Chapter 1
Introduction

Abstract In this chapter we provide a brief introduction to machine learning (ML). We introduce some of the important steps in ML including data acquisition, feature engineering, model selection, model estimation, model evaluation, and model explanation.

1.1 Machine Learning (ML) System

Machine Learning (ML) has been an active and important area of study for more than six decades. The phenomenal growth in ML may be attributed, among other things, to the availability of large datasets and improved computing power. It is important to consider Deep Learning (DL) as an offshoot of ML; this is because the generic framework for DL is based on neural networks which were widely studied under the umbrella of ML for more than six decades. In fact, perceptron classifier is the earliest popular ML tool and perceptron forms the basic building block of DL architectures. In the combined space of ML and DL, building a learning system involves the following crucial steps as illustrated in Fig. 1.1.

1. *Data Acquisition:* It depends to a large extent on the knowledge of the application domain. For example, to distinguish between *adults* and *children*, measurements of their *height* and *weight* are adequate; however, to distinguish between *normal* and *abnormal* health status of humans their *body temperature* and *blood test* results might be more important to measure. Further, it is likely that in some domains, there could be missing data possibility. This could be a consequence of being unable to measure the feature value or the entries in the physical records being illegible. There are some statistical tools to deal with the missing values.
2. *Feature Engineering:* This step involves a combination of data preprocessing and representation or dimensionality reduction. Specifically,

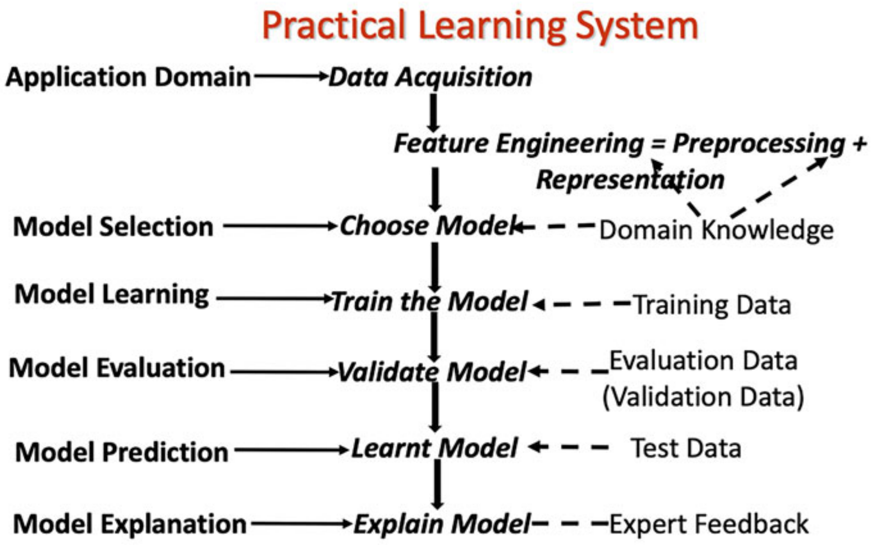

Fig. 1.1 Important steps in machine learning

- **Preprocessing:** Typically there are three different difficulties encountered in data collection.

  a. **Missing Data:** It is possible that some values in the data may be missing. This could be because there was no measurement taken or data was not recorded.
  b. **Data from Different Domains:** The scales behind different features could be highly different. This will bias the proximity measure to be dependent on features that have larger size domains, ignoring the contributions of features with smaller ranges.
  c. **Presence of Outliers:** An outlier is a data item that is either noisy or erroneous. Presence of an outlier could be because of noise in the measuring instruments or erroneous data entries.

3. *Model Selection:* It depends upon the types of data and relevant domain knowledge. For some types of data 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 machines (SVM) are not suitable whereas Bayes classifier and decision-tree based classifiers are ideally suited to deal with such data.
4. *Model Estimation:* It may also be called *model learning*. This learning is dependent on the size and type of the training data. In practice a subset of the labelled data is used as training data and another subset is used for model validation.

5. *Model Evaluation:* This step is also called *model validation*. This step requires a 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.
6. *Model Explanation:* This step is important to get expert's attention and feedback from the expert. Explanation had an important role earlier in expert systems and other artificial intelligence (AI) systems. However, explanation has become all the more important in the era of Deep Learning (DL). It is because DL systems typically employ neural networks and their functioning cannot be easily explained at a level of detail that can be appreciated by the domain expert. Such an opaque behaviour has motivated the need for explainable AI.

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

We consider these steps in detail in the subsequent sections.

1.2 Main Steps in an ML System

An ML system is built using several steps. The main steps in the systems are detailed in this section.

1.2.1 Data Collection/Acquisition

The input to an ML system is a set of *data instances or data items*; so data collection is an important step. However, it highly depends on the application domain. For example, consider classification of villages into those which have *fertile lands* versus those which have *barren lands*. In this application, data related to the name of the village, how far away it is from the capital city, what is the height of tallest hillock in the village, etc. may not be so important. On the other hand, data corresponding to the soil moisture content, yield of various crops in the past, cost of a hectare of land, etc. in the village may be more important.

Similarly, in classifying documents into *politics* and *sports* classes, data pertaining to the age of the document, length of the document, and document file type like whether it is a pdf file, a doc file, or an excel file may not be important. However.

frequency of occurrence of some related words like *election, party, candidate, ball, bat, soccer, etc.* are more relevant.

So, the kind of data that needs to be collected or acquired is application/problem specific. In dealing with such practical applications, one will face several difficulties including:

- *Missing Data:* It is possible that in an application domain all the data that one would like to collect is not available. For example, for one book we may have the entire soft copy in the form of a pdf and for another book that is currently out-of-stock, it is possible that a soft copy of only the table of contents is available.

  In the *education* domain, it is possible that students in section A are tested on their understanding of concept 1, whereas students in section B are tested on some other concept, say concept 2, but not on concept 1. So, when this data is used in analysing the performance of students across both the sections, there is a case of missing data as depicted in Table 1.1. Consider the first row in the table; it corresponds to student 1 who is in section A and is not evaluated on concept 2; so, the entry corresponding to concept 2 is NA. Similarly, in the second row corresponding to student 2 from section B, information under concept 1 is not available; so the corresponding entry is NA. There could be other students who are assigned to one of the two sections and so there will be one NA entry in each row for each such student.

  In general, there can be zero or more missing entries corresponding to each item represented as a vector or some other structure like a string, a tree or a graph.
- *Different Types of Data:* There are several applications where we encounter different types of data in the same application. For example, in the agriculture domain consider the application of predicting the yield of soybean crop. A possible set of different data types acquired to build an ML model is shown in Table 1.2. Even in this simple example, one can see that there are different types of data: *types of soybean seeds* is a *binary* variable in this case with two possible

Table 1.1 Example of missing data

Student	Section	Concept 1	Concept 2
Student 1	Section A	90	NA
Student 2	Section B	NA	30
Student 3	Section B	NA	75
	⋮		

Table 1.2 Example of missing data

Attribute	Type of data	Domain
Type of soybean seed	Binary	{type1, type2}
Year	Numerical	[2000, 2015]
Rainfall	Numerical	[83 mm,11871 mm]
Temperature	Numerical	[1.9 °C, 51 °C]
Relative Humidity	Numerical	[50%, 73%]
Soybean Yield	Numerical	[580, 1080]

values type1 and type2 seeds. The variable *Year* indicates the year corresponding to the yield; here the possible values are integers in the range [2000, 2015]. The variables *Temperature, Relative Humidity, and Soybean Yield* are also numerical but their values are floating point numbers. Note that temperature is represented in degrees Celsius, relative humidity in percentage, and soybean yield in quintals per hectare.

In general data can be either:

1. *Categorical:* These are divided into either *nominal* or *ordinal* types. In the case of nominal data, there is no order among the elements of the domain. For example, for geometric shapes the domain is {circle, ellipse, triangle, rectangle, square}. This data is of categorical type and the elements of the domain are not ordered. In the case of ordinal data, there is an inherent order among the values of the domain. For example, the domain of the variable *size* could be {very small, small, medium, large, very large}; here, an ordering among the elements of the domain is observed.
2. *Numerical:* In the case of numerical data, the domain of values of the data type could be either a set of integers or a set of real numbers. For example, in Table 1.2, the domain of *Year* is the set {2000, 2001, $\cdots$, 2015} which is a collection of integer values. Further, the domain of *Relative Humidity* is a collection of floating point numbers in the range [50, 73].

   It is possible to have binary values in the domain for either categorical or numerical. For example domain of *status* could be {Pass, Fail} and this variable is nominal; an example binary ordinal type is {short, tall} of humans based on their height. A very popular binary numerical type is {0, 1}; also in the 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.

- *Difference in Domain Sizes:* In an application, if we encounter different types of data, then it is quite likely to have different domain sizes. For example, for the data types shown in Table 1.2, we have binary data (seed type), integer data (Year), and different sized floating point data types (rainfall, temperature, relative humidity, and yield). These size and type differences can create a problem while matching different data items using some proximity score in ML.

  Consider for example, classification of objects into one of two classes: *human* or *chair*. Let the objects be represented by *height* in meters and *weight* in grams. Consider an adult represented by the vector (1.7, 65,000) and a chair represented by the vector (1, 5000) where the heights of the adult and the chair in meters are 1.7 and 1 respectively and the weights of the adult and the chair in grams are 65,000 and 5000 respectively. Note that the domain of *height* can be [0.5, 2.5] and the domain of *weight* can be [2000, 200,000] in this example. So, there is a large difference in the sizes of these two domains.

  In ML, matching is done by using a *proximity measure* which can be either a distance/dissimilarity measure or a similarity measure. Two data items, represented as l-dimensional vectors X and Y, 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

$$d(X, Y) = \sqrt{\sum_{i=1}^{l} (X_i - Y_i)^2}.$$

The cosine similarity is given by

$$cos(X, Y) = \frac{X^t Y}{||X|| ||Y||},$$

where $X^t Y$ is the dot product between the vectors X and Y, and $||X||$ is the euclidean distance between X and the origin.

Now the euclidean distance between the adult and chair vectors given above is

$$\sqrt{1.7 - 1)^2 + (65{,}000 - 5000)^2} = \sqrt{0.49 + 3.6 \times 10^9} = 60{,}000.000004 \approx 60{,}000.$$

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

$$\frac{1.7 + 325 \times 10^6}{\sqrt{25{,}000{,}001} \times \sqrt{4{,}225{,}000{,}002.89}} = 0.9999999849 \approx 1.0.$$

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* and the contributions of *height* are negligible. This is because of the difference in the magnitudes of the two data types. This example illustrates how the magnitudes/ranges of values of different data types contribute differently to the overall proximity. A solution to this problem is provided by normalizing the data. A simple *normalization* scheme scales the magnitude of values such that all the features have the same range. For example, a popular scheme is to transform the data vectors so that they are zero mean unit variance vectors.

- *Presence of Outliers:* A common problem across various applications is the presence of outliers. A data item is usually called an *outlier* if it

  1. assumes values that are *far away* from those of the average data items
  2. *deviates* from the normally behaving data item
  3. is not connected/similar to any other object in terms of its characteristics.

  Outliers can occur because of different reasons including

1. *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.
2. *Erroneous data entry:* Outlying data can occur at the data entry level itself. For example, it is very common to introduce spelling mistakes while entering names. Further, it is common to introduce errors in the names because of typing the neighboring characters, instead of the intended character, on the key board. Also, it is possible to enter the salary erroneously as 2,000,000 instead of 200,000 by typing an extra zero (0).
3. *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 during the early times of a social network. Such isolated entities may or may not be outliers.
4. *Very naturally:* Instead of viewing an outlier as a noisy or unwanted data item, it may be very useful in several applications. For example, a novel idea or a breakthrough in a scientific discipline, a highly paid sportsperson, an expensive car can all be useful and influential outliers.

An outlying data item can be either out-of-range or with-in-range. For example, consider an organization in which the salary values are

$$\{10{,}000,\ 15{,}0000,\ 225{,}000,\ 300{,}000\}.$$

In this case an entry like 2,250,000 is an out-of-range outlier that occurs because of an erroneous zero (0). Also if there are only 500 people drawing 10,000, 400 drawing 150,000, 300 at the salary level 225,000 and 175 drawing 300,000, then an entry like 270,000 could be a with-in-range outlier.

There are different schemes for detecting outliers. They are based on density around various points in the data. If a data point is located in a sparse region, then 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 it could be a possible outlier.

1.2.2 Feature Engineering and Representation

In the previous subsection, we have seen several issues associated with data items. It is convenient and customary to view these items as vectors. *Feature engineering* (FE) is concerned with addressing several problems associated with data vectors. In this section we will examine how these problems could be handled.

- *Missing Data:* There are some ML algorithms that can work even when there are missing data values and in such cases there is no need to do any pre-processing. However there are a large number of other cases where the classifiers cannot handle missing values. So, there is a need to examine techniques for dealing with missing data. Some possibilities are:

– *Use the nearest neighbour:* Let X be a d-dimensional data vector that has its ith component x_i missing. Let $\mathcal{X} = \{X^1, X^2, \cdots, X^n\}$ be the set of n training pattern vectors. Let $X^i \in \mathcal{X}$ be the nearest neighbour of X based on the remaining $l - 1$ (excluding the ith) components. The nearest neighbour of X, $NN(X)$ is defined as

$$NN(X) = argmin_{X^j \in \mathcal{X}} d(X, X^j),$$

where X^j is the jth training pattern and $d(X, X^j)$ is the euclidean distance between X and X^j. Intuitively, $NN(X)$ is in the proximity of X; so $NN(X)$ is at a minimum euclidean distance from X and is maximally similar to X. Predict the value of x_i to be x_i^p, that is if the ith component, x_i, of X is missing, then use the ith component of $X^p = NN(X)$ instead.

– *Use a larger neighbourhood:* It is well-known that the NN-based scheme can be easily affected by outliers. For example, if $NN(X)$ is an outlier, then the value predicted based on $NN(X)$, that is an outlier pattern can be erroneous. Even if $NN(X)$ is not an outlier, it is advisable to use more neighbors of X for predicting the missing value x_i in a robust manner. So, we use K nearest neighbors (KNNs) of X to predict the missing x_i. Let the KNNs of X, using the remaining $l - 1$ components, from $\mathcal{X}$ be $X1, X2, \cdots, XK$. Now the predicted value of x_i is the average of the ith components of these KNNs. That is the predicted value of x_i is

$$\frac{1}{K} \sum_{j=1}^{K} Xj_i.$$

– *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 x_i 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 ith component of μ^q.

Note that the approaches mentioned above to deal with missing data are all applicable when the components of the vectors are all numerical. However when the data vectors have categorical entries, then one needs to use median or mode instead of centroid. We consider a distance function that can deal with vectors having missing component values next.

• *Combination of categorical and numerical features:* Let the vectors be l-dimensional out of which let m be categorical and the remaining $l - m$ be numerical. Without loss of generality, let $x_1, x_2, \cdots, x_m$ be m categorical features and $x_{m+1}, \cdots, x_l$ be numerical features. There are some classifiers that can deal with vectors that have a mix of categorical and numerical components.

Also, a more recent view has been to represent categorical attribute values using one-hot representation that is popularly used in neural networks. If the number of distinct values of a variable is p, then each value is represented as a binary vector with one 1 and remaining $p - 1$ values being zero (0). The ith distinct value is represented as a binary vector with a 1 in the ith location and remaining zeros.

However, both distance based and dot product based classifiers will face difficulties in dealing with such mixed data. In the case of proximity based classifiers, it is not possible to use euclidean distance type distances or cosine like similarity measures. Some of the possible distances are:

1. *Gower Distance:* It is defined between two vectors X^i and X^j as

$$d(X^i, X^j) = \sum_{p=1}^{m} mat(x_p^i, x_p^j) + \sum_{q=m+1}^{l} \frac{|x_q^i - x_q^j|}{range(q)},$$

where

– $mat(x_p^i, x_p^j) = 0$ if $x_p^i = x_p^j$ else it is 1.
– range(q) is the range of the domain of the numerical feature.

Another normalizing variant involves dividing the sum by the dimensionality l.

We can illustrate this using the data given in Table 1.3. Note that the Gower distance between X^i and X^j is

$$d(X^i, X^j) = 1 + 1 + 0 + \frac{1}{2} + \frac{100}{1000} = 2.6.$$

Its normalized variant is $\frac{2.6}{5} = 0.52$.

2. *Heterogeneous Value Distance Metric* ($HVDM$)*:* It can deal with missing values also. However, it is applicable in classification contexts only. It is defined as

$$HVDM(X^i, X^j) = \sqrt{\sum_{p=1}^{l} d^2(x_p^i, x_p^j)},$$

Table 1.3 Mixed type data used to illustrate gower distance

	Categorical 1	Categorical 2	Categorical 3	Numerical 1	Numerical 2
X^i	a	1	0	1.5	600
X^j	b	0	0	0.5	700
Range	NA	NA	NA	2	1000

where

$$d(x_p^i, x_p^j) = \begin{cases} 1 & if\ x_p^i,\ or\ x_p^j\ is\ unknown \\ nvdm(x_p^i, x_p^j) & if\ p\ is\ categorical \\ ndiff(x_p^i, x_p^j) & if\ p\ is\ numerical \end{cases}$$

where

- $nvdm(x_p^i, x_p^j) = \sqrt{\sum_{c=1}^{C}(P(c|x_p^i) - P(c|x_p^j))^2}$ if there are C classes and $P(c|x_p^i)$ is the posterior probability of class c given that x_p^i is the value of the pth component of X^i, and
- $ndiff(x_p^i, x_p^j) = \frac{|x_p^i - x_p^j|}{4\sigma_p}$, where σ_p is the standard deviation of the values of the pth component.

So, Gower distance can deal with the vectors even if they have both categorical and numerical components. Further, $HVDM$ can deal with vectors that have missing values also; however, it requires computation of posterior probabilities for each of the categorical variables.

- *Varying domain sizes:* In applications where different components of the vectors have different domain sizes, it is possible for some components to dominate in contributing to the distance between any pair of patterns. This can be handled by scaling different components differently and such a process of scaling is called *normalization*. We have seen such a scaling in the computation of both the Gower distance and $HVDM$. There are typically two types of popular normalization schemes:

1. *Scaling using the range:* On any categorical data type, the values of two patterns either match or mismatch, respectively the distance contribution is either zero (0) or 1. Note that both the Gower distance and $HVDM$ use this. So, the contribution of any categorical variable is an element of the set $\{0, 1\}$. To be consistent, in the case of any numerical feature we want the contribution to be in the range [0,1]. This is achieved by scaling the difference by the range of the variable. So, if pth component is of numerical type, then its contribution to the distance between X^i and X^j is

$$\frac{|x_p^i - x_p^j|}{R_p},$$

where R_p is the range of the pth component. Note that the minimum value of this term is 0 (zero) and the maximum value is 1 when $|x_p^i - x_p^j| = R_p$. Such a scaling will ensure the required behaviour and the contribution, to the distance, of a categorical or a numerical component will be in the range [0, 1].

2. *Standardization:* Here, each numerical variable value is scaled so that the mean value of the variable is 0 (zero) and the standard deviation is 1. This

is motivated by the standard normal distribution that has zero mean and unit variance.

This may be illustrated using the following example.

– Let there be 5 l-dimensional data vectors and without loss of generality, let the lth components of the 5 vectors be 600, 800, 200, 1000, and 400.
– The mean of this collection is

$$\frac{600 + 800 + 200 + 1000 + 400}{5} = 600.$$

– We get zero mean data by subtracting this mean from each of the 5 data items to get 0, 200, −400, 400, −200. Note that this is a zero mean data.
– 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

$$\frac{0 + 200^2 + (-400)^2 + 400^2 + (-200)^2}{5} = 80{,}000$$

and the standard deviation is 282.84.
– So, the scaled data is 0, 0.707, −1.414, 1.414, −0.707. Note that this data has zero mean and unit variance.

This is illustrated further using the following 4 two-dimensional vectors. Let

$$X_1 = (1, 100{,}000), X_2 = (2, 100{,}000), X_3 = (1, 200{,}000), X_4 = (2, 2{,}000{,}000)$$

be four vectors with the first feature taking values in $[1, 2]$ and the second feature assuming values from $[100{,}000, 200{,}000]$. The mean of these four vectors is $(1.5, 150{,}000)$. By subtracting the sample mean vector from the four patterns gives us four mean normalized data points which are

$$(-0.5, -50, 000), (0.5, -50, 000), (-0.5, 50{,}000), (0.5, 50{,}000).$$

Now we can make it unit variance data by dividing each value by the sample standard deviation value. Note that the standard deviation for the first feature is $\frac{1}{2}$ and for the second feature, it is 50,000. By dividing the values by their respective standard deviations, we get the vectors

$$(-1, -1), (1, -1), (-1, 1), (1, 1).$$

Note that these four points lie on the surface of a unit circle centered around the origin. In a higher dimensional space, the normalized patterns fall on the surface of a unit hypersphere.

- *Presence of outliers:* In the previous section, we have examined outliers and the reasons for their presence. We will see how the data can be processed to detect and eliminate outliers. Some possibilities are:
  - *Out-of-range outliers:* Some ways of dealing with this kind of outliers are:

    Based on statistics: If the data is normally distributed, an outlier can be characterized based on how far away it is from the mean in terms of the variance/covariance structure. For example, one can fit a normal distribution with mean μ and variance σ^2 to one-dimensional data. It is well-known that in a uni-variate normal distribution, the values in the range $[\mu - 3\sigma, \mu + 3\sigma]$ account for more than 99%. So, any object/value falling outside this range may qualify to be an outlier as it is away from the mean. One may collect all such data points and look for possible outliers and deal with them.
    Knowledge-based approach: knowledge-based If the domain of a variable is known beforehand, then it is easy to identify out-of-range data points and eliminate or correct such entries. For example, removing an extra zero (0) in some numeric entry or using a spell correction software package belong to this category.

  - *within-range outliers:* Typically these outliers can be detected based on the sparseness of the region to which the point belongs. Some possible schemes are:
    - *Clustering based:* One can use a clustering algorithm to cluster the data and identify clusters that have a small number of data items. One can concentrate on such small size clusters and examine for possible outliers. There are clustering algorithms that can generate clusters by taking into account the density of data.
    - *Frequency based:* There are applications like *information retrieval* where it is observed that frequent words are not helpful in classification. So, such frequent words are eliminated. Further, *rare terms* that are infrequent are not important to several users of the search engines or information retrieval systems. Such rare words could be viewed as some kind of outliers as they could be of interest to a very tiny fraction of users.

An important step in ML is to represent the data items as vectors appropriately. While using such vectors, one will encounter several difficulties.

- *Problems with high-dimensional vectors:* Matching based algorithms like NNC *fail in dealing with high-dimensional vectors*. This is because as the dimensionality increases the notion of similarity gets affected. If X is a pattern, then its nearest neighbor $NN(X)$ and its farthest neighbor $FN(X)$ will be located such that as the dimensionality $\rightarrow \infty$, $d(X, NN(X)) \rightarrow d(X, FN(X))$. That is $NN(X)$ and $FN(X)$ lie at the same distance from X corrupting the notion of NN in high dimensional spaces. So, representing the vectors in a lower-dimensional space is important for the matching based algorithms to work properly. Some of the solutions to this problem are:

1. *Approximate matching:* An approximate matching algorithm based on random subspace matching is used to achieve the approximation. It is based on hashing that is sensitive to nearer or local regions. This is studied under the paradigm of locality sensitive hashing (LSH). We study LSH in detail in a subsequent chapter.

   There are other random subspace classifiers based on building one classifier for each random subset of features. Such multiple classifiers are combined to form the overall composite classifier.
2. *Reducing the dimensionality:* Another approach is based on reducing the dimenisonality and then building the ML model on the reduced dimensional patterns. Some popular schemes under this category are:

   a. *Feature Selection:* Here, we select a subset of l features from the given $L(> l)$ features. This subset selection is carried out in a variety of ways. A popular scheme, in the case of categorical features, is based on *mutual information*. We will discuss mutual information based feature selection in a subsequent chapter.
   b. *Linear Feature Extraction:* In this type of schemes, we extract l out of $L(> l)$ features where each of the extracted l features is a linear combination of the L features. There are different schemes under this category. Some popular ones are:

      i. *Principal Components* (PCs)*:* These are the leading eigenvector directions of the covariance matrix of the data. These PCs, or the leading eigenvectors, capture the maximum variance directions in the data. The first PC is in the maximum variance direction, the next PC captures the second largest variance direction, and so on. Also these PCs are uncorrelated pairwise. We will analyse the behaviour of PCs in classification in a later chapter.
      ii. *Non-negative matrix factorization* (NMF)*:* Here, the data (n items) in L dimensional space is represented as a matrix of size $n \times L$; it is called the *data matrix*. Let this matrix be A. It is factorized into a product of two matrices B and C such that

      $$A_{n\times L} \approx B_{n\times l} C_{l\times L}.$$

      Typically, the entries in A are non-negative. The factorization is carried out so that the resulting approximate factors B and C also have non-negative entries. Each of the l columns of the matrix B characterizes a feature extracted. We will consider it in detail in a later chapter.

   c. *Non-Linear Feature Extraction:* In this category, from the given collection of L features, a smaller set of l features is extracted where each of the l extracted features is a non-linear combination of the original L features.

Such non-linear feature extractors are easily achieved with the help of neural networks. Autoencoder is a popular architecture for realizing non-linear feature extraction. We will examine autoencoders in detail later.

- *Visualization:* It is important to verify whether the patterns represented in the low-dimensional $l(< L)$ space preserve some structural properties present in the original L-dimensional space. Visualization tools help us project the data, present either in the L or the l dimensional space, to a two-dimensional space so that we can analyse the projected data manually. There are schemes including the t-distributed stochastic neighbor embedding (t-SNE) plot which is a nonlinear projection based statistical scheme.

1.2.3 Model Selection

Feature engineering is an important step in ML. Once the data is represented in an appropriate manner, then one can select the model appropriately. For example, if there are both numerical and categorical entities, then it is possible to use classifiers based on decision trees, and Bayes classifier; we will not be able to exploit support vector machine and neural network based models directly on such vectors. So, we need to select a model based on the application considered and the training data types involved.

1.2.4 Model Estimation

Once a model for ML is selected, one needs to estimate the parameters associated with the model using the *training data*. For example, a linear discriminant function based classifier can be characterized using $W^t X$ where W and X are l-dimensional vectors. Learning the weight vector W is done using the training data. Given any test pattern X, X is assigned to class1 if $W^t X < 0$, else X is assigned to class2. Similarly, in the case of Bayes classifier one needs to estimate the probability structure, the prior probabilities and likelihood values, underlying the learning problem.

1.2.5 Model Validation

It is important to tune the parameters of the estimated model using additional data called *validation data*. This step is called model validation. The validation data is used to fine tune the model estimation. Typically the available training data is split into train and validation parts in practice. One of the most popular schemes is K-

fold cross-validation. Here, the training data set is partitioned into K equal size sets. Out of these K, $K - 1$ sets are used for training and the remaining one is used for validation. This process is done so that every one of the K subsets participates in validation. Once the parameters are fine tuned, the ML model is ready for use.

1.2.6 Model Explanation

In several practical applications, an expert would typically be involved in providing the underlying domain knowledge, training data, etc. Further, the learnt ML model is to be used by the domain expert, for example in domains like medicine, security, transport, and education. So, it would be possible for the expert to provide an appropriate *feedback* to improve the model if the learning process is *transparent* and easy for the expert to appreciate. In order to meet this requirement, it is essential to provide an explanation that could be easily understood by the expert. For example, decision tree based classifiers are the simplest and are easy for providing an explanation to the expert. ML models based on neural networks fall at the other end of the spectrum as they are typically opaque.

1.3 Data Sets Used

We make use of two data sets to conduct experiments and present results in various chapters of the book. These are:

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 0–255. 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. Visit http://yann.lecun.com/exdb/mnist/ for more details.
2. *The ORL Face Data Set:* Ten different images of 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 112×92. Visit https://ai.stanford.edu/~marinka/nimfa/nimfa.examples.orl_images.html for more details.

1.4 Summary

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

- Data acquisition that is domain application 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 the learnt model based on validation data.

This book deals with **representation** that is the most important part of ML.

References

1. Murphy, K.P.: Machine Learning - A Probabilistic Perspective. MIT Press (2012)
2. Murty, M.N., Biswas, A.: Centrality and Diversity in Search: Roles in A.I., Machine Learning, Social Networks, and Pattern Recognition. Springer Briefs in Intelligent Systems (2019)
3. Aggarwal, M., Murty, M.N.: Machine Learning in Social Networks - Embedding Nodes, Edges, Communities, and Graphs. Springer (2021)
4. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (2005)
5. Murty M.N., Devi V.S.: Introduction to Pattern Recognition and Machine Learning. World-Scientific (2020)
6. Kabán, A.: Fractional norm regularization: learning with very few relevant features. IEEE Trans. Neural Netw. Learn. Syst. **24**(6), 953–963 (2013)
7. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51**(1), 117–122 (2008)