social network. Such isolated entities may or may not be outliers.
- *Very naturally:* Instead of viewing an outlier as a noisy or unwanted data item, it may be better to view it as something useful. For example, a novel idea or breakthrough in a scientific discipline, a highly paid sportsperson or an expensive car can all be natural and influential outliers.

An outlying data item can be either out-of-range or within-range. For example, consider an organization in which the salary could be from $\{10000, 150000, 225000, 300000\}$. In this case, an entry like 2250000 is an out-of-range outlier that occurs possibly because of an erroneous zero (0). Also, if there are only 500 people drawing 10000, 400 drawing 150000, 300 at 225000 and 175 drawing 300000, then an entry like 270000 could be a within-range outlier.

There are different schemes for detecting outliers. They are based on the density around points in the data. If a data point is located in a sparse region, it could be a possible outlier. It is possible to use clustering to locate such outliers. It does not matter whether it is within-range or out-of-range. If the clustering output has a singleton cluster, that is, a one-element cluster, then that element could be a possible outlier.

1.5.3 Data Representation

Representation is an important step in building ML models. This subsection introduces how data items are represented. It also discusses the importance of representation in ML. In the process, it deals with both feature selection and feature extraction and introduces different categories of dimensionality reduction.

It is often stated in DL literature that feature engineering is important in ML, but not in DL because DL systems have automatic representation learning capability. This is a highly debatable issue. However, it is possible that, in some application domains, DL systems can avoid the representation step explicitly. However, preprocessing including handling missing data and eliminating outliers is still an important part of any DL system. Even though representation is not explicit, it is implicitly handled in DL by choosing the appropriate number of layers and number of neurons in each layer of the neural network.

Representation of Data Items

The most active and state-of-the-art paradigm for ML is **statistical machine learning**. Here, each data item is represented as a vector. Typically, we consider addition of vectors in computing the mean or centroid of a collection of vectors, multiplication of a vector by a scalar in dealing with operations on matrices, and the dot product between a pair of vectors for computing similarity as important operations on the set of vectors. In most of the practical applications, the dimensionality

of the data or correspondingly size of the vectors representing data items, L, can be very large. For example, there are around 468 billion Google Ngrams. In this case, the dimensionality of the vectors is the vocabulary size or the number of Ngrams; so, the dimensionality could be very large. Such high-dimensional data is common in bioinformatics, information retrieval, satellite imagery, and so on. So, representation is an important component of any ML system. An arbitrary representation may also be adequate to build an ML model. However, the predictions made using such a model may not be meaningful.

Current-day applications deal with high-dimensional data. Some of the difficulties associated with ML using such high-dimensional data vectors are:

- Computation time increases with the dimensionality.
- Storage space requirement also increases with the dimensionality.
- *Performance of the model:* It is well-known that as the dimensionality increases, we require a larger training data set to build an ML model. There is a result, popularly called the **peaking phenomenon**, that shows that as the dimensionality keeps increasing, the accuracy of a classification model increases until some value, and beyond that value, the accuracy starts decreasing.

This may be attributed to the well-known concept of **overfitting**. The model will tend to remember the training data and fails to perform well on validation data. With a larger training data set, we can afford to use a higher dimensional data set and still avoid overfitting. Even though the dimensionality of the data set in an application is large, it is possible that the number of available training vectors is small. In such cases, a popular technique used in ML is to reduce the dimensionality so that the learnt model does not overfit the available data.

Well-known dimensionality reduction approaches are:

- **Feature selection:** Let $\mathcal{F} = \{f_1, f_2, \ldots, f_L\}$ be the set of L features. In the feature selection approach, we would like to select a subset F_l of $\mathcal{F}$ having $l(< L)$ features such that F_l maximizes the performance of the ML model.
- **Feature extraction:** Here, from the set $\mathcal{F}$ of L features, a set $\mathcal{H} = \{h_1, h_2, \ldots, h_l\}$ of l $(< L)$ features is extracted. It is possible to categorize these schemes as follows:
  1. *Linear schemes:* In this case,
  $$h_j = \sum_{i=1}^{L} \alpha_{ij} f_i$$
  That is, each element of $\mathcal{H}$ is a linear combination of the original features. Note that feature selection is a specialization of feature extraction. Some prominent schemes under this category are:
     a. *Principal components (PCs)*: Consider the data set of n vectors in an L-dimensional space; this may be represented as a matrix A of size $n \times L$. The covariance matrix Σ of size $L \times L$ associated with the data is computed and the eigenvectors of Σ form the principal components. The eigenvector corresponding to the largest eigenvalue is the first principal component (PC). Similarly, the second largest eigenvalue provides its corresponding eigenvector as the second PC. Finally, the eigenvector corresponding to the l^{th} largest eigenvalue is the l^{th} PC. Both the original feature vectors and PCs are sufficiently powerful to represent any data vector. So, PCs may be viewed as linear combinations of the given features.
     b. *Non-negative matrix factorization (NMF)*: Even when the data is non-negative, it is possible that PCs have negative entries. However, it is useful to have representations

using non-negative entries; NMF is such a factorization of $A_{n\times L}$ into a product of $B_{n\times l}$ and $C_{l\times L}$. Its use is motivated by the notion that NMF can be used to characterize objects in an image represented by A. In NMF, the columns of B can be viewed as linear combinations of the columns of A because of linear independence. We will examine, in detail, the concepts of eigenvalue, eigenvector and linear independence in later chapters.

2. *Non-linear feature extraction:* Here, we represent using $\mathcal{H} = \{h_1, \ldots, h_l\}$, such that

$$h_i = t(f_1, f_2, \ldots, f_L),$$

where t is a non-linear function of the features. For example, if $\mathcal{F} = \{f_1, f_2\}$, then $h_1 = af_1 + bf_2 + cf_1f_2$ is one such non-linear combination; it is non-linear because we have a term of the form f_1f_2 in h_1.
Autoencoder is a popular, state-of-the-art, non-linear feature extraction tool. Here, a neural network which has an encoder and a decoder is used. The middle layer has l neurons so that the l outputs from the middle layer give an $l(< L)$-dimensional representation of the L-dimensional pattern that is input to the autoencoder. Note that the encoder encodes or represents the L-dimensional pattern in the l-dimensional space while the decoder decodes or converts the l-dimensional pattern into the L-dimensional space. Note that it is called autoencoder because the same L-dimensional pattern is associated with the input and output layers.

1.5.4 Model Selection

Selection of the model to be used to train an ML system depends upon the nature of the data and knowledge of the application domain. For some applications, only a subset of the ML models can be used. For example, if some features are numerical and others are categorical, then classifiers based on perceptrons and support vector machine (SVM) are not suitable as they compute the dot product between vectors and dot products do not make sense when some values in the corresponding vectors are non-numerical. On the other hand, Bayesian models and decision tree–based models are ideally suited to deal with such data as they depend upon the frequency of occurrence of values.

1.5.5 Model Learning

This step depends on the size and type of the training data. In practice, a subset of the labelled data is used as training data for learning the model and another subset is used for model validation or model evaluation. Some of the ML models are highly transparent while others are opaque or black box models. For example, decision tree–based models are ideally suited to provide transparency; this is because in a decision tree, at each internal or decision node, branching is carried out based on the value assumed by a feature. For example, if the height of an object is larger than 5 feet, it is likely to be an adult and not a child; such easy-to-understand rules are abstracted by decision trees. Neural networks are typically opaque as the outputs of intermediate/hidden layer neurons may not offer transparency.

1.5.6 Model Evaluation

This step is also called **model validation**. This step requires specifically earmarked data called **validation data**. It is possible that the ML model works well on the training data; then we say

that the model is well trained. However, it may not work well on the validation data. In such a case, we say that the ML model overfits the training data. In order to overcome overfitting, we typically use the validation data to tune the ML model so that it works well on both the training and validation data sets.

1.5.7 Model Prediction

This step deals with testing the model that is learnt and validated. It is used for prediction because both classification and regression tasks are predictive tasks. This step employs the test data set earmarked for the purpose. In the real world, the model is used for prediction as new patterns keep coming in. Imagine an ML model built for medical diagnosis. It is like a doctor who predicts and makes a diagnosis when a new patient comes in.

1.5.8 Model Explanation

This step is important to explain to an expert or a manager why a decision was taken by the ML model. This will help in explicit or implicit feedback from the user to further improve the model. Explanation had an important role earlier in expert systems and other AI systems. However, explanation has become very important in the era of DL. This is because DL systems typically employ neural networks that are relatively opaque. So, their functioning cannot be easily explained at a level of detail that can be appreciated by the domain expert/user. Such opaque behaviour has created the need for explainable AI.

1.6 Search and Learning

Search is a very basic and fundamental operation in both ML and AI. Search had a special role in conventional AI where it was successfully used in problem solving, theorem proving, planning and knowledge-based systems.
Further, search plays an important role in several computer science applications. Some of them are as follows:

- Exact search is popular in databases for answering queries, in operating systems for operations like *grep*, and in looking for entries in symbol tables.
- In ML, search is important in learning a classification model, a proximity measure for clustering and classification, and the appropriate model for regression.
- Inference is search in logic and probability. In linear algebra, matrix factorization is search. In optimization, we use a regularizer to simplify the search in finding a solution. In information theory, we search for purity (low entropy).

So, several activities including optimization, inference and matrix factorization that are essential for ML are all based on search. Learning itself is search. We will examine how search aids learning of each ML model in the respective chapters.

1.7 Explanation Offered by the Model

Conventional AI systems were logic-based or rule-based systems. So, the corresponding reasoning systems naturally exhibited transparency and, as a consequence, explainability. Both forward and

backward reasoning was possible. In fact, the same knowledge base, based on experts' input, was used in both diagnosis and in teaching because of this flexibility. Specifically, the knowledge base used by the MYCIN expert system was used in tutoring medical students using another expert system called GUIDON.
However, there were some problems associated with conventional AI systems:

- There was no general framework for building AI systems. Acquiring knowledge, using additional heuristics and dealing with exceptions led to adhocism; experience in building one AI system did not simplify the building of another AI system.
- Acquiring knowledge was a great challenge. Different experts typically differed in their conclusions, leading to inconsistencies. Conventional logic-based systems found it difficult to deal with such inconsistent knowledge.

There has been a gradual shift from using knowledge to using data in building AI systems. Current-day AI systems, which are mostly based on DL, are by and large data dependent. They can learn representations automatically. They employ variants of multi-layer neural networks and backpropagation algorithms in training models.
Some difficulties associated with DL systems are:

- They are data dependent. Their performance improves as the size of the data set increases. So, they need larger data sets. Fortunately, it is not difficult to provide large data sets in most of the current applications.
- Learning in DL systems involves a simple change of weights in the neural network to optimize the objective function. This is done with the help of backpropagation, which is a gradient-descent algorithm and which can get stuck with a locally optimal solution. Combining this with large data sets may possibly lead to overfitting. This is typically avoided by using a variety of simplifications in the form of regularizers and other heuristics.
- A major difficulty is that DL systems are black box systems and lack explanation capability. This problem is currently attracting the attention of AI researchers.

We will be discussing how each of the ML models is equipped with explanation capability in the respective chapters.

1.8 Data Sets Used

In this book, we make use of two data sets to conduct experiments and present results in various chapters. These are:

- **Data Sets for Classification**
  1. *MNIST Handwritten Digits Data Set:*
     - There are 10 classes (corresponding to digits 0, 1, ..., 9) and each digit is viewed as an image of size 28×28 ($= 784$) pixels; each pixel having values in the range 0 to 255.
     - There are around 6000 digits as training patterns and around 1000 test patterns in each class and the class label is also provided for each of the digits.
     - For more details, visit `http://yann.lecun.com/exdb/mnist/`
  2. *Fashion MNIST Data Set:*
     - It is a data set of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples.

- Each example is a 28×28 greyscale image, associated with a label from 10 classes.
- It is intended to serve as a possible replacement for the original MNIST data set for benchmarking ML models.
- It has the same image size and structure of training and testing splits as the MNIST data.
- For more details, visit `https://www.kaggle.com/datasets/zalando-research/fashionmnist`

3. *Olivetti Face Data Set:*
   - It consists of 10 different images each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses).
   - All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).
   - Each image is of size $64 \times 64 = 4096$.
   - It is available on the scikit-learn platform.
   - For more details, visit `https://ai.stanford.edu/~marinka/nimfa/nimfa.examples.orl\_images.html`
4. *Wisconsin Breast Cancer Data Set:*
   - It consists of 569 patterns and each is a 30-dimensional vector.
   - There are two classes *Benign* and *Malignant*. The number of patterns from *Benign* is 357 and the number of *Malignant* class patterns is 212.
   - It is available on the scikit-learn platform.
   - For more details, visit `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\_breast\_cancer.html`

- **Data Sets for Regression**

1. *Boston Housing Data Set:*
   - It has 506 patterns.
   - Each pattern is a 13-dimensional vector.
   - It is available on the scikit-learn platform.
   - For more details, visit `https://scikit-learn.org/0.15/modules/generated/sklearn.datasets.load\_boston.html`
2. *Airline Passengers Data Set:*
   - This data set provides monthly totals of US airline passengers from 1949 to 1960.
   - This data set is taken from an inbuilt data set of Kaggle called AirPassengers.
   - For more details, visit `https://www.kaggle.com/datasets/chirag19/air-passengers`
3. *Australian Weather Data Set:*
   - It provides various weather record details for cities in Australia.
   - The features include location, min and max temperature, etc.
   - For more details, visit `https://www.kaggle.com/datasets/arunavakrchakraborty/australia-weather-data`

SUMMARY

Machine learning (ML) is an important topic and has affected research practices in both science and engineering significantly. The important steps in building an ML system are:

- Data acquisition that is application domain dependent.
- Feature engineering that involves both data preprocessing and representation.
- Selecting a model based on the type of data and the knowledge of the domain.
- Learning the model based on the training data.
- Evaluating and tuning the learnt model based on validation data.
- Providing explanation capability so that the model is transparent to the user/expert.

EXERCISES

1. You are given that 9×17 is 153 and 4×17 is 68. From this data, you need to learn the value of 13×17. Which learning paradigm is useful here? Specify any assumptions you need to make.
2. You are given the following statements:
   a. The sum of two even numbers is even.
   b. 12 is an even number.
   c. 22 is an even number.

   What can you deduce from the above statements?
3. Consider the following statements:
   a. If x is an even number, then $x+1$ is odd and $x+2$ is even.
   b. 34 is an even number.

   Which learning paradigm is used to learn that 37 is odd and 38 is even?
4. Consider the following reasoning:
   a. If x is odd, then $x+1$ is even.
   b. 22 is even.

   You have learnt from the above that 21 is odd. Which learning paradigm is used? Specify any assumptions to be made.
5. Consider the following attributes. Find out whether they are nominal, ordinal or numerical features. Give a reason for your choice.
   a. Telephone number
   b. Feature that takes values from {ball, bat, wicket, umpire, batsman, bowler}
   c. Temperature
   d. Weight
   e. Feature that takes values from {short, medium height, tall}
6. Let x_i and x_j be two l-dimensional unit norm vectors; that is, $\| x_i \| = 1$ and $\| x_j \| = 1$. Derive a relation between the Euclidean distance $d(x_i, x_j)$ and cosine of the angle between x_i and x_j.
7. Consider the data set:

$$(1,1,1), (1,1,2), (1,1,3), (1,2,2), (1,1,-), (6,6,10).$$

Predict the missing value in the pattern vector $(1, 1, -)$ using KNNs with $K = 3$. What happens when $K = 5$?

8. Consider the data in Q7 above. Use this data to predict the missing value in the pattern vector (100,–,100) with $K = 1$. Is there any problem with (100,–,100)? What can you say about this pattern?
9. Consider the following 4 two-dimensional vectors. Let

$$X_1 = (1, 100000), X_2 = (2, 100000), X_3 = (1, 200000), X_4 = (2, 200000)$$

   a. How do we use range scaling scheme in this example.
   b. Normalize the data using standard scaler.
10. Show that feature selection is a special case of feature extraction.

Bibliography

- Sastry, PS. "Introduction to Statistical Pattern Recognition" in *Pattern Recognition*, National Programme on Technology Enhanced Learning. https://nptel.ac.in/courses/117108048
- Murphy, KP. 2012. *Machine Learning: A Probabilistic Perspective*, The MIT Press: Cambridge, England.
- Murty, MN and Biswas, A. 2019. *Centrality and Diversity in Search: Roles in A.I., Machine Learning, Social Networks, and Pattern Recognition*, Springer Briefs in Intelligent Systems, Springer.
- Aggarwal, M and Murty MN. 2021. *Machine Learning in Social Networks: Embedding Nodes, Edges, Communities, and Graphs*, Springer Nature.
- Bishop, CM. 2005. *Neural Networks for Pattern Recognition*, Oxford University Press.
- Murty MN and Susheela Devi, V. 2015. *Introduction to Pattern Recognition and Machine Learning*, World Scientific Publishing Co. Pte. Ltd.: Singapore.
- Murty MN and Susheela Devi, V. 2011. *Pattern Recognition: An Algorithmic Approach*, Springer Science and Business Media.
- Ranga Suri, NNR, Murty MN and Athithan G. 2019. *Outlier Detection: Techniques and Applications - A Data Mining Perspective*, Intelligent Systems Reference Library, Springer.

Colour Plate 1

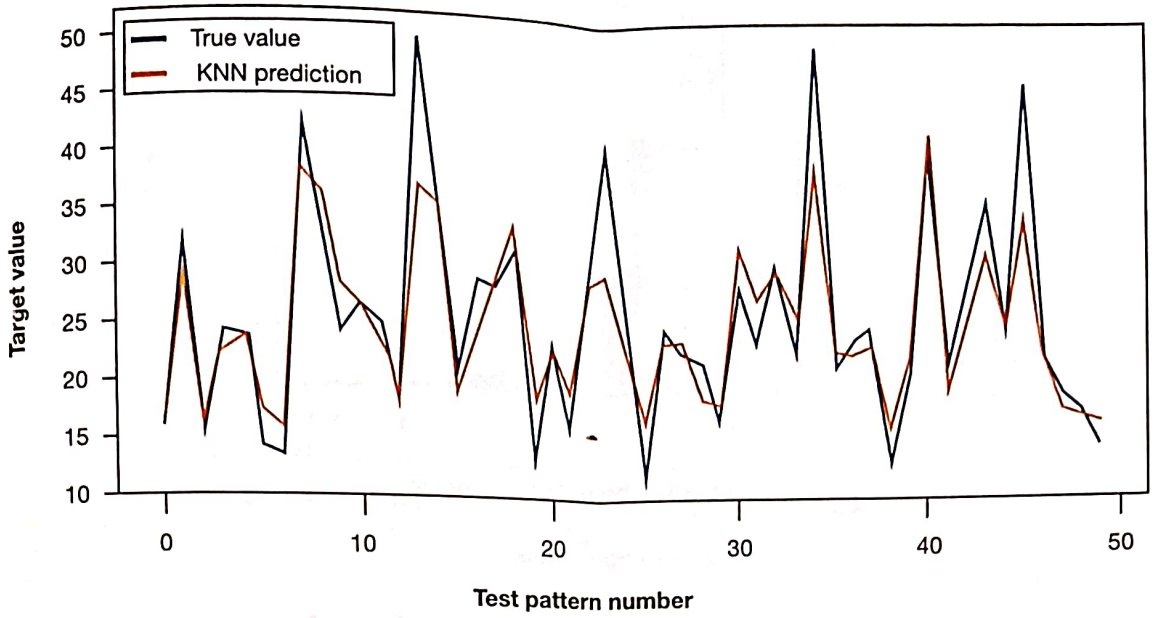
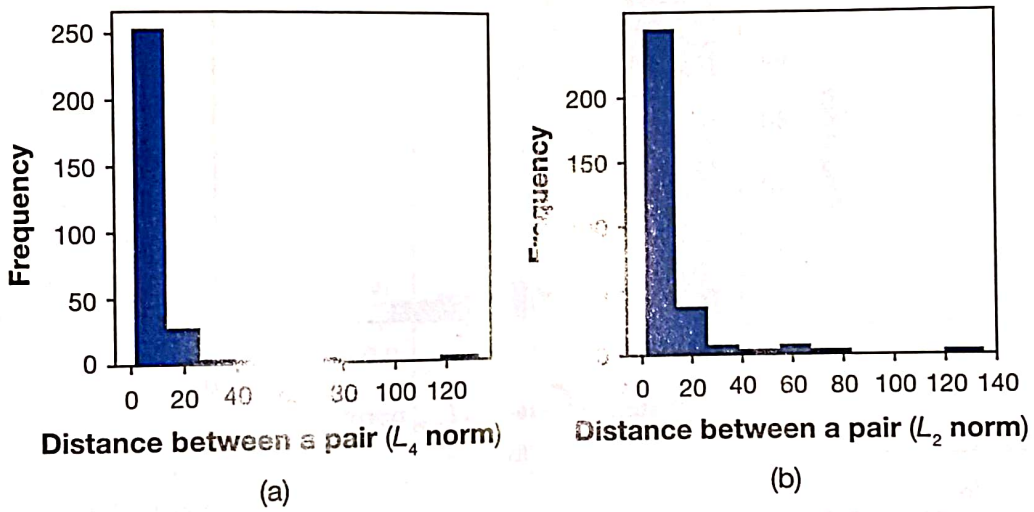


FIG. 2.7 KNN regression: results on the Boston Housing data



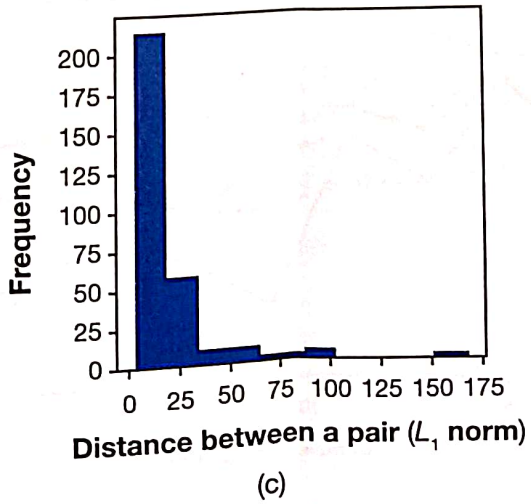
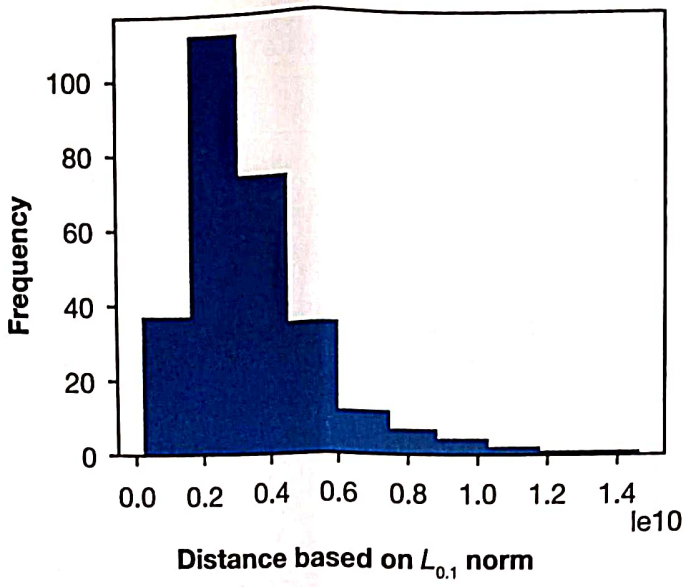
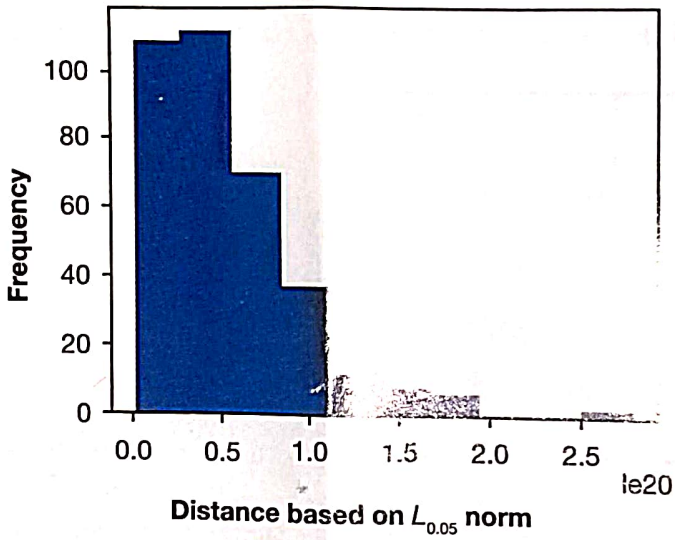


FIG. 2.9 Concentration of distances using different norms



(a)



(b)

FIG. 2.10 Concentration of distances using different norms

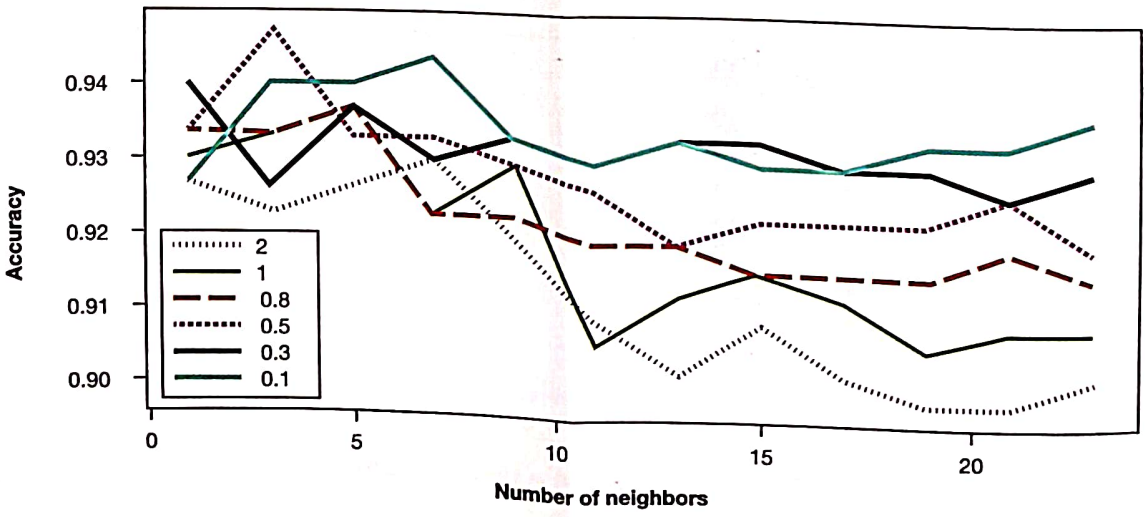


FIG. 2.11 Accuracy of KNN using different norms

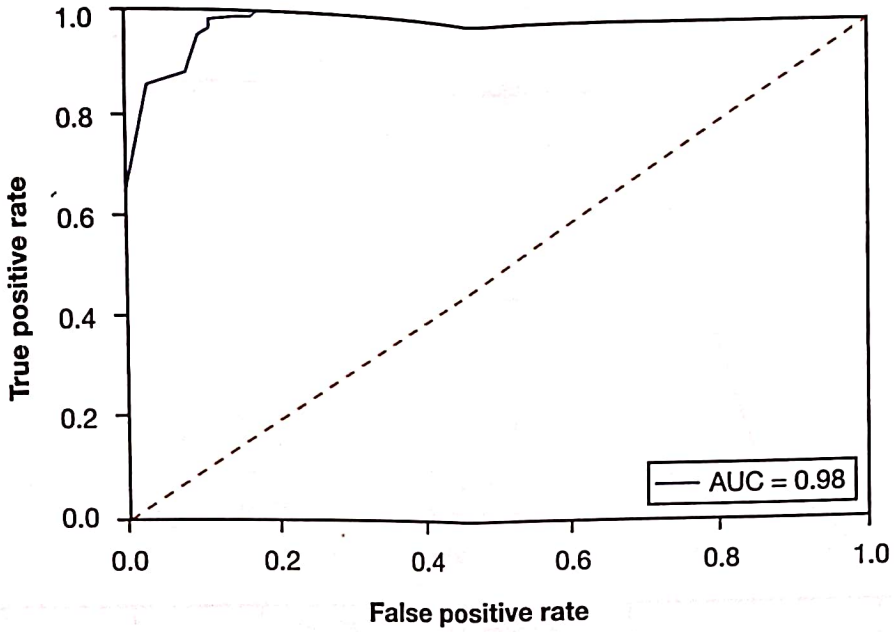


FIG. 2.12 ROC curve for Breast Cancer data using KNN

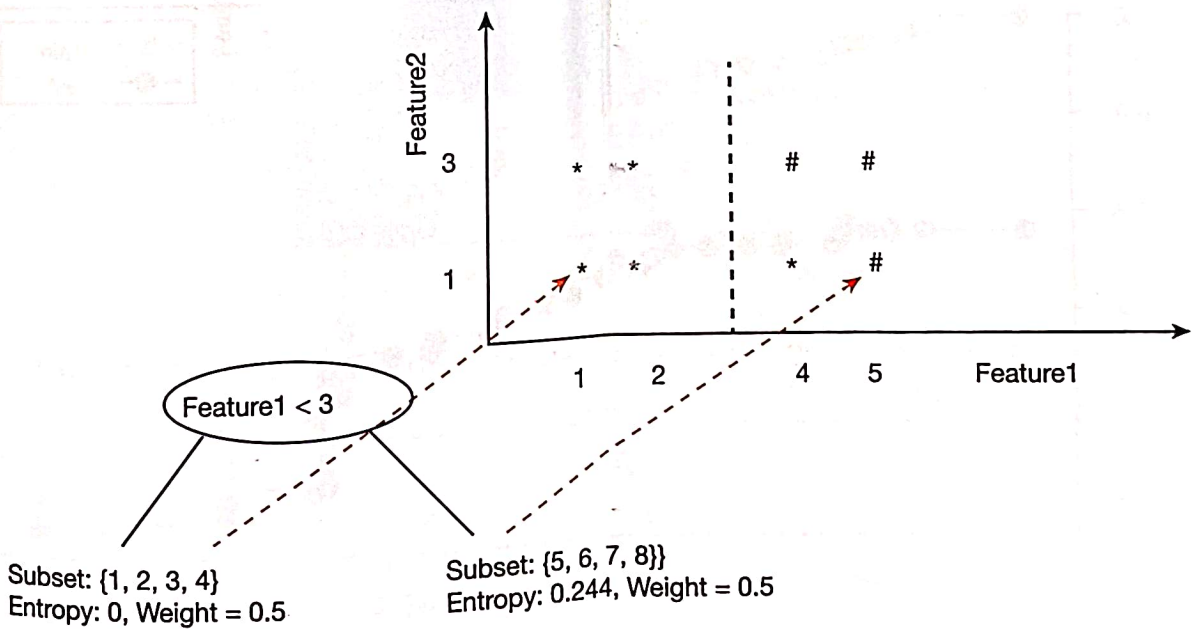


FIG. 3.2 The process of splitting at the root

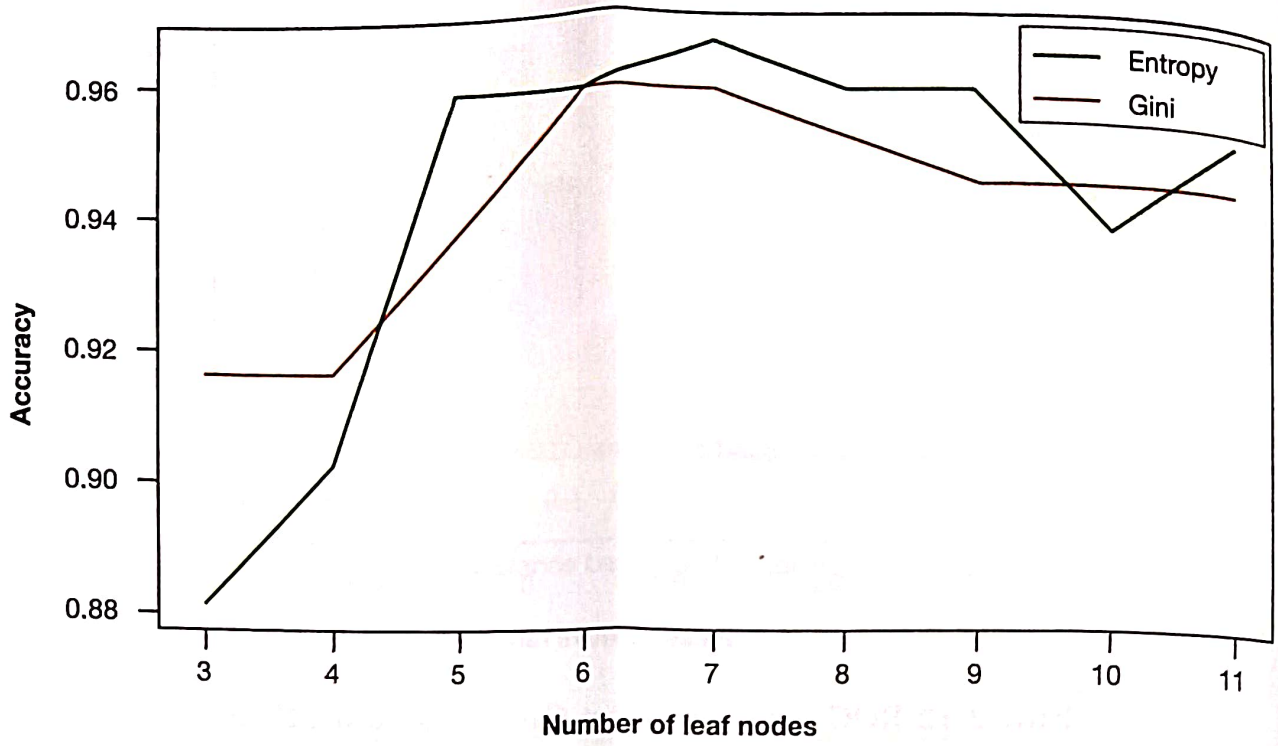


FIG. 3.11 Accuracies of decision trees using Gini index and entropy

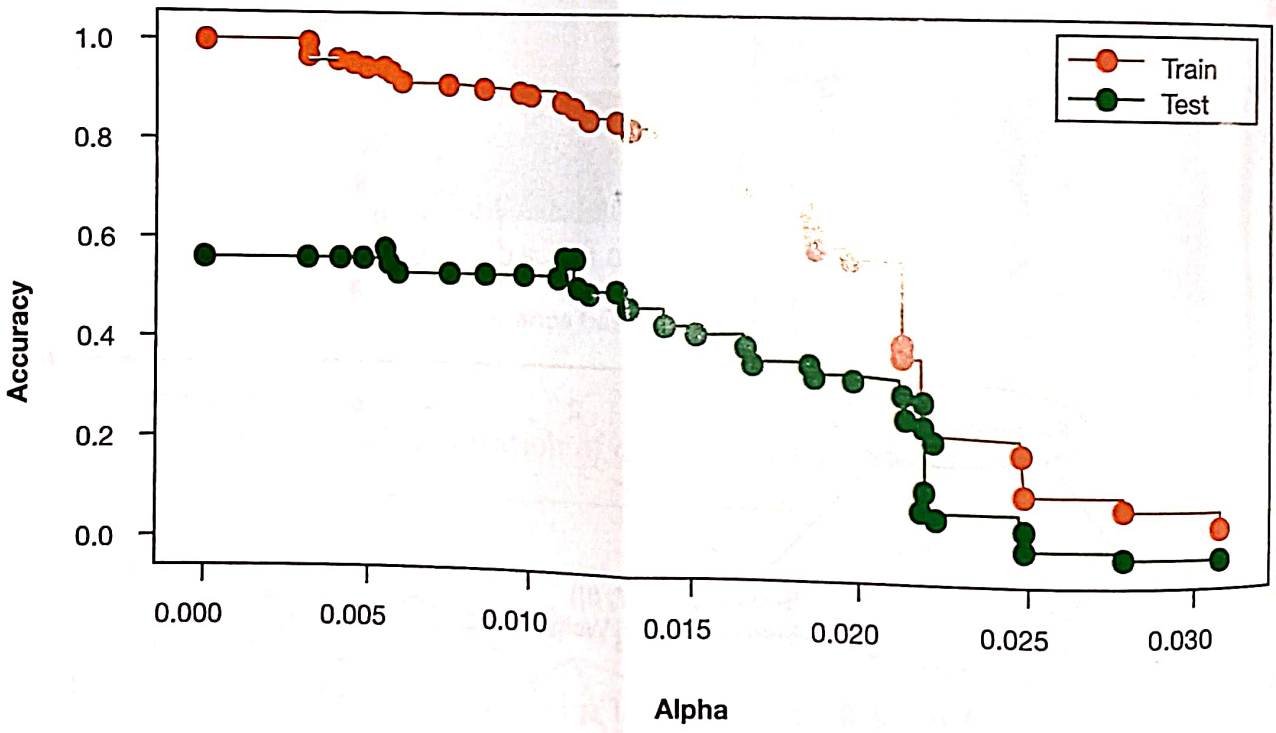


FIG. 3.14 Alpha versus accuracy on the Olivetti Face data set

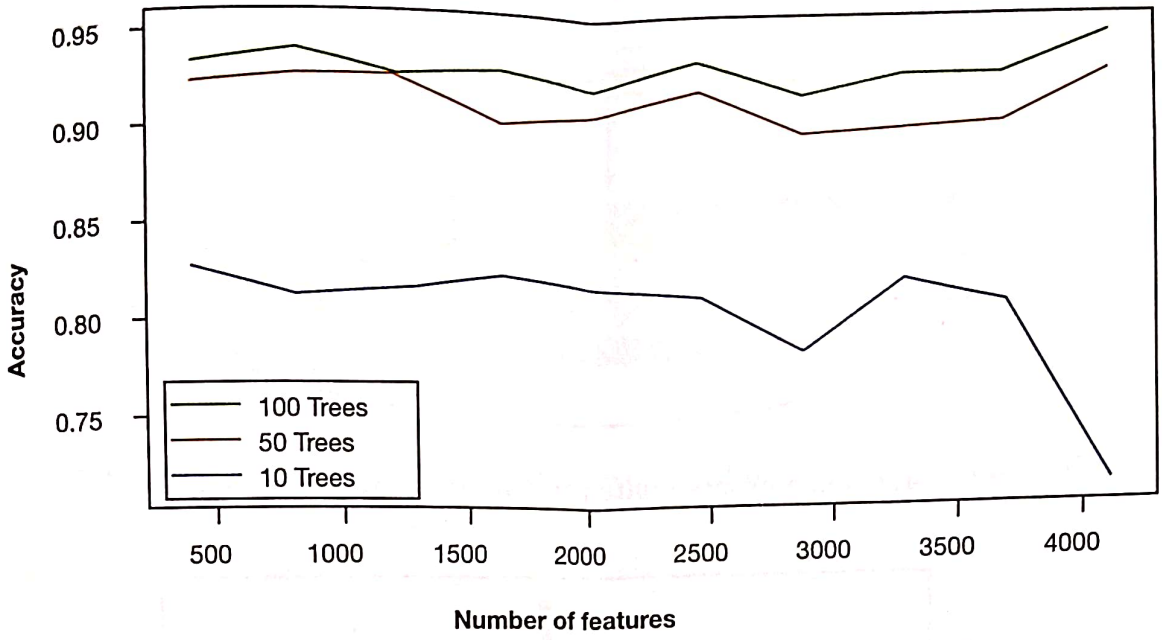


FIG. 3.16 Number of features versus accuracy using a random forest classifier on the Olivetti Face data set

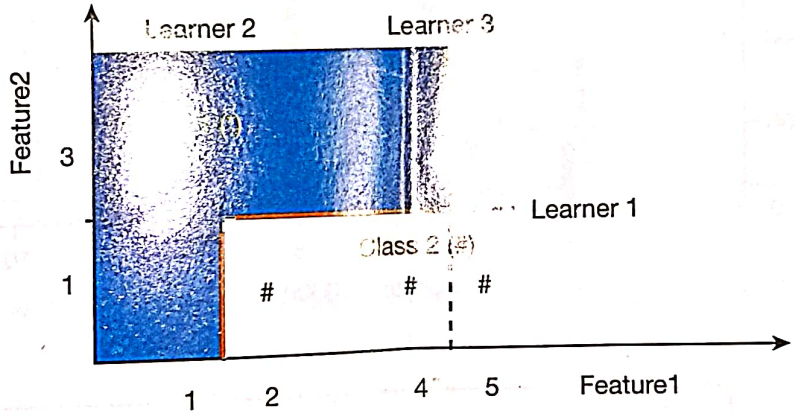


FIG. 3.20 Illustration of the boundary between the two classes using AdaBoost

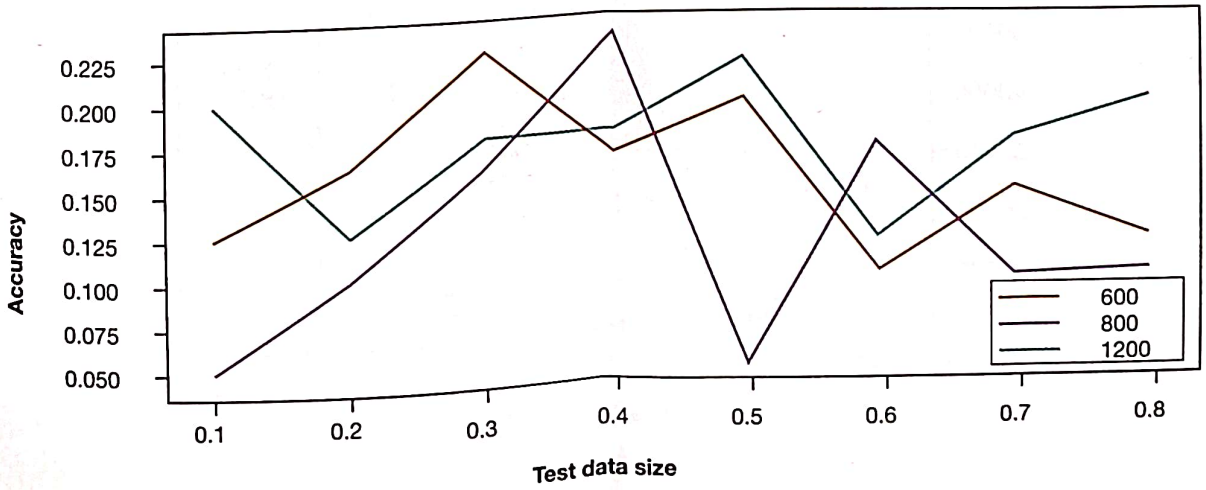


FIG. 3.23 Results of the AdaBoost classifier on the Olivetti Face data set with different number of weak learners (600, 800, 1200) and different test data sizes

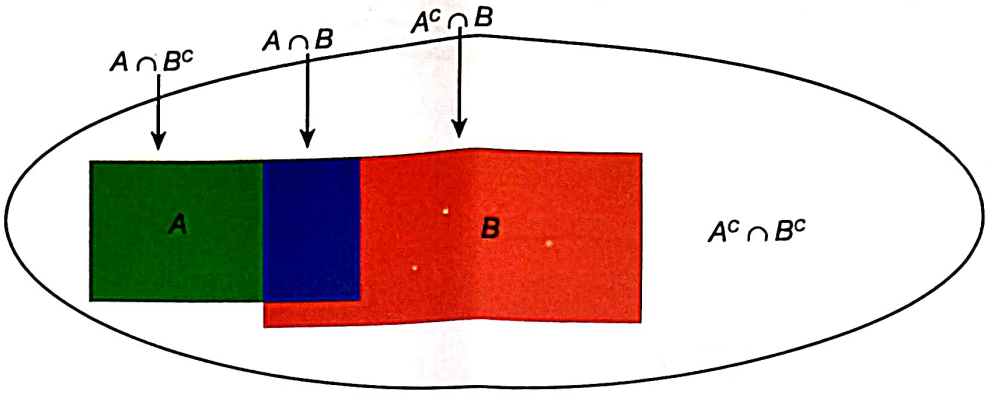
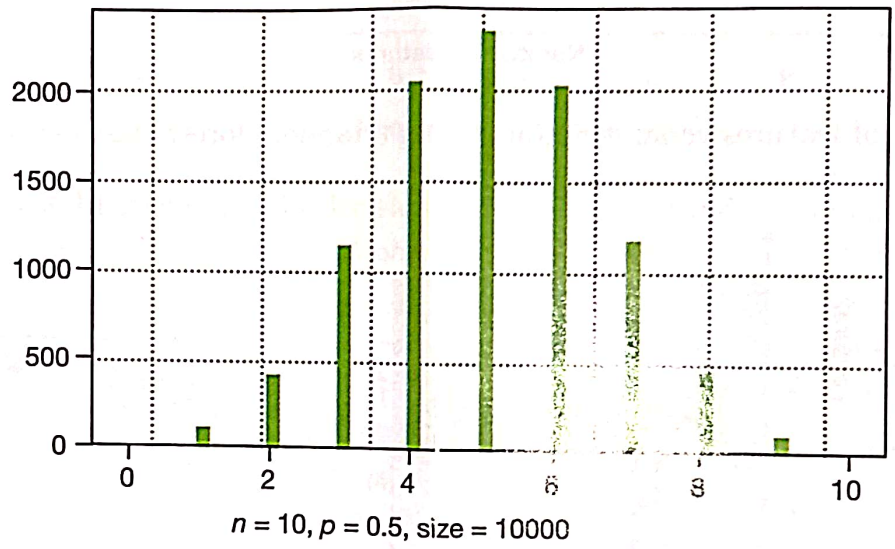
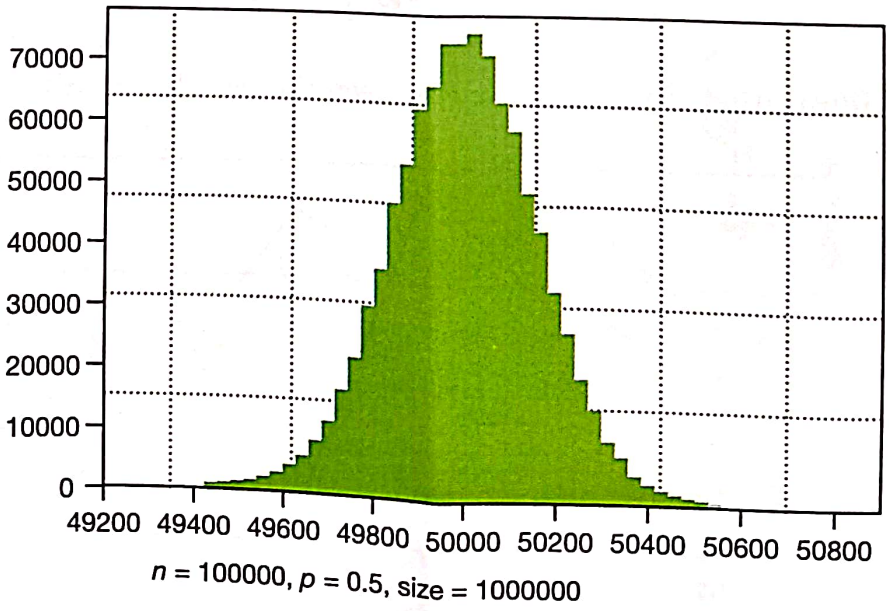


FIG. 4.1 Some events related to two given events A and B



(a)



(b)

FIG. 4.4 Probability mass function of the binomial RV

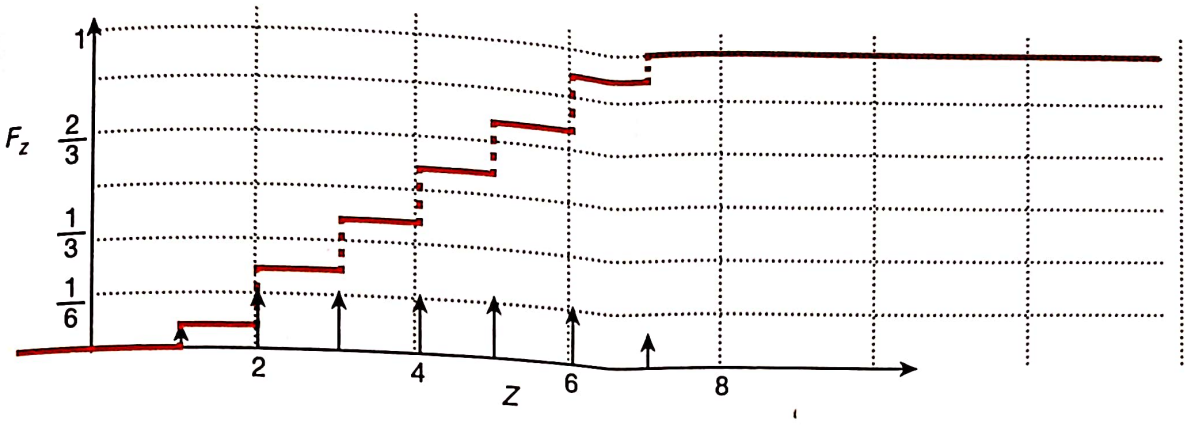


FIG. 4.5 An example of the cumulative distribution function of an RV

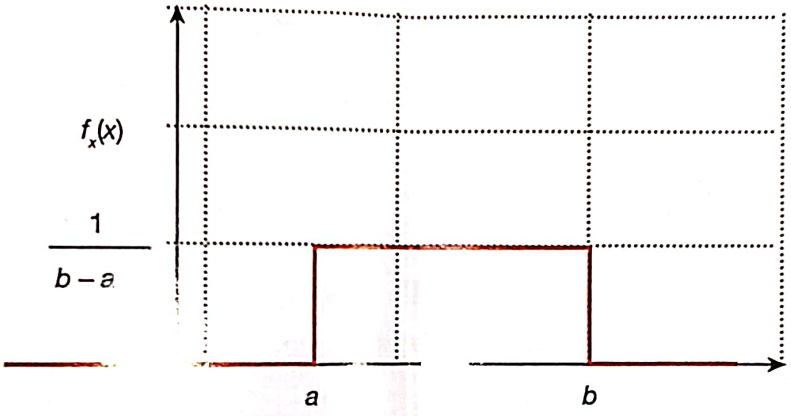


FIG. 4.6 Probability density function of a uniform RV X

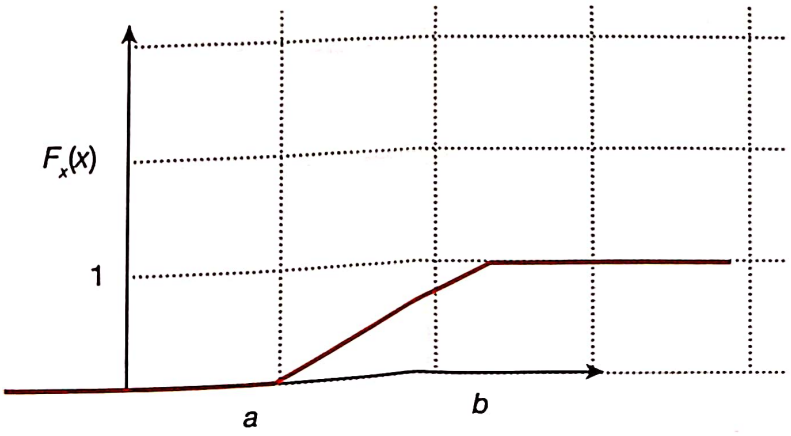
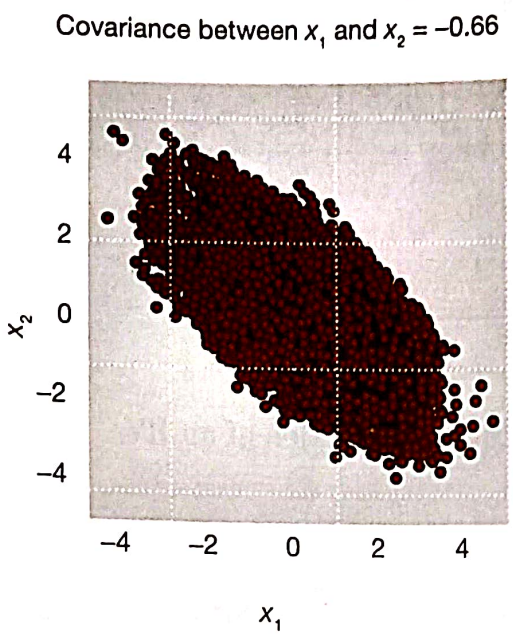
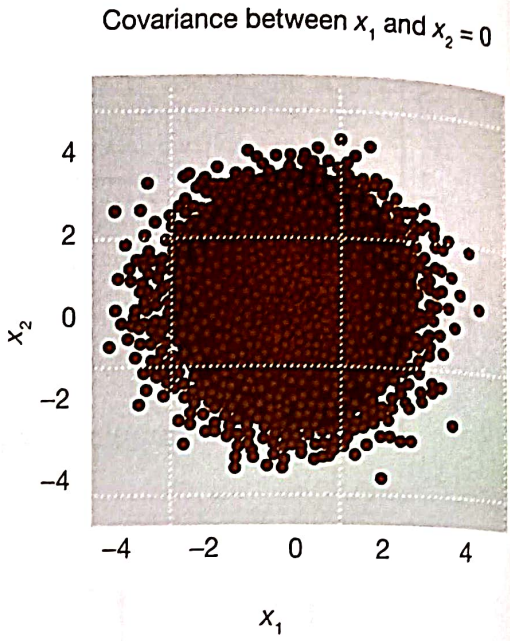


FIG. 4.7 Cumulative distribution function of the uniform RV X



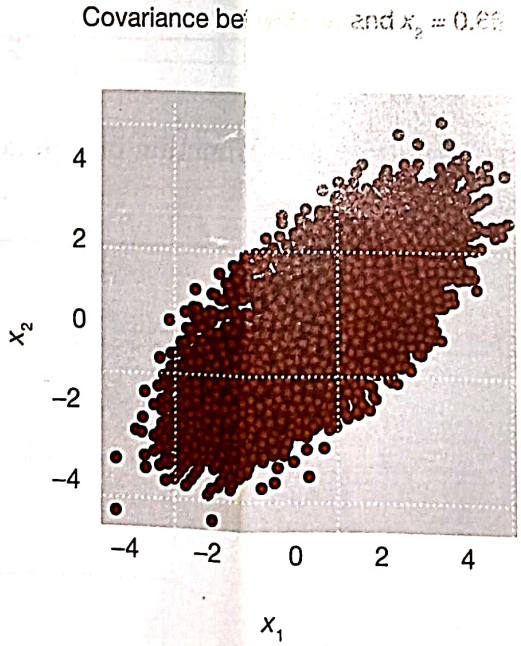
$$\Sigma = \begin{bmatrix} 1 & -0.66 \\ -0.66 & 1 \end{bmatrix}$$

(a)



$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(b)



$$\begin{bmatrix} 1 & 0.66 \\ 0.66 & 1 \end{bmatrix}$$

(c)

FIG. 4.10 Normally distributed two-dimensional points