Module 1

Introduction to machine learning

In this chapter, we consider different definitions of the term “machine learning” and explain what is meant by “learning” in the context of machine learning. We also discuss the various components of the machine learning process. There are also brief discussions about different types learning like supervised learning, unsupervised learning and reinforcement learning.

Introduction

Definition of machine learning

Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term “Machine Learning” in 1959 while at IBM. He defined machine learning as “the field of study that gives computers the ability to learn without being explicitly programmed.” However, there is no universally accepted definition for machine learning. Different authors define the term differently. We give below two more definitions.

1. Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both (see [2] p.3).

2. The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience (see [4], Preface.).

Remarks

In the above definitions we have used the term “model” and we will be using this term at several contexts later in this book. It appears that there is no universally accepted one sentence definition of this term. Loosely, it may be understood as some mathematical expression or equation, or some mathematical structures such as graphs and trees, or a division of sets into disjoint subsets, or a set of logical “if . . . then . . . else . . .” rules, or some such thing. It may be noted that this is not an exhaustive list.

Definition of learning

Definition

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

Examples

i) Handwriting recognition learning problem

- Task T: Recognising and classifying handwritten words within images
- Performance P: Percent of words correctly classified
- Training experience E: A dataset of handwritten words with given classifications

ii) A robot driving learning problem

- Task T: Driving on highways using vision sensors
- Performance measure P: Average distance traveled before an error
- training experience: A sequence of images and steering commands recorded while observing a human driver

iii) A chess learning problem

- Task T: Playing chess
- Performance measure P: Percent of games won against opponents
- Training experience E: Playing practice games against itself

Definition

A computer program which learns from experience is called a *machine learning program* or simply a *learning program*. Such a program is sometimes also referred to as a *learner*.

How machines learn

Basic components of learning process

The learning process, whether by a human or a machine, can be divided into four components, namely, data storage, abstraction, generalization and evaluation. Figure 1.1 illustrates the various components and the steps involved in the learning process.

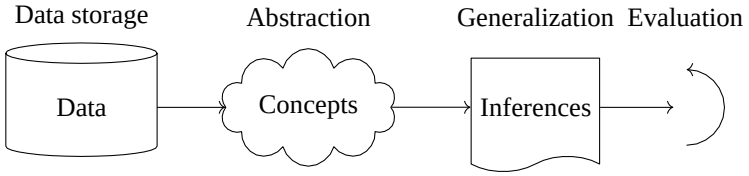


Figure 1.1: Components of learning process

1. **Data storage**

   Facilities for storing and retrieving huge amounts of data are an important component of the learning process. Humans and computers alike utilize data storage as a foundation for advanced reasoning.

   - In a human being, the data is stored in the brain and data is retrieved using electrochemical signals.
   - Computers use hard disk drives, flash memory, random access memory and similar devices to store data and use cables and other technology to retrieve data.

2. **Abstraction**

   The second component of the learning process is known as *abstraction*.

   Abstraction is the process of extracting knowledge about stored data. This involves creating general concepts about the data as a whole. The creation of knowledge involves application of known models and creation of new models.

   The process of fitting a model to a dataset is known as *training*. When the model has been trained, the data is transformed into an abstract form that summarizes the original information.

3. **Generalization**

   The third component of the learning process is known as *generalisation.*

   The term generalization describes the process of turning the knowledge about stored data into a form that can be utilized for future action. These actions are to be carried out on tasks that are similar, but not identical, to those what have been seen before. In generalization, the goal is to discover those properties of the data that will be most relevant to future tasks.

4. **Evaluation**

   Evaluation is the last component of the learning process.

   It is the process of giving feedback to the user to measure the utility of the learned knowledge. This feedback is then utilised to effect improvements in the whole learning process.

Applications of machine learning

Application of machine learning methods to large databases is called data mining. In data mining, a large volume of data is processed to construct a simple model with valuable use, for example, having high predictive accuracy.

The following is a list of some of the typical applications of machine learning.

1. In retail business, machine learning is used to study consumer behaviour.
2. In finance, banks analyze their past data to build models to use in credit applications, fraud detection, and the stock market.
3. In manufacturing, learning models are used for optimization, control, and troubleshooting.
4. In medicine, learning programs are used for medical diagnosis.
5. In telecommunications, call patterns are analyzed for network optimization and maximizing the quality of service.
6. In science, large amounts of data in physics, astronomy, and biology can only be analyzed fast enough by computers. The World Wide Web is huge; it is constantly growing and searching for relevant information cannot be done manually.
7. In artificial intelligence, it is used to teach a system to learn and adapt to changes so that the system designer need not foresee and provide solutions for all possible situations.
8. It is used to find solutions to many problems in vision, speech recognition, and robotics.
9. Machine learning methods are applied in the design of computer-controlled vehicles to steer correctly when driving on a variety of roads.
10. Machine learning methods have been used to develop programmes for playing games such as chess, backgammon and Go.

Understanding data

Since an important component of the machine learning process is data storage, we briefly consider in this section the different types and forms of data that are encountered in the machine learning process.

Unit of observation

By a *unit of observation* we mean the smallest entity with measured properties of interest for a study.

Examples

- A person, an object or a thing
- A time point
- A geographic region
- A measurement

Sometimes, units of observation are combined to form units such as person-years.

Examples and features

Datasets that store the units of observation and their properties can be imagined as collections of data consisting of the following:

- **Examples**

  An "example" is an instance of the unit of observation for which properties have been recorded. An "example" is also referred to as an "instance", or "case" or "record." (It may be noted that the word "example" has been used here in a technical sense.)

- **Features**

  A "feature" is a recorded property or a characteristic of examples. It is also referred to as "attribute", or "variable" or "feature."

Examples for "examples" and "features"

1. **Cancer detection**

   Consider the problem of developing an algorithm for detecting cancer. In this study we note the following.

   (a) The units of observation are the patients.

   (b) The examples are members of a sample of cancer patients.

   (c) The following attributes of the patients may be chosen as the features:

   - gender
   - age
   - blood pressure
   - the findings of the pathology report after a biopsy

2. **Pet selection**

   Suppose we want to predict the type of pet a person will choose.

   (a) The units are the persons.

   (b) The examples are members of a sample of persons who own pets.

features

year	model	price	mileage	color	transmission
2011	SEL	21992	7413	Yellow	AUTO
2011	SEL	20995	10926	Gray	AUTO
2011	SEL	19995	7351	Silver	AUTO
2011	SEL	17809	11613	Gray	AUTO
2012	SE	17500	8367	White	MANUAL
2010	SEL	17495	25125	Silver	AUTO
2011	SEL	17000	27393	Blue	AUTO
2010	SEL	16995	21026	Silver	AUTO
2011	SES	16995	32655	Silver	AUTO

examples

Figure 1.2: Example for "examples" and "features" collected in a matrix format (data relates to automobiles and their features)

(c) The features might include age, home region, family income, etc. of persons who own pets.

3. **Spam e-mail**

   Let it be required to build a learning algorithm to identify spam e-mail.

   (a) The unit of observation could be an e-mail messages.

   (b) The examples would be specific messages.

   (c) The features might consist of the words used in the messages.

Examples and features are generally collected in a "matrix format". Fig. 1.2 shows such a data set.

Different forms of data

1. **Numeric data**

   If a feature represents a characteristic measured in numbers, it is called a numeric feature.

2. **Categorical or nominal**

   A categorical feature is an attribute that can take on one of a limited, and usually fixed, number of possible values on the basis of some qualitative property. A categorical feature is also called a nominal feature.

3. **Ordinal data**

   This denotes a nominal variable with categories falling in an ordered list. Examples include clothing sizes such as small, medium, and large, or a measurement of customer satisfaction on a scale from "not at all happy" to "very happy."

Examples

In the data given in Fig.1.2, the features "year", "price" and "mileage" are numeric and the features "model", "color" and "transmission" are categorical.

General classes of machine learning problems

Learning associations

1. Association rule learning

Association rule learning is a machine learning method for discovering interesting relations, called "association rules", between variables in large databases using some measures of "interestingness".

2. Example

Consider a supermarket chain. The management of the chain is interested in knowing whether there are any patterns in the purchases of products by customers like the following:

> "If a customer buys onions and potatoes together, then he/she is likely to also buy hamburger."

From the standpoint of customer behaviour, this defines an association between the set of products {onion, potato} and the set {burger}. This association is represented in the form of a rule as follows:

$$\{\text{onion, potato}\} \Rightarrow \{\text{burger}\}$$

The measure of how likely a customer, who has bought onion and potato, to buy burger also is given by the conditional probability

$$P(\{\text{onion, potato}\}|\{\text{burger}\})$$

If this conditional probability is 0.8, then the rule may be stated more precisely as follows:

> "80% of customers who buy onion and potato also buy burger."

3. How association rules are made use of

Consider an association rule of the form

$$X \Rightarrow Y,$$

that is, if people buy X then they are also likely to buy Y.

Suppose there is a customer who buys X and does not buy Y. Then that customer is a potential Y customer. Once we find such customers, we can target them for cross-selling. A knowledge of such rules can be used for promotional pricing or product placements.

4. General case

In finding an association rule $X \Rightarrow Y$, we are interested in learning a conditional probability of the form $P(Y|X)$ where Y is the product the customer may buy and X is the product or the set of products the customer has already purchased.

If we may want to make a distinction among customers, we may estimate $P(Y|X, D)$ where D is a set of customer attributes, like gender, age, marital status, and so on, assuming that we have access to this information.

5. Algorithms

There are several algorithms for generating association rules. Some of the well-known algorithms are listed below:

a) Apriori algorithm

b) Eclat algorithm

c) FP-Growth Algorithm (FP stands for Frequency Pattern)

Classification

1. Definition

In machine learning, *classification* is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

2. Example

Consider the following data:

Score1	29	22	10	31	17	33	32	20
Score2	43	29	47	55	18	54	40	41
Result	Pass	Fail	Fail	Pass	Fail	Pass	Pass	Pass

Table 1.1: Example data for a classification problem

Data in Table 1.1 is the training set of data. There are two attributes “Score1” and “Score2”. The class label is called “Result”. The class label has two possible values “Pass” and “Fail”. The data can be divided into two categories or classes: The set of data for which the class label is “Pass” and the set of data for which the class label is“Fail”.

Let us assume that we have no knowledge about the data other than what is given in the table. Now, the problem can be posed as follows: If we have some new data, say “Score1 = 25” and “Score2 = 36”, what value should be assigned to “Result” corresponding to the new data; in other words, to which of the two categories or classes the new observation should be assigned? See Figure 1.3 for a graphical representation of the problem.

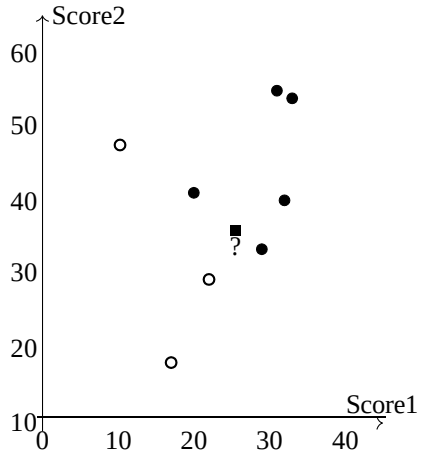


Figure 1.3: Graphical representation of data in Table 1.1. Solid dots represent data in “Pass” class and hollow dots data in “Fail” class. The class label of the square dot is to be determined.

To answer this question, using the given data alone we need to find the rule, or the formula, or the method that has been used in assigning the values to the class label “Result”. The problem of finding this rule or formula or the method is the classification problem. In general, even the general form of the rule or function or method will not be known. So several different rules, etc. may have to be tested to obtain the correct rule or function or method.

3. Real life examples

i) Optical character recognition

Optical character recognition problem, which is the problem of recognizing character codes from their images, is an example of classification problem. This is an example where there are multiple classes, as many as there are characters we would like to recognize. Especially interesting is the case when the characters are handwritten. People have different handwriting styles; characters may be written small or large, slanted, with a pen or pencil, and there are many possible images corresponding to the same character.

ii) Face recognition

In the case of *face recognition*, the input is an image, the classes are people to be recognized, and the learning program should learn to associate the face images to identities. This problem is more difficult than optical character recognition because there are more classes, input image is larger, and a face is three-dimensional and differences in pose and lighting cause significant changes in the image.

iii) Speech recognition

In *speech recognition*, the input is acoustic and the classes are words that can be uttered.

iv) Medical diagnosis

In *medical diagnosis*, the inputs are the relevant information we have about the patient and the classes are the illnesses. The inputs contain the patient's age, gender, past medical history, and current symptoms. Some tests may not have been applied to the patient, and thus these inputs would be missing.

v) Knowledge extraction

Classification rules can also be used for *knowledge extraction*. The rule is a simple model that explains the data, and looking at this model we have an explanation about the process underlying the data.

vi) Compression

Classification rules can be used for *compression*. By fitting a rule to the data, we get an explanation that is simpler than the data, requiring less memory to store and less computation to process.

vii) More examples

Here are some further examples of classification problems.

(a) An emergency room in a hospital measures 17 variables like blood pressure, age, etc. of newly admitted patients. A decision has to be made whether to put the patient in an ICU. Due to the high cost of ICU, only patients who may survive a month or more are given higher priority. Such patients are labeled as "low-risk patients" and others are labeled "high-risk patients". The problem is to device a rule to classify a patient as a "low-risk patient" or a "high-risk patient".

(b) A credit card company receives hundreds of thousands of applications for new cards. The applications contain information regarding several attributes like annual salary, age, etc. The problem is to devise a rule to classify the applicants to those who are credit-worthy, who are not credit-worthy or to those who require further analysis.

(c) Astronomers have been cataloguing distant objects in the sky using digital images created using special devices. The objects are to be labeled as star, galaxy, nebula, etc. The data is highly noisy and are very faint. The problem is to device a rule using which a distant object can be correctly labeled.

4. Discriminant

A *discriminant* of a classification problem is a rule or a function that is used to assign labels to new observations.

Examples

i) Consider the data given in Table 1.1 and the associated classification problem. We may consider the following rules for the classification of the new data:

IF Score1 + Score2 $\geq$ 60, THEN "Pass" ELSE "Fail".
IF Score1 $\geq$ 20 AND Score2 $\geq$ 40 THEN "Pass" ELSE "Fail".

Or, we may consider the following rules with unspecified values for M, m_1, m_2 and then by some method estimate their values.

IF Score1 + Score2 $\geq M$, THEN "Pass" ELSE "Fail".
IF Score1 $\geq m_1$ AND Score2 $\geq m_2$ THEN "Pass" ELSE "Fail".

ii) Consider a finance company which lends money to customers. Before lending money, the company would like to assess the risk associated with the loan. For simplicity, let us assume that the company assesses the risk based on two variables, namely, the annual income and the annual savings of the customers.

Let x_1 be the annual income and x_2 be the annual savings of a customer.

- After using the past data, a rule of the following form with suitable values for θ_1 and θ_2 may be formulated:

IF $x_1 > \theta_1$ AND $x_2 > \theta_2$ THEN "low-risk" ELSE "high-risk".

This rule is an example of a discriminant.

- Based on the past data, a rule of the following form may also be formulated:

IF $x_2 - 0.2x_1 > 0$ THEN "low-risk" ELSE "high-risk".

In this case the rule may be thought of as the discriminant. The function $f(x_1, x_2) = x_2 - 0,2x_1$ can also be considered as the discriminant.

5. Algorithms

There are several machine learning algorithms for classification. The following are some of the well-known algorithms.

a) Logistic regression

b) Naive Bayes algorithm

c) k-NN algorithm

d) Decision tree algorithm

e) Support vector machine algorithm

f) Random forest algorithm

Remarks

- A classification problem requires that examples be classified into one of two or more classes.
- A classification can have real-valued or discrete input variables.
- A problem with two classes is often called a two-class or binary classification problem.
- A problem with more than two classes is often called a multi-class classification problem.
- A problem where an example is assigned multiple classes is called a multi-label classification problem.

Regression

1. Definition

In machine learning, a *regression problem* is the problem of predicting the value of a numeric variable based on observed values of the variable. The value of the output variable may be a number, such as an integer or a floating point value. These are often quantities, such as amounts and sizes. The input variables may be discrete or real-valued.

2. Example

Consider the data on car prices given in Table 1.2.

Price (US$)	Age (years)	Distance (KM)	Weight (pounds)
13500	23	46986	1165
13750	23	72937	1165
13950	24	41711	1165
14950	26	48000	1165
13750	30	38500	1170
12950	32	61000	1170
16900	27	94612	1245
18600	30	75889	1245
21500	27	19700	1185
12950	23	71138	1105

Table 1.2: Prices of used cars: example data for regression

Suppose we are required to estimate the price of a car aged 25 years with distance 53240 KM and weight 1200 pounds. This is an example of a regression problem beause we have to predict the value of the numeric variable “Price”.

3. General approach

Let x denote the set of input variables and y the output variable. In machine learning, the general approach to regression is to assume a model, that is, some mathematical relation between x and y, involving some parameters say, θ, in the following form:

$$y = f(x, \theta)$$

The function $f(x, \theta)$ is called the *regression function*. The machine learning algorithm optimizes the parameters in the set θ such that the approximation error is minimized; that is, the estimates of the values of the dependent variable y are as close as possible to the correct values given in the training set.

Example

For example, if the input variables are "Age", "Distance" and "Weight" and the output variable is "Price", the model may be

$$y = f(x, \theta)$$
$$\text{Price} = a_0 + a_1 \times (\text{Age}) + a_2 \times (\text{Distance}) + a_3 \times (\text{Weight})$$

where $x = (\text{Age}, \text{Distance}, \text{Weight})$ denotes the the set of input variables and $\theta = (a_0, a_1, a_2, a_3)$ denotes the set of parameters of the model.

4. Different regression models

There are various types of regression techniques available to make predictions. These techniques mostly differ in three aspects, namely, the number and type of independent variables, the type of dependent variables and the shape of regression line. Some of these are listed below.

- *Simple linear regression*: There is only one continuous independent variable x and the assumed relation between the independent variable and the dependent variable y is

$$y = a + bx.$$

- *Multivariate linear regression*: There are more than one independent variable, say $x_1, \ldots, x_n$, and the assumed relation between the independent variables and the dependent variable is

$$y = a_0 + a_1 x_1 + \ldots + a_n x_n.$$

- *Polynomial regression*: There is only one continuous independent variable x and the assumed model is

$$y = a_0 + a_1 x + \ldots + a_n x^n.$$

- *Logistic regression*: The dependent variable is binary, that is, a variable which takes only the values 0 and 1. The assumed model involves certain probability distributions.

Different types of learning

In general, machine learning algorithms can be classified into three types.

Supervised learning

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

In supervised learning, each example in the training set is a pair consisting of an input object (typically a vector) and an output value. A supervised learning algorithm analyzes the training data and produces a function, which can be used for mapping new examples. In the optimal case, the function will correctly determine the class labels for unseen instances. Both classification and regression problems are supervised learning problems.

A wide range of supervised learning algorithms are available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems.

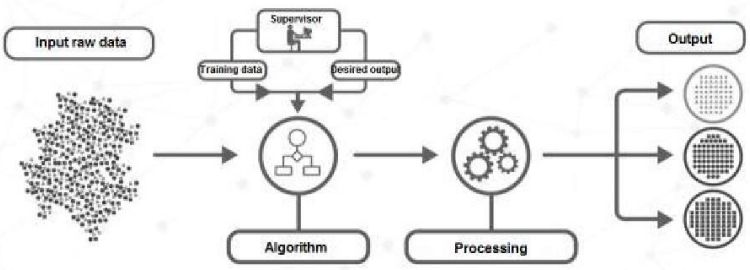


Figure 1.4: Supervised learning

Remarks

A "supervised learning" is so called because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers (that is, the correct outputs), the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Example

Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients and each patient is labeled as "healthy" or "sick".

gender	age	label
M	48	sick
M	67	sick
F	53	healthy
M	49	healthy
F	34	sick
M	21	healthy

Based on this data, when a new patient enters the clinic, how can one predict whether he/she is healthy or sick?

Unsupervised learning

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

In unsupervised learning algorithms, a classification or categorization is not included in the observations. There are no output values and so there is no estimation of functions. Since the examples given to the learner are unlabeled, the accuracy of the structure that is output by the algorithm cannot be evaluated.

The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.

Example

Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients.

gender	age
M	48
M	67
F	53
M	49
F	34
M	21

Based on this data, can we infer anything regarding the patients entering the clinic?

Reinforcement learning

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

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

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

Reinforcement learning is different from supervised learning. Supervised learning is learning from examples provided by a knowledgeable expert.

Sample questions

(a) Short answer questions

1. What is meant by “learning” in the context of machine learning?
2. List out the types of machine learning.
3. Distinguish between classification and regression.
4. What are the differences between supervised and unsupervised learning?
5. What is meant by supervised classification?
6. Explain supervised learning with an example.
7. What do you mean by reinforcement learning?
8. What is an association rule?
9. Explain the concept of Association rule learning. Give the names of two algorithms for generating association rules.
10. What is a classification problem in machine learning. Illustrate with an example.
11. Give three examples of classification problems from real life situations.
12. What is a discriminant in a classification problem?
13. List three machine learning algorithms for solving classification problems.
14. What is a binary classification problem? Explain with an example. Give also an example for a classification problem which is not binary.
15. What is regression problem. What are the different types of regression?

(b) Long answer questions

1. Give a definition of the term “machine learning”. Explain with an example the concept of learning in the context of machine learning.
2. Describe the basic components of the machine learning process.
3. Describe in detail applications of machine learning in any three different knowledge domains.
4. Describe with an example the concept of association rule learning. Explain how it is made use of in real life situations.
5. What is the classification problem in machine learning? Describe three real life situations in different domains where such problems arise.
6. What is meant by a discriminant of a classification problem? Illustrate the idea with examples.
7. Describe in detail with examples the different types of learning like the supervised learning,etc.

Input representation

The general classification problem is concerned with assigning a class label to an unknown instance from instances of known assignments of labels. In a real world problem, a given situation or an object will have large number of features which may contribute to the assignment of the labels. But in practice, not all these features may be equally relevant or important. Only those which are significant need be considered as inputs for assigning the class labels. These features are referred to as the "input features" for the problem. They are also said to constitute an "*input representation*" for the problem.

Example

Consider the problem of assigning the label "family car" or "not family car" to cars. Let us assume that the features that separate a family car from other cars are the price and engine power. These attributes or features constitute the input representation for the problem. While deciding on this input representation, we are ignoring various other attributes like seating capacity or colour as irrelevant.

Hypothesis space

In the following discussions we consider only "binary classification" problems; that is, classification problems with only two class labels. The class labels are usually taken as "1" and "0". The label "1" may indicate "True", or "Yes", or "Pass", or any such label. The label "0" may indicate "False", or "No" or "Fail", or any such label. The examples with class labels 1 are called "positive examples" and examples with labels "0" are called "negative examples".

Definition

1. **Hypothesis**

   In a binary classification problem, a *hypothesis* is a statement or a proposition purporting to explain a given set of facts or observations.

2. Hypothesis space

The *hypothesis space* for a binary classification problem is a set of hypotheses for the problem that might possibly be returned by it.

3. Consistency and satisfying

Let x be an example in a binary classification problem and let $c(x)$ denote the class label assigned to x ($c(x)$ is 1 or 0). Let D be a set of training examples for the problem. Let h be a hypothesis for the problem and $h(x)$ be the class label assigned to x by the hypothesis h.

(a) We say that the hypothesis h is *consistent* with the set of training examples D if $h(x) = c(x)$ for all $x \in D$.

(b) We say that an example x *satisfies* the hypothesis h if $h(x) = 1$.

Examples

1. Consider the set of observations of a variable x with the associated class labels given in Table 2.1:

x	27	15	23	20	25	17	12	30	6	10
Class	1	0	1	1	1	0	0	1	0	0

Table 2.1: Sample data to illustrate the concept of hypotheses

Figure 2.1 shows the data plotted on the x-axis.

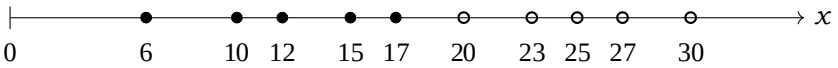


Figure 2.1: Data in Table 2.1 with hollow dots representing positive examples and solid dots representing negative examples

Looking at Figure 2.1, it appears that the class labeling has been done based on the following rule.

$$h' \quad : \quad \text{IF } x \geq 20 \text{ THEN “1” ELSE “0”.} \tag{2.1}$$

Note that h' is consistent with the training examples in Table 2.1. For example, we have:

$$h'(27) = 1, \quad c(27) = 1, \quad h'(27) = c(27)$$
$$h'(15) = 0, \quad c(15) = 0, \quad h'(15) = c(15)$$

Note also that, for $x = 5$ and $x = 28$ (not in training data),

$$h'(5) = 0, \quad h'(28) = 1.$$

The hypothesis h' explains the data. The following proposition also explains the data:

$$h'' \quad : \quad \text{IF } x \geq 19 \text{ THEN “0” ELSE “1”.} \tag{2.2}$$

It is not enough that the hypothesis explains the given data; it must also predict correctly the class label of future observations. So we consider a set of such hypotheses and choose the “best” one. The set of hypotheses can be defined using a parameter, say m, as given below:

$$h_m \quad : \quad \text{IF } x \geq m \text{ THEN “1” ELSE ”0”.} \tag{2.3}$$

The set of all hypotheses obtained by assigning different values to m constitutes the hypothesis space H; that is,

$$H = \{h_m : m \text{ is a real number}\}. \tag{2.4}$$

For the same data, we can have different hypothesis spaces. For example, for the data in Table 2.1, we may also consider the hypothesis space defined by the following proposition:

$$h'_m \quad : \quad \text{IF } x \leq m \text{ THEN "0" ELSE "1".}$$

2. Consider a situation with four binary variables x_1, x_2, x_3, x_4 and one binary output variable y. Suppose we have the following observations.

x_1	x_2	x_3	x_4	y
0	0	0	1	1
0	1	0	1	0
1	1	0	0	1
0	0	1	0	0

The problem is to predict a function f of x_1, x_2, x_3, x_4 which predicts the value of y for any combination of values of x_1, x_2, x_3, x_4. In this problem, the hypothesis space is the set of all possible functions f. It can be shown that the size of the hypothesis space is $2^{(2^4)} = 65536$.

3. Consider the problem of assigning the label "family car" or "not family car" to cars. For convenience, we shall replace the label "family car" by "1" and "not family car" by "0". Suppose we choose the features "price ('000 $)" and "power (hp)" as the input representation for the problem. Further, suppose that there is some reason to believe that for a car to be a family car, its price and power should be in certain ranges. This supposition can be formulated in the form of the following proposition:

$$\text{IF } (p_1 < \text{price} < p_2) \text{ AND } (e_1 < \text{power} < e_2) \text{ THEN "1" ELSE "0"} \tag{2.5}$$

for suitable values of p_1, p_2, e_1 and e_2. Since a solution to the problem is a proposition of the form Eq.(2.5) with specific values for p_1, p_2, e_1 and e_2, the hypothesis space for the problem is the set of all such propositions obtained by assigning all possible values for p_1, p_2, e_1 and e_2.

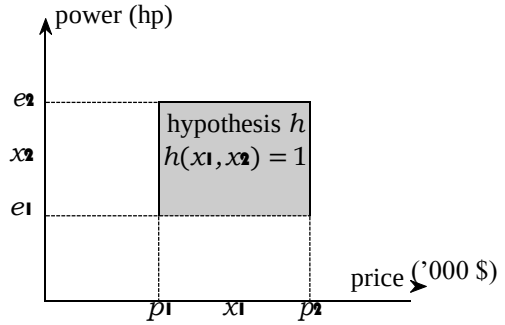


Figure 2.2: An example hypothesis defined by Eq. (2.5)

It is interesting to observe that the set of points in the power–price plane which satisfies the condition

$$(p_1 < \text{price} < p_2) \text{ AND } (e_1 < \text{power} < e_2)$$

defines a rectangular region (minus the boundary) in the price–power space as shown in Figure 2.2. The sides of this rectangular region are parallel to the coordinate axes. Such a rectangle

is called an *axis-aligned rectangle* If h is the hypothesis defined by Eq.(2.5), and x_1, x_2 is any point in the price–power plane, then $h(x_1, x_2) = 1$ if and only if (x_1, x_2) is within the rectangular region. Hence we may identify the hypothesis h with the rectangular region. Thus, the hypothesis space for the problem can be thought of as the set of all axis-aligned rectangles in the price–power plane.

4. Consider the trading agent trying to infer which books or articles the user reads based on keywords supplied in the article. Suppose the learning agent has the following data ("1" indicates "True" and "0" indicates "False"):

article	crime	academic	local	music	reads
a1	true	false	false	true	1
a2	true	false	false	false	1
a3	false	true	false	false	0
a4	false	false	true	false	0
a5	true	true	false	false	1

The aim is to learn which articles the user reads. The aim is to find a definition such as

IF (crime OR (academic AND (NOT music))) THEN "1" ELSE "0".

The hypothesis space H could be all boolean combinations of the input features or could be more restricted, such as conjunctions or propositions defined in terms of fewer than three features.

Ordering of hypotheses

Definition

Let X be the set of all possible examples for a binary classification problem and let h' and h'' be two hypotheses for the problem.

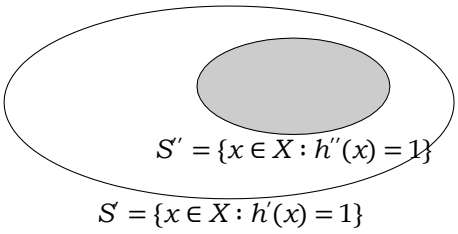


Figure 2.3: Hypothesis h' is more general than hypothesis h'' if and only if $S'' \subseteq S'$

1. We say that h' is *more general than* h'' if and only if for every $x \in X$, if x satisfies h'' then x satisfies h' also; that is, if $h''(x) = 1$ then $h'(x) = 1$ also. The relation "is more general than" defines a partial ordering relation in hypothesis space.

2. We say that h' is *more specific* than h'', if h'' is more general than h'.

3. We say that h' is *strictly more general than* h'' if h' is more general than h'' and h'' is *not* more general than h'.

4. We say that h' is *strictly more specific than* h'' if h' is more specific than h'' and h'' is *not* more specific than h'.

Example

Consider the hypotheses h' and h'' defined in Eqs.(2.1),(2.2). Then it is easy to check that if $h'(x) = 1$ then $h''(x) = 1$ also. So, h'' is more general than h'. But, h' is not more general than h'' and so h'' is strictly more general than h'.

Version space

Definition

Consider a binary classification problem. Let D be a set of training examples and H a hypothesis space for the problem. The *version space* for the problem with respect to the set D and the space H is the set of hypotheses from H consistent with D; that is, it is the set

$$\text{VS}_{D,H} = \{h \in H : h(x) = c(x) \text{ for all } x \in D\}.$$

Examples

Example 1

Consider the data D given in Table 2.1 and the hypothesis space defined by Eqs.(2.3)-(2.4).

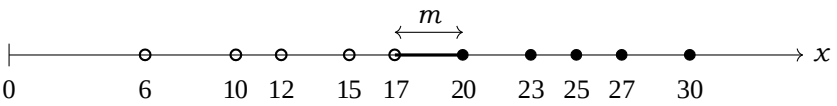


Figure 2.4: Values of m which define the version space with data in Table 2.1 and hypothesis space defined by Eq.(2.4)

From Figure 2.4 we can easily see that the hypothesis space with respect this dataset D and hypothesis space H is as given below:

$$\text{VS}_{D,H} = \{h_m : 17 < m \leq 20\}.$$

Example 2

Consider the problem of assigning the label "family car" (indicated by "1") or "not family car" (indicated by "0") to cars. Given the following examples for the problem and assuming that the hypothesis space is as defined by Eq. (2.5), the version space for the problem.

x_1: Price in '000 ($)	32	82	44	34	43	80	38
x_2: Power (hp)	170	333	220	235	245	315	215
Class	0	0	1	1	1	0	1

x_1	47	27	56	28	20	25	66	75
x_2	260	290	320	305	160	300	250	340
Class	1	0	0	0	0	0	0	0

Solution

Figure 2.5 shows a scatter plot of the given data. In the figure, the data with class label "1" (family car) is shown as hollow circles and the data with class labels "0" (not family car) are shown as solid dots.

A hypothesis as given by Eq.(2.5) with specific values for the parameters p_1, p_2, e_1 and e_2 specifies an axis-aligned rectangle as shown in Figure 2.2. So the hypothesis space for the problem can be thought as the set of axis-aligned rectangles in the price-power plane.

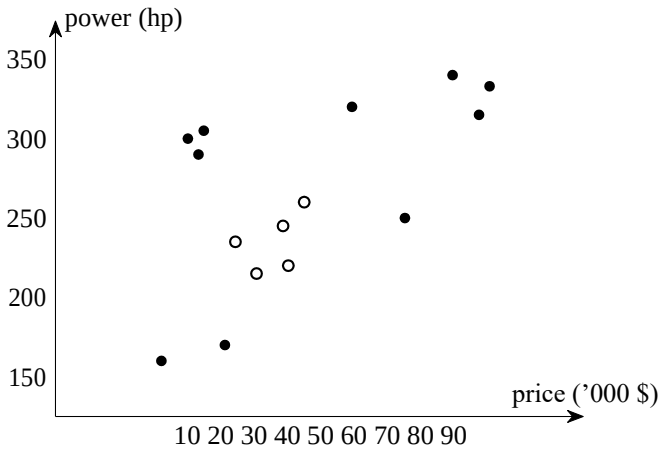


Figure 2.5: Scatter plot of price-power data (hollow circles indicate positive examples and solid dots indicate negative examples)

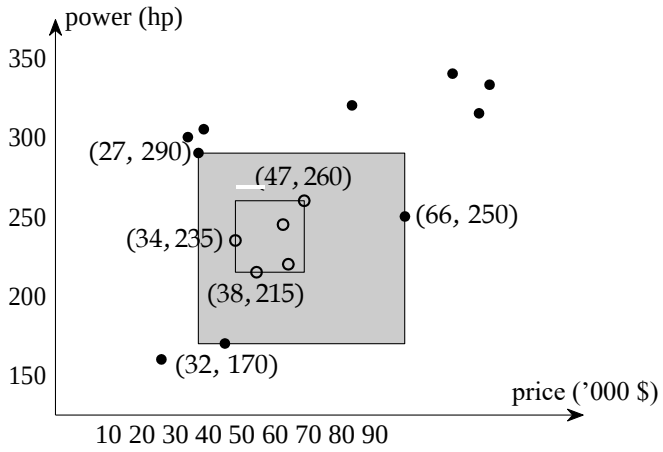


Figure 2.6: The version space consists of hypotheses corresponding to axis-aligned rectangles contained in the shaded region

The version space consists of all hypotheses specified by axis-aligned rectangles contained in the shaded region in Figure 2.6. The inner rectangle is defined by

$$(34 < \text{price} < 47) \text{ AND } (215 < \text{power} < 260)$$

and the outer rectangle is defined by

$$(27 < \text{price} < 66) \text{ AND } (170 < \text{power} < 290).$$

Example 3

Consider the problem of finding a rule for determining days on which one can enjoy water sport. The rule is to depend on a few attributes like "temp", "humidity", etc. Suppose we have the following data to help us devise the rule. In the data, a value of "1" for "enjoy" means "yes" and a value of "0" indicates "no".

Example	sky	temp	humidity	wind	water	forecast	enjoy
1	sunny	warm	normal	strong	warm	same	1
2	sunny	warm	high	strong	warm	same	1
3	rainy	cold	high	strong	warm	change	0
4	sunny	warm	high	strong	cool	change	1

Find the hypothesis space and the version space for the problem. (For a detailed discussion of this problem see [4] Chapter2.)

Solution

We are required to find a rule of the following form, consistent with the data, as a solution of the problem.

$$(\text{sky} = x_1) \wedge (\text{temp} = x_2) \wedge (\text{humidity} = x_3) \wedge$$
$$(\text{wind} = x_4) \wedge (\text{water} = x_5) \wedge (\text{forecast} = x_6) \leftrightarrow \text{yes} \qquad (2.6)$$

where

x_1 = sunny, warm, Y
x_2 = warm, cold, Y
x_3 = normal, high, Y
x_4 = strong, Y
x_5 = warm, cool, Y
x_6 = same, change, Y

(Here a "Y" indicates other possible values of the attributes.) The hypothesis may be represented compactly as a vector

$$(a_1, a_2, a_3, a_4, a_5, a_6)$$

where, in the positions of $a_1, \ldots, a_6$, we write

- a "?" to indicate that any value is acceptable for the corresponding attribute,
- a "$\emptyset$" to indicate that no value is acceptable for the corresponding attribute,
- some specific single required value for the corresponding attribute

For example, the vector

(?, cold, high, ?, ?, ?)

indicates the hypothesis that one enjoys the sport only if "temp" is "cold" and "humidity" is "high" whatever be the values of the other attributes.

It can be shown that the version space for the problem consists of the following six hypotheses only:

(sunny,	warm,	?,	strong,	?,	?)
(sunny,	?,	?,	strong,	?,	?)
(sunny,	warm,	?,	?,	?,	?)
(?,	warm,	?,	strong,	?,	?)
(sunny,	?,	?,	?,	?,	?)
(?,	warm,	?,	?,	?,	?)

Noise

Noise and their sources

Noise is any unwanted anomaly in the data ([2] p.25). Noise may arise due to several factors:

1. There may be imprecision in recording the input attributes, which may shift the data points in the input space.
2. There may be errors in labeling the data points, which may relabel positive instances as negative and vice versa. This is sometimes called teacher noise.
3. There may be additional attributes, which we have not taken into account, that affect the label of an instance. Such attributes may be hidden or latent in that they may be unobservable. The effect of these neglected attributes is thus modeled as a random component and is included in "noise."

Effect of noise

Noise distorts data. When there is noise in data, learning problems may not produce accurate results. Also, simple hypotheses may not be sufficient to explain the data and so complicated hypotheses may have to be formulated. This leads to the use of additional computing resources and the needless wastage of such resources.

For example, in a binary classification problem with two variables, when there is noise, there may not be a simple boundary between the positive and negative instances and to separate them. A rectangle can be defined by four numbers, but to define a more complicated shape one needs a more complex model with a much larger number of parameters. So, when there is noise, we may make a complex model which makes a perfect fit to the data and attain zero error; or, we may use a simple model and allow some error.

Learning multiple classes

So far we have been discussing binary classification problems. In a general case there may be more than two classes. Two methods are generally used to handle such cases. These methods are known by the names "one-against-all" and "one-against-one".

Procedures for learning multiple classes

"One-against all" method

Consider the case where there are K classes denoted by $C_1, \ldots, C_K$. Each input instance belongs to exactly one of them.

We view a K-class classification problem as K two-class problems. In the i-th two-class problem, the training examples belonging to C_i are taken as the positive examples and the examples of all other classes are taken as the negative examples. So, we have to find K hypotheses $h_1, \ldots, h_K$ where h_i is defined by

$$h_i(x) = \begin{cases} 1 & \text{if } x \text{ is in class } C_i \\ 0 & \text{otherwise} \end{cases}$$

For a given x, ideally only one of $h_i(x)$ is 1 and then we assign the class C_i to x. But, when no, or, two or more, $h_i(x)$ is 1, we cannot choose a class. In such a case, we say that the classifier *rejects* such cases.

"One-against-one" method

In the *one-against-one* (OAO) (also called *one-vs-one* (OVO)) strategy, a classifier is constructed for each pair of classes. If there are K different class labels, a total of $(K\ K\ 1)/2$ classifiers are constructed. An unknown instance is classified with the class getting the most votes. Ties are broken arbitrarily.

For example, let there be three classes, A, B and C. In the OVO method we construct $3\ 3\ (-1)/2 = 3$ binary classifiers. Now, if any x is to be classified, we apply each of the three classifiers to x. Let the three classifiers assign the classes A, B, B respectively to x. Since a label to x is assigned by the majority voting, in this example, we assign the class label of B to x.

Model selection

As we have pointed earlier in Section 1.1.1, there is no universally accepted definition of the term "model". It may be understood as some mathematical expression or equation, or some mathematical structures such as graphs and trees, or a division of sets into disjoint subsets, or a set of logical "if . . . then . . . else . . ." rules, or some such thing.

In order to formulate a hypothesis for a problem, we have to choose some model and the term "model selection" has been used to refer to the process of choosing a model. However, the term has been used to indicate several things. In some contexts it has been used to indicates the process of choosing one particular approach from among several different approaches. This may be choosing an appropriate algorithms from a selection of possible algorithms, or choosing the sets of features to be used for input, or choosing initial values for certain parameters. Sometimes "model selection" refers to the process of picking a particular mathematical model from among different mathematical models which all purport to describe the same data set. It has also been described as the process of choosing the right inductive bias.

Inductive bias

In a learning problem we only have the data. But data by itself is not sufficient to find the solution. We should make some extra assumptions to have a solution with the data we have. The set of assumptions we make to have learning possible is called the *inductive bias* of the learning algorithm. One way we introduce inductive bias is when we assume a hypothesis class.

Examples

- In learning the class of family car, there are infinitely many ways of separating the positive examples from the negative examples. Assuming the shape of a rectangle is an inductive bias.
- In regression, assuming a linear function is an inductive bias.

The model selection is about choosing the right inductive bias.

Advantages of a simple model

Even though a complex model may not be making any errors in prediction, there are certain advantages in using a simple model.

1. A simple model is easy to use.
2. A simple model is easy to train. It is likely to have fewer parameters.

   It is easier to find the corner values of a rectangle than the control points of an arbitrary shape.
3. A simple model is easy to explain.

4.A simple model would generalize better than a complex model. This principle is known as Occam's razor, which states that simpler explanations are more plausible and any unnecessary complexity should be shaved off.

Remarks

A model should not be too simple! With a small training set when the training instances differ a little bit, we expect the simpler model to change less than a complex model: A simple model is thus said to have less *variance*. On the other hand, a too simple model assumes more, is more rigid, and may fail if indeed the underlying class is not that simple. A simpler model has more bias. Finding the optimal model corresponds to minimizing both the bias and the variance.

Generalisation

How well a model trained on the training set predicts the right output for new instances is called *generalization*.

Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning. The goal of a good machine learning model is to generalize well from the training data to any data from the problem domain. This allows us to make predictions in the future on data the model has never seen. Overfitting and underfitting are the two biggest causes for poor performance of machine learning algorithms. The model should be selected having the best generalisation. This is said to be the case if these problems are avoided.

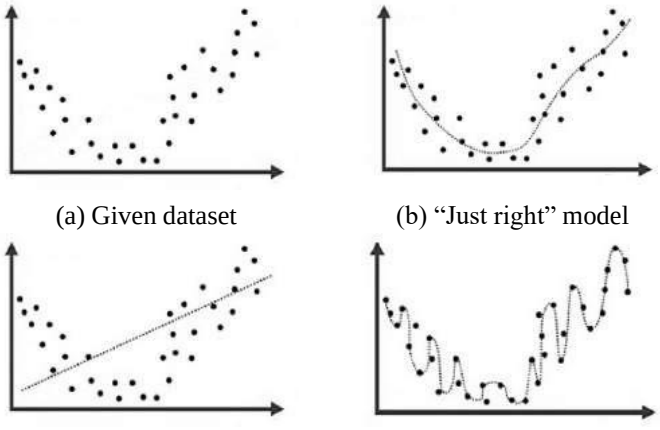
- **Underfitting**

  Underfitting is the production of a machine learning model that is not complex enough to accurately capture relationships between a datasetâĂŹs features and a target variable.

- **Overfitting**

  Overfitting is the production of an analysis which corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.

Example 1



(a) Given dataset (b) "Just right" model

(c) Underfitting model (d) Overfitting model

Figure 2.7: Examples for overfitting and overfitting models

Consider a dataset shown in Figure 2.7(a). Let it be required to fit a regression model to the data. The graph of a model which looks "just right" is shown in Figure 2.7(b). In Figure 2.7(c)we have a linear regression model for the same dataset and this model does seem to capture the essential features of the dataset. So this model suffers from underfitting. In Figure 2.7(d) we have a regression model which corresponds too closely to the given dataset and hence it does not account for small random noises in the dataset. Hence it suffers from overfitting.

Example 2

(a) Underfitting (b) Right fitting (c) Overfitting

Figure 2.8: Fitting a classification boundary

Suppose we have to determine the classification boundary for a dataset two class labels. An example situation is shown in Figure 2.8 where the curved line is the classification boundary. The three figures illustrate the cases of underfitting, right fitting and overfitting.

Testing generalisation: Cross-validation

We can measure the generalization ability of a hypothesis, namely, the quality of its inductive bias, if we have access to data outside the training set. We simulate this by dividing the training set we have into two parts. We use one part for training (that is, to find a hypothesis), and the remaining part is called the *validation set* and is used to test the generalization ability. Assuming large enough training and validation sets, the hypothesis that is the most accurate on the validation set is the best one (the one that has the best inductive bias). This process is called *cross-validation*.

Sample questions

(a) Short answer questions

1. Explain the general-to-specific ordering of hypotheses.
2. In the context of classification problems explain with examples the following: (i) hypothesis (ii) hypothesis space.
3. Define the version space of a binary classification problem.
4. Explain the "one-against-all" method for learning multiple classes.
5. Describe the "one-against-one" method for learning multiple classes.
6. What is meant by inductive bias in machine learning? Give an example.
7. What is meant by overfitting of data? Explain with an example.
8. What is meant by overfitting and underfitting of data with examples.

(b) Long answer questions

1. Define version space and illustrate it with an example.

2. Given the following data

x	0	3	5	9	12	18	23
Label	0	0	0	1	1	1	1

and the hypothesis space

$$H = \{h_m \mid m \text{ a real number}\}$$

where h_m is defined by

$$\text{IF } x \leq m \text{ THEN 1 ELSE 0,}$$

find the version space the problem with respect to D and H.

3. What is meant by "noise" in data? What are its sources and how it is affecting results?

4. Consider the following data:

x	2	3	5	8	10	15	16	18	20
y	12	15	10	6	8	10	7	9	10
Class label	0	0	1	1	1	1	0	0	0

Determine the version space if the hypothesis space consists of all hypotheses of the form

$$\text{IF } (x_1 < x < x_2) \text{ AND } (y_1 < y < y_2) \text{ THEN "1" ELSE "0".}$$

5. For the date in problem 4, what would be the version space if the hypothesis space consists of all hypotheses of the form

$$\text{IF } (x - x_1)^2 + (y - y_1)^2 \leq r^2 \text{ THEN "1" ELSE "0".}$$

6. What issues are to be considered while selecting a model for applying machine learning in a given problem

VC dimension and PAC learning

The concepts of Vapnik-Chervonenkis dimension (VC dimension) and probably approximate correct (PAC) learning are two important concepts in the mathematical theory of learnability and hence are mathematically oriented. The former is a measure of the capacity (complexity, expressive power, richness, or flexibility) of a space of functions that can be learned by a classification algorithm. It was originally defined by Vladimir Vapnik and Alexey Chervonenkis in 1971. The latter is a framework for the mathematical analysis of learning algorithms. The goal is to check whether the probability for a selected hypothesis to be approximately correct is very high. The notion of PAC learning was proposed by Leslie Valiant in 1984.

Vapnik-Chervonenkis dimension

Let H be the hypothesis space for some machine learning problem. The Vapnik-Chervonenkis dimension of H, also called the VC dimension of H, and denoted by $VC(H)$, is a measure of the complexity (or, capacity, expressive power, richness, or flexibility) of the space H. To define the VC dimension we require the notion of the shattering of a set of instances.

Shattering of a set

Let D be a dataset containing N examples for a binary classification problem with class labels 0 and 1. Let H be a hypothesis space for the problem. Each hypothesis h in H partitions D into two disjoint subsets as follows:

$$\{x \in D \mid h(x) = 0\} \text{ and } \{x \in D \mid h(x) = 1\} .$$

Such a partition of S is called a "*dichotomy*" in D. It can be shown that there are 2^N possible dichotomies in D. To each dichotomy of D there is a unique assignment of the labels "1" and "0" to the elements of D. Conversely, if S is any subset of D then, S defines a unique hypothesis h as follows:

$$h(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases}$$

Thus to specify a hypothesis h, we need only specify the set $\{x \in D \mid h(x) = 1\}$.

Figure 3.1 shows all possible dichotomies of D if D has three elements. In the figure, we have shown only one of the two sets in a dichotomy, namely the set $\{x \in D \mid h(x) = 1\}$. The circles and ellipses represent such sets.

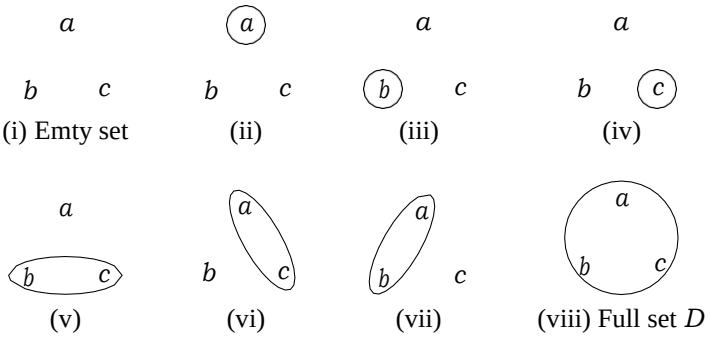


Figure 3.1: Different forms of the set $\{x \in S : h(x) = 1\}$ for $D = \{a, b, c\}$

We require the notion of a hypothesis consistent with a set of examples introduced in Section 2.4 in the following definition.

Definition

A set of examples D is said to be *shattered* by a hypothesis space H if and only if for every dichotomy of D there exists some hypothesis in H consistent with the dichotomy of D.

Vapnik-Chervonenkis dimension

The following example illustrates the concept of Vapnik-Chervonenkis dimension.

Example

Let the instance space X be the set of all real numbers. Consider the hypothesis space defined by Eqs.(2.3)-(2.4):

$$H = \{h_m : m \text{ is a real number}\},$$

where

$$h_m \quad : \quad \text{IF } x \geq m \text{ THEN "1" ELSE "0".}$$

i) Let D be a subset of X containing only a single number, say, $D = \{3.5\}$. There are 2 dichotomies for this set. These correspond to the following assignment of class labels:

x	3.25
Label	0

x	3.25
Label	1

$h_4 \in H$ is consistent with the former dichotomy and $h_3 \in H$ is consistent with the latter. So, to every dichotomy in D there is a hypothesis in H consistent with the dichotomy. Therefore, the set D is shattered by the hypothesis space H.

ii) Let D be a subset of X containing two elements, say, $D = \{3.25, 4.75\}$. There are 4 dichotomies in D and they correspond to the assignment of class labels shown in Table 3.1.

In these dichotomies, h_5 is consistent with (a), h_4 is consistent with (b) and h_3 is consistent with (d). But there is no hypothesis $h_m \in H$ consistent with (c). Thus the two-element set D is not shattered by H. In a similar way it can be shown that there is no two-element subset of X which is shattered by H.

It follows that the size of the largest finite subset of X shattered by H is 1. This number is the VC dimension of H.

x	3.25	4.75
Label	0	0

(a)

x	3.25	4.75
Label	0	1

(b)

x	3.25	4.75
Label	1	0

(c)

x	3.25	4.75
Label	1	1

(d)

Table 3.1: Different assignments of class labels to the elements of {3.25, 4.75}

Definition

The *Vapnik-Chervonenkis dimension* (*VC dimension*) of a hypothesis space H defined over an instance space (that is, the set of all possible examples) X, denoted by $VC(H)$, is the size of the largest finite subset of X shattered by H. If arbitrarily large subsets of X can be shattered by H, then we define $VC(H) = \infty$

Remarks

It can be shown that $VC(H) \leq \log_2(|H|)$ where H is the number of hypotheses in H.

Examples

1. Let X be the set of all real numbers (say, for example, the set of heights of people). For any real numbers a and b define a hypothesis $h_{a,b}$ as follows:

$$h_{a,b}(x) = \begin{cases} 1 & \text{if } a < x < b \\ 0 & \text{otherwise} \end{cases}$$

Let the hypothesis space H consist of all hypotheses of the form $h_{a,b}$. We show that $VC(H) = 2$. We have to show that there is a subset of X of size 2 shattered by H and there is no subset of size 3 shattered by H.

- Consider the two-element set $D = \{3.25, 4.75\}$. The various dichotomies of D are given in Table 3.1. It can be seen that the hypothesis $h_{5,6}$ is consistent with (a), $h_{4,5}$ is consistent with (b), $h_{3,4}$ is consistent with (c) and $h_{3,5}$ is consistent with (d). So the set D is shattered by H.
- Consider a three-element subset $D = \{x_1, x_2, x_3\}$. Let us assume that $x_1 < x_2 < x_3$. H cannot shatter this subset because the dichotomy represented by the set $\{x_1, x_3\}$ cannot be represented by a hypothesis in H (any interval containing both x_1 and x_3 will contain x_2 also).

Therefore, the size of the largest subset of X shattered by H is 2 and so $VC(H) = 2$.

2. Let the instance space X be the set of all points (x, y) in a plane. For any three real numbers, a, b, c define a class labeling as follows:

$$h_{a,b,c}(x,y) = \begin{cases} 1 & \text{if } ax + by + c \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

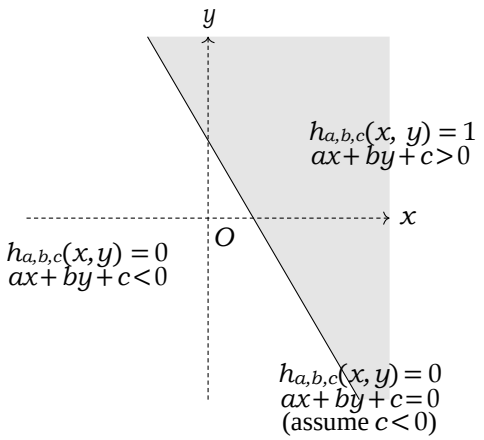


Figure 3.2: Geometrical representation of the hypothesis $h_{a,b,c}$

Let H be the set of all hypotheses of the form $h_{a,b,c}$. We show that $VC(H) = 3$. We have show that there is a subset of size 3 shattered by H and there is no subset of size 4 shattered by H.

- Consider a set $D = \{A, B, C\}$ of three non-collinear points in the plane. There are 8 subsets of D and each of these defines a dichotomy of D. We can easily find 8 hypotheses corresponding to the dichotomies defined by these subsets (see Figure 3.3).

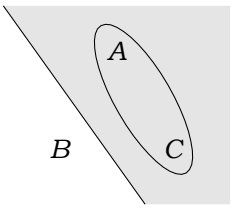


Figure 3.3: A hypothesis $h_{a,b,c}$ consistent with the dichotomy defined by the subset $\{A, C\}$ of $\{A, B, C\}$

- Consider a set $S = \{A, B, C, D\}$ of four points in the plane. Let no three of these points be collinear. Then, the points form a quadrilateral. It can be easily seen that, in this case, there is no hypothesis for which the two element set formed by the ends of a diagonal is the corresponding dichotomy (see Figure 3.4).

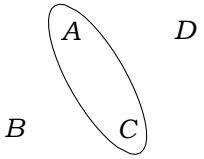


Figure 3.4: There is no hypothesis $h_{a,b,c}$ consistent with the dichotomy defined by the subset $\{A, C\}$ of $\{A, B, C, D\}$

So the set cannot be shattered by H. If any three of them are collinear, then by some trial and error, it can be seen that in this case also the set cannot be shattered by H. No set with four elements cannot be shattered by H.

From the above discussion we conclude that $VC(H) = 3$.

3. Let X be set of all conjunctions of n boolean literals. Let the hypothesis space H consists of conjunctions of up to n literals. It can be shown that $VC(H) = n$. (The full details of the proof of this is beyond the scope of these notes.)

MODULE 2

Probably approximately correct learning

In computer science, *computational learning theory* (or just *learning theory*) is a subfield of artificial intelligence devoted to studying the design and analysis of machine learning algorithms. In computational learning theory, *probably approximately correct learning* (PAC learning) is a framework for mathematical analysis of machine learning algorithms. It was proposed in 1984 by Leslie Valiant.

In this framework, the learner (that is, the algorithm) receives samples and must select a hypothesis from a certain class of hypotheses. The goal is that, with high probability (the "probably" part), the selected hypothesis will have low generalization error (the "approximately correct" part).

In this section we first give an informal definition of PAC-learnability. After introducing a few nore notions, we give a more formal, mathematically oriented, definition of PAC-learnability. At the end, we mention one of the applications of PAC-learnability.

PAC-learnability

To define PAC-learnability we require some specific terminology and related notations.

- Let X be a set called the *instance space* which may be finite or infinite. For example, X may be the set of all points in a plane.
- A *concept class* C for X is a family of functions $c: X \to \{0, 1\}$. A member of C is called a *concept*. A concept can also be thought of as a subset of X. If C is a subset of X, it defines a unique function $\mu_C : X \to \{0, 1\}$ as follows:

$$\mu_C(x) = \begin{cases} 1 \text{ if } x \in C \\ 0 \text{ otherwise} \end{cases}$$

- A *hypothesis* h is also a function $h: X \to \{0, 1\}$. So, as in the case of concepts, a hypothesis can also be thought of as a subset of X. H will denote a set of hypotheses.
- We assume that F is an arbitrary, but fixed, *probability distribution* over X.
- Training examples are obtained by taking random samples from X. We assume that the samples are randomly generated from X according to the probability distribution F.

Now, we give below an informal definition of PAC-learnability.

Definition (informal)

Let X be an instance space, C a concept class for X, h a hypothesis in C and F an arbitrary, but fixed, probability distribution. The concept class C is said to be *PAC-learnable* if there is an algorithm A which, for samples drawn with any probability distribution F and any concept $c \in C$, will with high probability produce a hypothesis $h \in C$ whose error is small.

Additional notions

- **True error**

  To formally define PAC-learnability, we require the concept of the true error of a hypothesis h with respect to a target concept c denoted by $\text{error}_F(h)$. It is defined by

$$\text{error}_F(h) = P_{x \in F}(h(x) \neq c(x))$$

  where the notation $P_{x \in F}$ indicates that the probability is taken for x drawn from X according to the distribution F. This error is the probability that h will misclassify an instance x drawn at random from X according to the distribution F. This error is not directly observable to the learner; it can only see the training error of each hypothesis (that is, how often $h(x) \neq c(x)$ over training instances).

- **Length or dimension of an instance**

  We require the notion of the length or dimension or size of an instance in the instance space X. If the instance space X is the n-dimensional Euclidean space, then each example is specified by n real numbers and so the length of the examples may be taken as n. Similarly, if X is the space of the conjunctions of n Boolean literals, then the length of the examples may be taken as n. These are the commonly considered instance spaces in computational learning theory.

- **Size of a concept**

  We need the notion of the size of a concept c. For any concept c, we define size(c) to be the size of the smallest representation of c using some finite alphabet Σ.

(For a detailed discussion of these and related ideas, see [6] pp.7-15.)

Definition ([4] p.206)

Consider a concept class C defined over a set of instances X of length n and a learner (algorithm) L using hypothesis space H. C is said to be PAC-learnable by L using H if for all $c \in C$, distribution F over X, s such that $0 < s < 1/2$ and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1 - \delta)$ output a hypothesis h such that error$_F(h) \leq s$, in time that is polynomial in $1/s$, $1/\delta$, n and size(c).

Examples

To illustrate the definition of PAC-learnability, let us consider some concrete examples.

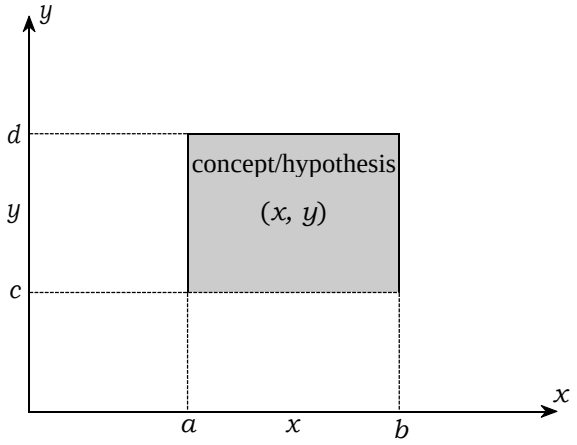


Figure 3.5: An axis-aligned rectangle in the Euclidean plane

Example 1

- Let the instance space be the set X of all points in the Euclidean plane. Each point is represented by its coordinates (x, y) So, the dimension or length of the instances is 2.

- Let the concept class C be the set of all "axis-aligned rectangles" in the plane; that is, the set of all rectangles whose sides are parallel to the coordinate axes in the plane (see Figure 3.5).

- Since an axis-aligned rectangle can be defined by a set of inequalities of the following form having four parameters

$$a \leq x \leq b, \quad c \leq y \leq d$$

  the size of a concept is 4.

- We take the set H of all hypotheses to be equal to the set C of concepts, $H = C$.

- Given a set of sample points labeled positive or negative, let L be the algorithm which outputs the hypothesis defined by the axis-aligned rectangle which gives the tightest fit to the positive examples (that is, that rectangle with the smallest area that includes all of the positive examples and none of the negative examples) (see Figure 3.6).

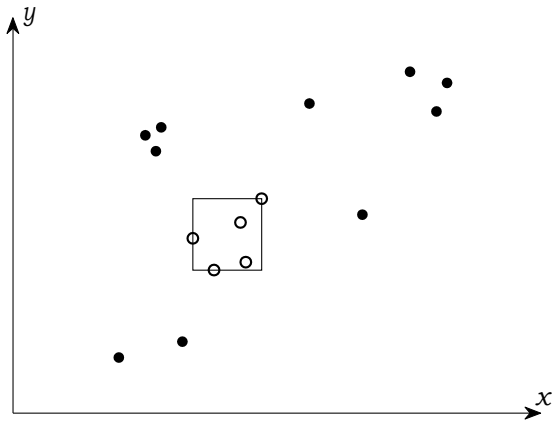


Figure 3.6: Axis-aligned rectangle which gives the tightest fit to the positive examples

It can be shown that, in the notations introduced above, the concept class C is PAC-learnable by the algorithm L using the hypothesis space H of all axis-aligned rectangles.

Example 2

- Let X the set of all n-bit strings. Each n-bit string may be represented by an ordered n-tuple $(a_1, \ldots, a_n)$ where each a_i is either 0 or 1. This may be thought of as an assignment of 0 or 1 to n boolean variables $x_1, \ldots, x_n$. The set X is sometimes denoted by $\{0,1\}^n$.

- To define the concept class, we distinguish certain subsets of X in a special way. By a literal we mean, a Boolean variable x_i or its negation $\neg x_i$. We consider conjunctions of literals over $x_1, \ldots, x_n$. Each conjunction defines a subset of X. for example, the conjunction $x_1 \wedge \neg x_2 \wedge x_4$ defines the following subset of X:

$$\{a = (a_1, \ldots, a_n) \in X \mid a_1 = 1, a_2 = 0, a_4 = 1\}$$

  The concept class C consists of all subsets of X defined by conjunctions of Boolean literals over $x_1, \ldots, x_n$.

- The hypothesis class H is defined as equal to the concept class C.

- Let L be a certain algorithm called "Find-S algorithm" used to find a most specific hypothesis (see [4] p.26).

The concept class C of all subsets of $X = \{0, 1\}^n$ defined by conjunctions of Boolean literals over $x_1, \ldots, x_n$ is PAC-learnable by the Find-S algorithm using the hypothesis space $H = C$.

Applications

To make the discussions complete, we introduce one simple application of the PAC-learning theory. The application is the derivation of a mathematical expression to estimate the size of samples that would produce a hypothesis with a given high probability and which has a generalization error of given low probability.

We use the following assumptions and notations:

- We assume that the hypothesis space H is *finite*. Let $|H|$ denote the number of elements in H.

- We assume that the concept class C be equal to H.
- Let m be the number of elements in the set of samples.
- Let s and δ be such that $0 < s, \delta < 1$.
- The algorithm can be any *consistent algorithm*, that is, any algorithm which correctly classifies the training examples.

It can be shown that, if m is chosen such that

$$m \geq \frac{1}{s}(\ln(|H|) + \ln(1/\delta))$$

then any consistent algorithm will successfully produce any concept in H with probability $1(\delta -)$ and with an error having a maximum probability of s.

Sample questions

(a) Short answer questions

1. What is VC dimension?
2. Explain Vapnik-Chervonenkis dimension.
3. Give an informal definition of PAC learnability.
4. Give a precise definition of PAC learnability.
5. Give an application of PAC learnable algorithm.

(b) Long answer questions

1. Let X be the set of all real numbers. Describe a hypothesis for X for which the VC dimension is 0.
2. Let X be the set of all real numbers. Describe a hypothesis for X for which the VC dimension is 1.
3. Let X be the set of all real numbers. Describe a hypothesis for X for which the VC dimension is 2. Describe an example for which the VC dimension is 3.
4. Describe an example of a PAC learnable concept class.
5. An open interval in R is defined as $(a, b) = \{x \in R \mid a < x < b\}$. It has two parameters a and b. Show that the sets of all open intervals has a VC dimension of 2.

Dimensionality reduction

The complexity of any classifier or regressor depends on the number of inputs. This determines both the time and space complexity and the necessary number of training examples to train such a classifier or regressor. In this chapter, we discuss various methods for decreasing input dimensionality without losing accuracy.

Introduction

In many learning problems, the datasets have large number of variables. Sometimes, the number of variables is more than the number of observations. For example, such situations have arisen in many scientific fields such as image processing, mass spectrometry, time series analysis, internet search engines, and automatic text analysis among others. Statistical and machine learning methods have some difficulty when dealing with such high-dimensional data. Normally the number of input variables is reduced before the machine learning algorithms can be successfully applied.

In statistical and machine learning, dimensionality reduction or dimension reduction is the process of reducing the number of variables under consideration by obtaining a smaller set of principal variables.

Dimensionality reduction may be implemented in two ways.

- **Feature selection**

  In feature selection, we are interested in finding k of the total of n features that give us the most information and we discard the other $(n-k)$ dimensions. We are going to discuss *subset selection* as a feature selection method.

- **Feature extraction**

  In feature extraction, we are interested in finding a new set of k features that are the combination of the original n features. These methods may be supervised or unsupervised depending on whether or not they use the output information. The best known and most widely used feature extraction methods are *Principal Components Analysis* (PCA) and *Linear Discriminant Analysis* (LDA), which are both linear projection methods, unsupervised and supervised respectively.

Measures of error

In both methods we require a measure of the error in the model.

- In regression problems, we may use the *Mean Squared Error* (MSE) or the *Root Mean Squared Error* (RMSE) as the measure of error. MSE is the sum, over all the data points, of the square of the difference between the predicted and actual target variables, divided by

the number of data points. If $y_1, \ldots, y_n$ are the observed values and $\hat{y}_i, \ldots, \hat{y}_n$ are the predicted values, then

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

- In classification problems, we may use the *misclassification rate* as a measure of the error. This is defined as follows:

$$\text{misclassification rate} = \frac{\text{no. of misclassified examples}}{\text{total no. of examples}}$$

Why dimensionality reduction is useful

There are several reasons why we are interested in reducing dimensionality.

- In most learning algorithms, the complexity depends on the number of input dimensions, d, as well as on the size of the data sample, N, and for reduced memory and computation, we are interested in reducing the dimensionality of the problem. Decreasing d also decreases the complexity of the inference algorithm during testing.
- When an input is decided to be unnecessary, we save the cost of extracting it.
- Simpler models are more robust on small datasets. Simpler models have less variance, that is, they vary less depending on the particulars of a sample, including noise, outliers, and so forth.
- When data can be explained with fewer features, we get a better idea about the process that underlies the data, which allows knowledge extraction.
- When data can be represented in a few dimensions without loss of information, it can be plotted and analyzed visually for structure and outliers.

Subset selection

In machine learning *subset selection*, sometimes also called *feature selection*, or *variable selection*, or *attribute selection*, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.

Feature selection techniques are used for four reasons:

- simplification of models to make them easier to interpret by researchers/users
- shorter training times,
- to avoid the curse of dimensionality
- enhanced generalization by reducing overfitting

The central premise when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information.

There are several approaches to subset selection. In these notes, we discuss two of the simplest approaches known as forward selection and backward selection methods.

Forward selection

In *forward selection*, we start with no variables and add them one by one, at each step adding the one that decreases the error the most, until any further addition does not decrease the error (or decreases it only sightly).

Procedure

We use the following notations:

n	:	number of input variables
$x_1, \ldots, x_n$	:	input variables
F_i	:	a subset of the set of input variables
$E(F_i)$	:	error incurred on the validation sample when only the inputs in F_i are used

1. Set $F_0 = \emptyset$ and $E(F_0) = \infty$.
2. For $i = 0, 1, \ldots$, repeat the following until $E(F_{i+1}) \geq E(F_i)$:
   (a) For all possible input variables x_j, train the model with the input variables $F_i \cup \{x_j\}$ and calculate $E(F_i \cup \{x_j\})$ on the validation set.
   (b) Choose that input variable x_m that causes the least error $E(F_i \cup \{x_j\})$:
   $$m = \arg\min_j E(F_i \cup \{x_j\})$$
   (c) Set $F_{i+1} = F_i \cup \{x_m\}$.
3. The set F_i is outputted as the best subset.

Remarks

1. In this procedure, we stop if adding any feature does not decrease the error E. We may even decide to stop earlier if the decrease in error is too small, where there is a user-defined threshold that depends on the application constraints.
2. This process may be costly because to decrease the dimensions from n to k, we need to train and test the system
   $$n + (n-1) + (n-2) + \ldots + (n-k)$$
   times, which is $O(n^2)$.

Backward selection

In sequential backward selection, we start with the set containing all features and at each step remove the one feature that causes the least error.

Procedure

We use the following notations:

n	:	number of input variables
$x_1, \ldots, x_n$	:	input variables
F_i	:	a subset of the set of input variables
$E(F_i)$	:	error incurred on the validation sample when only the inputs in F_i are used

1. Set $F_0 = \{x_1, \ldots, x_n\}$ and $E(F_0) = \infty$.
2. For $i = 0, 1, \ldots$, repeat the following until $E(F_{i+1}) \geq E(F_i)$:
   (a) For all possible input variables x_j, train the model with the input variables $F_i - \{x_j\}$ and calculate $E(F_i - \{x_j\})$ on the validation set.
   (b) Choose that input variable x_m that causes the least error $E(F_i - \{x_j\})$:
   $$m = \arg\min_j E(F_i - \{x_j\})$$

(c) Set $F_i = F_i - \{x_m\}$.

3. The set F_i is outputted as the best subset.

Principal component analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

Graphical illustration of the idea

Consider a two-dimensional data, that is, a dataset consisting of examples having two features. Let each of the features be numeric data. So, each example can be plotted on a coordinate plane (x-coordinate indicating the first feature and y-coordinate indicating the second feature). Plotting the example, we get a scatter diagram of the data. Now let us examine some typical scatter diagram and make some observations regarding the directions in which the points in the scatter diagram are spread out.

Let us examine the figures in Figure 4.1.

(i) Figure 4.1a shows a scatter diagram of a two-dimensional data.

(ii) Figure 4.1b shows spread of the data in the x direction and Figure 4.1c shows the spread of the data in the y-direction. We note that the spread in the x-direction is more than the spread in the y direction.

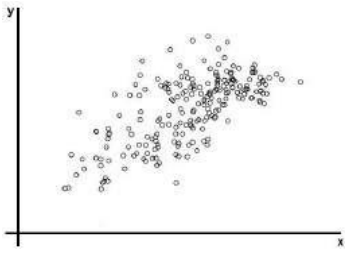
(iii) Examining Figures 4.1d and 4.1e, we note that the maximum spread occurs in the direction shown in Figure 4.1e. Figure 4.1e also shows the point whose coordinates are the mean values of the two features in the dataset. This direction is called the *direction of the first principal component* of the given dataset.

(iv) The direction which is perpendicular (orthogonal) to the direction of the first principal component is called the *direction of the second principal component* of the dataset. This direction is shown in Figure 4.1f. (This is only with reference to a two-dimensional dataset.)

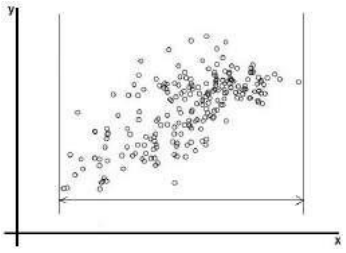
(v) The unit vectors along the directions of principal components are called the *principal component vectors*, or simply, *principal components*. These are shown in Figure 4.1g.

Remark

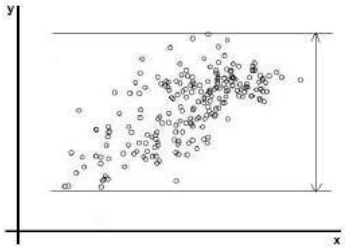
let us consider a dataset consisting of examples with three or more features. In such a case, we have an n-dimensional dataset with $n \geq 3$. In this case, the first principal component is defined exactly as in item iii above. But, for the second component, it may be noted that there would be many directions perpendicular to the direction of the first principal component. The direction of the second principal component is that direction, which is perpendicular to the first principal component, in which the spread of data is largest. The third and higher order principal components are constructed in a similar way.



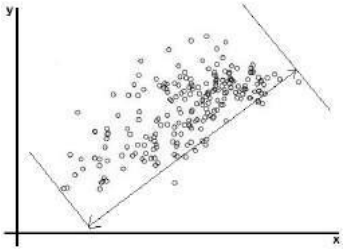
(a) Scatter diagram



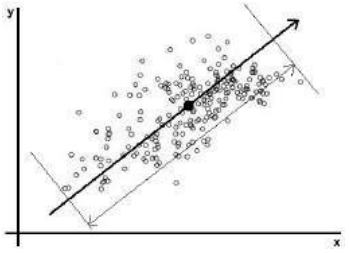
(b) Spread along x-direction



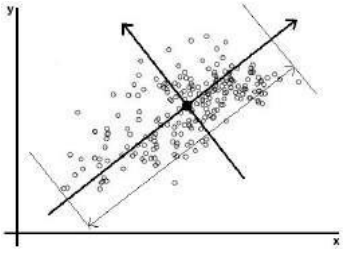
(c) Spread along y-direction



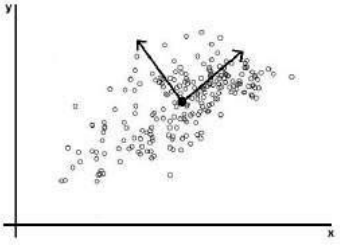
(d) Largest spread



(e) Direction of largest spread : Direction of the first principal component (solid dot is the point whose coordinates are the means of x and y)



(f) Directions of principal components



(g) Principal component vectors (unit vectors in the directions of principal components)

Figure 4.1: Principal components

A warning!

The graphical illustration of the idea of PCA as explained above is slightly misleading. For the sake of simplicity and easy geometrical representation, in the graphical illustration we have used *range* as the measure of spread. The direction of the first principal component was taken as the direction of maximum range. But, due to theoretical reasons, in the implementation of PCA in practice, it is the variance that is taken as as the measure of spread. The first principal component is the the direction in which the variance is maximum.

Computation of the principal component vectors (PCA algorithm)

The following is an outline of the procedure for performing a principal component analysis on a given data. The procedure is heavily dependent on mathematical concepts. A knowledge of these concepts is essential to carry out this procedure.

Step 1.Data

We consider a dataset having n features or variables denoted by $X_1, X_2, \ldots, X_n$. Let there be N examples. Let the values of the i-th feature X_i be $X_{i1}, X_{i2}, \ldots, X_{iN}$ (see Table 4.1).

Features	Example 1	Example 2	...	Example N
X_1	X_{11}	X_{12}		X_{1N}
X_2	X_{21}	X_{22}	...	X_{2N}
$\vdots$				
X_i	X_{i1}	X_{i2}	...	X_{iN}
$\vdots$				
X_n	X_{n1}	X_{n2}	...	X_{nN}

Table 4.1: Data for PCA algorithm

Step 2.Compute the means of the variables

We compute the mean $\bar{X}_i$ of the variable X_i:

$$\bar{X}_i = \frac{1}{N}(X_{i1} + X_{i2} + \ldots + X_{iN}).$$

Step 3.Calculate the covariance matrix

Consider the variables X_i and X_j (i and j need not be different). The covariance of the ordered pair (X_i, X_j) is defined as[1]

$$\text{Cov}(X_i, X_j) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{ik} - \bar{X}_i)(X_{jk} - \bar{X}_j). \tag{4.1}$$

We calculate the following $n \times n$ matrix S called the covariance matrix of the data. The element in the i-th row j-th column is the covariance $\text{Cov}(X_i, X_j)$:

$$S = \begin{pmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \ldots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) & \ldots & \text{Cov}(X_2, X_n) \\ \vdots & & & \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \ldots & \text{Cov}(X_n, X_n) \end{pmatrix}$$

Step 4.Calculate the eigenvalues and eigenvectors of the covariance matrix

Let S be the covariance matrix and let I be the identity matrix having the same dimension as the dimension of S.

i) Set up the equation:

$$\det(S - \lambda I) = 0. \tag{4.2}$$

This is a polynomial equation of degree n in λ. It has n real roots (some of the roots may be repeated) and these roots are the eigenvalues of S. We find the n roots $\lambda_1, \lambda_2, \ldots, \lambda_n$ of Eq. (4.2).

[1]There is an alternative definition of covariance. In this definition, covariance is defined as in Eq. (4.1) with $N-1$ replaced by N. There are certain theoretical reasons for adopting the definition as given here.

ii) If $\lambda = \lambda'$ is an eigenvalue, then the corresponding eigenvector is a vector

$$U = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

such that

$$(S - \lambda' I)U = 0.$$

(This is a system of n homogeneous linear equations in u_1, u_2, . . ., u_n and it always has a nontrivial solution.) We next find a set of n orthogonal eigenvectors $U_1, U_2, \ldots, U_n$ such that U_i is an eigenvector corresponding to λ_i.[2]

iii) We now normalise the eigenvectors. Given any vector X we normalise it by dividing X by its length. The length (or, the norm) of the vector

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

is defined as

$$||X|| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}.$$

Given any eigenvector U, the corresponding normalised eigenvector is computed as

$$\frac{1}{||U||} U.$$

We compute the n normalised eigenvectors $e_1, e_2, \ldots, e_n$ by

$$e_i = \frac{1}{||U_i||} U_i, \quad i = 1, 2, \ldots, n.$$

Step 5.Derive new data set

Order the eigenvalues from highest to lowest. The unit eigenvector corresponding to the largest eigenvalue is the first principal component. The unit eigenvector corresponding to the next highest eigenvalue is the second principal component, and so on.

i) Let the eigenvalues in descending order be $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and let the corresponding unit eigenvectors be $e_1, e_2, \ldots, e_n$.

ii) Choose a positive integer p such that $1 \leq p \leq n$.

iii) Choose the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ and form the following $p \times n$ matrix (we write the eigenvectors as row vectors):

$$F = \begin{pmatrix} e_1^T \\ e_2^T \\ \vdots \\ e_p^T \end{pmatrix},$$

where T in the superscript denotes the transpose.

[2]For $i \neq j$, the vectors U_i and U_j are orthogonal means $U_i^T U_j = 0$ where T denotes the transpose.

iv) We form the following $n \times N$ matrix:

$$X = \begin{pmatrix} X_{11} - \bar{X}_1 & X_{12} - \bar{X}_1 & \ldots & X_{1N} - \bar{X}_1 \\ X_{21} - \bar{X}_2 & X_{22} - \bar{X}_2 & \ldots & X_{2N} - \bar{X}_2 \\ \vdots & & & \\ X_{n1} - \bar{X}_n & X_{n2} - \bar{X}_n & \ldots & X_{nN} - \bar{X}_n \end{pmatrix}$$

v) Next compute the matrix:

$$X_{\text{new}} = FX.$$

Note that this is a $p \times N$ matrix. This gives us a dataset of N samples having p features.

Step 6.New dataset

The matrix X_{new} is the new dataset. Each row of this matrix represents the values of a feature. Since there are only p rows, the new dataset has only features.

Step 7.Conclusion

This is how the principal component analysis helps us in dimensional reduction of the dataset. Note that it is not possible to get back the original n-dimensional dataset from the new dataset.

Illustrative example

We illustrate the ideas of principal component analysis by considering a toy example. In the discussions below, all the details of the computations are given. This is to give the reader an idea of the complexity of computations and also to help the reader do a "worked example" by hand computations without recourse to software packages.

Problem

Given the data in Table 4.2, use PCA to reduce the dimension from 2 to 1.

Feature	Example 1	Example 2	Example 3	Example 4
X_1	4	8	13	7
X_2	11	4	5	14

Table 4.2: Data for illustrating PCA

Solution

1. Scatter plot of data

We have

$$\bar{X} = (\tfrac{1}{4} + 8 + 13 + 7) = 8,$$
$$\bar{X} = (\tfrac{1}{4}1 + 4 + 5 + 14) = 8.5.$$

Figure 4.2 shows the scatter plot of the data together with the point $(\bar{X}_1, \bar{X}_2)$.

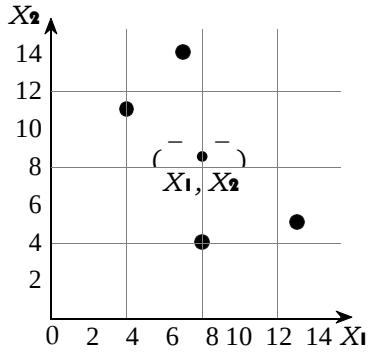


Figure 4.2: Scatter plot of data in Table 4.2

2. Calculation of the covariance matrix

The covariances are calculated as follows:

$$\text{Cov}(X_1, X_1) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)^2$$

$$= \frac{1}{3}((4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2)$$

$$= 14$$

$$\text{Cov}(X_1, X_2) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{1k} - \bar{X}_1)(X_{2k} - \bar{X}_2)$$

$$= \frac{1}{3}((4-8)(11-8.5) + (8-8)(4-8.5) + (13-8)(5-8.5) + (7-8)(14-8.5))$$

$$= -11$$

$$\text{Cov}(X_2, X_1) = \text{Cov}(X_1, X_2)$$

$$= -11$$

$$\text{Cov}(X_2, X_2) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{2k} - \bar{X}_2)^2$$

$$= \frac{1}{3}((11-8.5)^2 + (4-8.5)^2 + (5-8.5)^2 + (14-8.5)^2)$$

$$= 23$$

The covariance matrix is

$$S = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) \end{bmatrix}$$

$$= \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

3. Eigenvalues of the covariance matrix

The characteristic equation of the covariance matrix is

$$0 = \det(S - \lambda I)$$

$$= \begin{vmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{vmatrix}$$

$$= (14-\lambda)(23-\lambda) - (-11) \times (-11)$$

$$= \lambda^2 - 37\lambda + 201$$

Solving the characteristic equation we get

$$\lambda = \frac{1}{2}(37 \pm \sqrt{565})$$
$$= 30.3849, 6.6151$$
$$= \lambda_1, \lambda_2 \quad \text{(say)}$$

4. Computation of the eigenvectors

To find the first principal components, we need only compute the eigenvector corresponding to the largest eigenvalue. In the present example, the largest eigenvalue is λ_1 and so we compute the eigenvector corresponding to λ_1.

The eigenvector corresponding to $\lambda = \lambda_1$ is a vector $U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ satisfying the following equation:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = (S - \lambda_1 I)X$$
$$= \begin{bmatrix} 14 - \lambda_1 & -11 \\ -11 & 23 - \lambda_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$
$$= \begin{bmatrix} (14 - \lambda_1)u_1 - 11u_2 \\ -11u_1 + (23 - \lambda_1)u_1 \end{bmatrix}$$

This is equivalent to the following two equations:

$$(14 - \lambda_1)u_1 - 11u_2 = 0$$
$$-11u_1 + (23 - \lambda_1)u_2 = 0$$

Using the theory of systems of linear equations, we note that these equations are not independent and solutions are given by

$$\frac{u_1}{11} = \frac{u_2}{14 - \lambda_1} = t,$$

that is

$$u_1 = 11t, \quad u_2 = (14 - \lambda_1 t,)$$

where t is any real number. Taking $t = 1$, we get an eigenvector corresponding to λ_1 as

$$U_1 = \begin{bmatrix} 11 \\ 14 - \lambda_1 \end{bmatrix}.$$

To find a unit eigenvector, we compute the length of X_1 which is given by

$$||U_1|| = \sqrt{11^2 + (14 - \lambda_1)^2}$$
$$= \sqrt{11^2 + (14 - 30.3849)^2}$$
$$= 19.7348$$

Therefore, a unit eigenvector corresponding to $lambda_1$ is

$$e_1 = \begin{bmatrix} 11/||U_1|| \\ (14 - \lambda_1)/||U_1|| \end{bmatrix}$$
$$= \begin{bmatrix} 11/19.7348 \\ (14 - 30.3849)/19.7348 \end{bmatrix}$$
$$= \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

By carrying out similar computations, the unit eigenvector e_2 corresponding to the eigenvalue $\lambda = \lambda_2$ can be shown to be

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}.$$

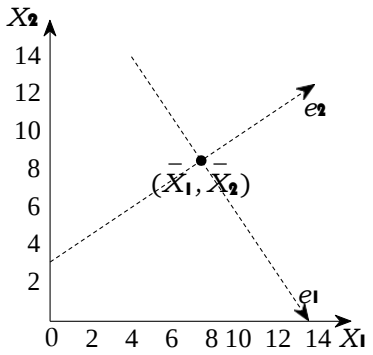


Figure 4.3: Coordinate system for principal components

5. Computation of first principal components

Let $\begin{bmatrix} X_{1k} \\ X_{2k} \end{bmatrix}$ be the k-th sample in Table 4.2. The first principal component of this example is given by (here "T" denotes the transpose of the matrix)

$$e_1^T \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix} = [0.5574 \quad -0.8303] \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix}$$
$$= 0.5574(X_{1k} - \bar{X}_1) - 0.8303(X_{2k} - \bar{X}_2).$$

For example, the first principal component corresponding to the first example $\begin{bmatrix} X_{11} \\ X_{21} \end{bmatrix} = \begin{bmatrix} 4 \\ 11 \end{bmatrix}$ is calculated as follows:

$$[0.5574 \quad -0.8303] \begin{bmatrix} X_{11} - \bar{X}_1 \\ X_{21} - \bar{X}_2 \end{bmatrix} = 0.5574(X_{11} - \bar{X}_1) - 0.8303(X_{21} - \bar{X}_2)$$
$$= 0.5574(4 - 8) - 0.8303(11 - 8, 5)$$
$$= -4.30535$$

The results of calculations are summarised in Table 4.3.

X_1	4	8	13	7
X_2	11	4	5	14
First principal components	-4.3052	3.7361	5.6928	-5.1238

Table 4.3: First principal components for data in Table 4.2

6. Geometrical meaning of first principal components

As we have seen in Figure 4.1, we introduce new coordinate axes. First we shift the origin to the "center" $(\bar{X}_1, \bar{X}_2)$ and then change the directions of coordinate axes to the directions of the eigenvectors e_1 and e_2 (see Figure 4.3).

Next, we drop perpendiculars from the given data points to the e_1-axis (see Figure 4.4). The first principal components are the e_1-coordinates of the feet of perpendiculars, that is, the projections on the e_1-axis. The projections of the data points on e_1-axis may be taken as approximations of the given data points hence we may replace the given data set with these points. Now, each of these

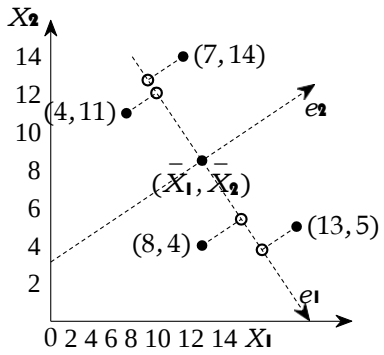


Figure 4.4: Projections of data points on the axis of the first principal component

PC1 components	-4.305187	3.736129	5.692828	-5.123769

Table 4.4: One-dimensional approximation to the data in Table 4.2

approximations can be unambiguously specified by a single number, namely, the e_1-coordinate of approximation. Thus the two-dimensional data set given in Table 4.2 can be represented approximately by the following one-dimensional data set (see Figure 4.5):

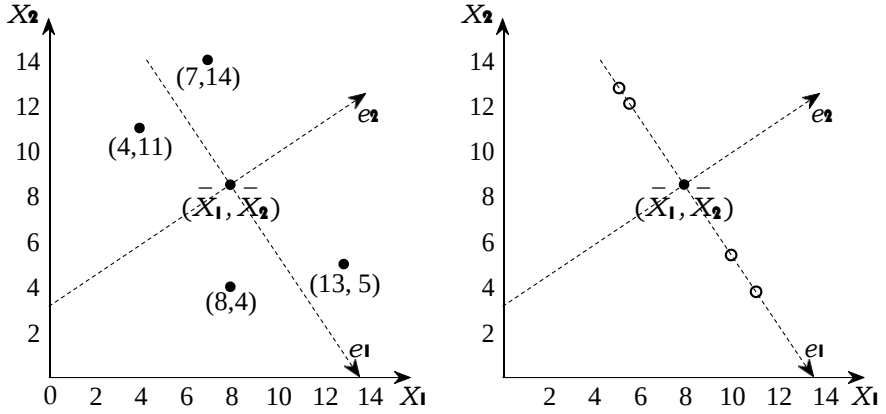


Figure 4.5: Geometrical representation of one-dimensional approximation to the data in Table 4.2

Sample questions

(a) Short answer questions

1. What is dimensionality reduction? How is it implemented?
2. Explain why dimensionality reduction is useful in machine learning.
3. What are the commonly used dimensionality reduction techniques in machine learning?
4. How is the subset selection method used for dimensionality reduction?
5. Explain the method of principal component analysis in machine learning.
6. What are the first principal components of a data?
7. Is subset selection problem an unsupervised learning problem? Why?

8. Is principal component analysis a supervised learning problem? Why?

(b) Long answer questions

1. Describe the forward selection algorithm for implementing the subset selection procedure for dimensionality reduction.
2. Describe the backward selection algorithm for implementing the subset selection procedure for dimensionality reduction.
3. What is the first principal component of a data? How one can compute it?
4. Describe with the use of diagrams the basic principle of PCA.
5. Explain the procedure for the computation of the principal components of a given data.
6. Describe how principal component analysis is carried out to reduce dimensionality of data sets.
7. Given the following data, compute the principal component vectors and the first principal components:

x	2	3	7
y	11	14	26

8. Given the following data, compute the principal component vectors and the first principal components:

x	-3	1	-2
y	2	-1	3

Module 3

In machine learning, there are several classification algorithms and, given a certain problem, more than one may be applicable. So there is a need to examine how we can assess how good a selected algorithm is. Also, we need a method to compare the performance of two or more different classification algorithms. These methods help us choose the right algorithm in a practical situation.

Methods of evaluation

Need for multiple validation sets

When we apply a classification algorithm in a practical situation, we always do a validation test. We keep a small sample of examples as validation set and the remaining set as the training set. The classifier developed using the training set is applied to the examples in the validation set. Based on the performance on the validation set, the accuracy of the classifier is assessed. But, the performance measure obtained by a single validation set alone does not give a true picture of the performance of a classifier. Also these measures alone cannot be meaningfully used to compare two algorithms. This requires us to have different validation sets.

Cross-validation in general, and k-fold cross-validation in particular, are two common method for generating multiple training-validation sets from a given dataset.

Statistical distribution of errors

We use a classification algorithm on a dataset and generate a classifier. If we do the training once, we have one classifier and one validation error. To average over randomness (in training data, initial weights, etc.), we use the same algorithm and generate multiple classifiers. We test these classifiers on multiple validation sets and record a sample of validation errors. We base our evaluation of the classification algorithm on the *statistical distribution of these validation errors*. We can use this distribution for assessing the *expected error* rate of the classification algorithm for that problem, or compare it with the error rate distribution of some other classification algorithm.

A detailed discussion of these ideas is beyond the scope of these notes.

No-free lunch theorem

Whatever conclusion we draw from our analysis is conditioned on the dataset we are given. We are not comparing classification algorithms in a domain-independent way but on some particular application. We are not saying anything about the expected error-rate of a learning algorithm, or comparing one learning algorithm with another algorithm, in general. Any result we have is only true for the particular application. There is no such thing as the “best” learning algorithm. For any

learning algorithm, there is a dataset where it is very accurate and another dataset where it is very poor. This is called the *No Free Lunch Theorem*.[1]

Other factors

Classification algorithms can be compared based not only on error rates but also on several other criteria like the following:

- risks when errors are generalized using loss functions
- training time and space complexity,
- testing time and space complexity,
- interpretability, namely, whether the method allows knowledge extraction which can be checked and validated by experts, and
- easy programmability.

Cross-validation

To test the performance of a classifier, we need to have a number of training/validation set pairs from a dataset X. To get them, if the sample X is large enough, we can randomly divide it then divide each part randomly into two and use one half for training and the other half for validation. Unfortunately, datasets are never large enough to do this. So, we use the same data split differently; this is called *cross-validation*.

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

The *holdout method* is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The algorithm fits a function using the training set only. Then the function is used to predict the output values for the data in the testing set (it has never seen these output values before). The errors it makes are used to evaluate the model.

K-fold cross-validation

In K-fold cross-validation, the dataset X is divided randomly into K equal-sized parts, X_i, $i = 1, \ldots, K$. To generate each pair, we keep one of the K parts out as the validation set V_i, and combine the remaining $K - 1$ parts to form the training set T_i. Doing this K times, each time leaving out another one of the K parts out, we get K pairs (V_i, T_i):

$$
\begin{aligned}
V_1 &= X_1, \quad T_1 = X_2 \cup X_3 \cup \ldots \cup X_K \\
V_2 &= X_2, \quad T_2 = X_1 \cup X_3 \cup \ldots \cup X_K \\
&\ldots \\
V_K &= X_K, \quad T_K = X_1 \cup X_2 \cup \ldots \cup X_{K-1}
\end{aligned}
$$

Remarks

1. There are two problems with this: First, to keep the training set large, we allow validation sets that are small. Second, the training sets overlap considerably, namely, any two training sets share $K-2$ parts.

[1] "We have dubbed the associated results NFL theorems because they demonstrate that if an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems."(David Wolpert and William Macready in [7])

2. K is typically 10 or 30. As K increases, the percentage of training instances increases and we get more robust estimators, but the validation set becomes smaller. Furthermore, there is the cost of training the classifier K times, which increases as K is increased.

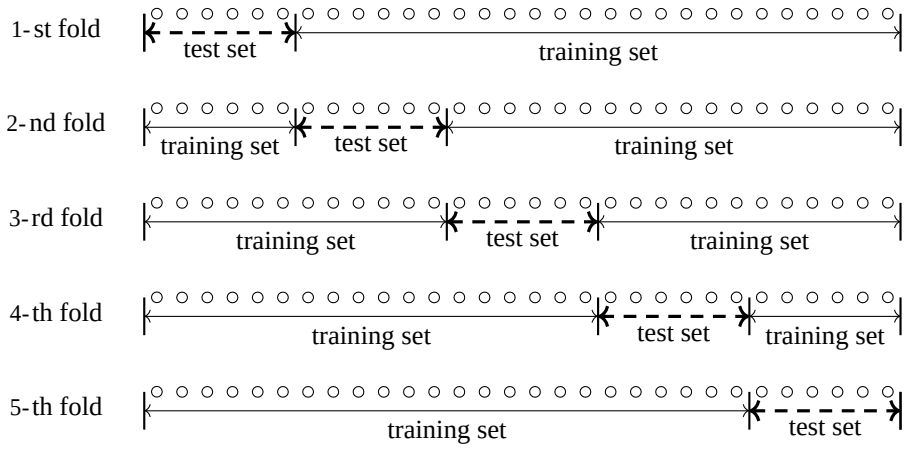


Figure 5.1: One iteration in a 5-fold cross-validation

Leave-one-out cross-validation

An extreme case of K-fold cross-validation is *leave-one-out* where given a dataset of N instances, only one instance is left out as the validation set and training uses the remaining $N-1$ instances. We then get N separate pairs by leaving out a different instance at each iteration. This is typically used in applications such as medical diagnosis, where labeled data is hard to find.

5×2 cross-validation

In this method, the dataset X is divided into two equal parts $X_1^{(1)}$ and $X_1^{(2)}$. We take as the training set and $X_1^{(2)}$ as the validation set. We then swap the two sets and take $X_1^{(2)}$ as the training set and $X_1^{(1)}$ as the validation set. This is the first fold. the process id repeated four more times to get ten pairs of training sets and validation sets.

$$
\begin{aligned}
T_1 &= X_1^{(1)}, \quad V_1 = X_1^{(2)} \\
T_2 &= X_1^{(2)}, \quad V_2 = X_1^{(1)} \\
T_3 &= X_2^{(1)}, \quad V_3 = X_2^{(2)} \\
T_4 &= X_2^{(2)}, \quad V_4 = X_2^{(1)} \\
&\vdots \\
T_9 &= X_5^{(1)}, \quad V_9 = X_5^{(2)} \\
T_{10} &= X_5^{(2)}, \quad V_{10} = X_5^{(1)}
\end{aligned}
$$

It has been shown that after five folds, the validation error rates become too dependent and do not add new information. On the other hand, if there are fewer than five folds, we get fewer data (fewer than ten) and will not have a large enough sample to fit a distribution and test our hypothesis.

Bootstrapping

Bootstrapping in statistics

In statistics, the term “bootstrap sampling”, the “bootstrap” or “bootstrapping” for short, refers to process of “random sampling with replacement”.

Example

For example, let there be five balls labeled A, B, C, D, E in an urn. We wish to select different samples of balls from the urn each sample containing two balls. The following procedure may be used to select the samples. This is an example for bootstrap sampling.

1. We select two balls from the basket. Let them be A and E. Record the labels.
2. Put the two balls back in the basket.
3. We select two balls from the basket. Let them be C and E. Record the labels.
4. Put the two balls back into the basket.

This is repeated as often as required. So we get different samples of size 2, say, A, E; B, E; etc. These samples are obtained by sampling with replacement, that is, by bootstrapping.

Bootstrapping in machine learning

In machine learning, bootstrapping is the process of computing performance measures using several randomly selected training and test datasets which are selected through a precess of sampling with replacement, that is, through bootstrapping. Sample datasets are selected multiple times.

The bootstrap procedure will create one or more new training datasets some of which are repeated. The corresponding test datasets are then constructed from the set of examples that were not selected for the respective training datasets.

Measuring error

True positive, false positive, etc.

Definitions

Consider a binary classification model derived from a two-class dataset. Let the class labels be c and $\neg c$. Let x be a test instance.

1. **True positive**

   Let the true class label of x be c. If the model predicts the class label of x as c, then we say that the classification of x is *true positive*.

2. **False negative**

   Let the true class label of x be c. If the model predicts the class label of x as e, then we say that the classification of x is *false negative*.

3. **True negative**

   Let the true class label of x be $\neg c$. If the model predicts the class label of x as e, then we say that the classification of x is *true negative*.

4. **False positive**

   Let the true class label of x be e. If the model predicts the class label of x as c, then we say that the classification of x is *false positive*.

	Actual label of x is c	Actual label of x is $\neg c$
Predicted label of x is c	True positive	False positive
Predicted label of x is c	False negative	True negative

Confusion matrix

A confusion matrix is used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. A *confusion matrix* is a table that categorizes predictions according to whether they match the actual value.

Two-class datasets

For a two-class dataset, a confusion matrix is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives.

Assume that a classifier is applied to a two-class test dataset for which the true values are known. Let TP denote the number of true positives in the predicted values, TN the number of true negatives, etc. Then the confusion matrix of the predicted values can be represented as follows:

	Actual condition is true	Actual condition is false
Predicted condition is true	TP	FP
Predicted condition is false	FN	FN

Table 5.1: Confusion matrix for two-class dataset

Multiclass datasets

Confusion matrices can be constructed for multiclass datasets also.

Example

If a classification system has been trained to distinguish between cats, dogs and rabbits, a confusion matrix will summarize the results of testing the algorithm for further inspection. Assuming a sample of 27 animals - 8 cats, 6 dogs, and 13 rabbits, the resulting confusion matrix could look like the table below: This confusion matrix shows that, for example, of the 8 actual cats, the system predicted that

	Actual "cat"	Actual "dog"	Actual "rabbit"
Predicted "cat"	5	2	0
Predicted "dog"	3	3	2
Predicted " rabbit"	0	1	11

three were dogs, and of the six dogs, it predicted that one was a rabbit and two were cats.

Precision and recall

In machine learning, precision and recall are two measures used to assess the quality of results produced by a binary classifier. They are formally defined as follows.

Definitions

Let a binary classifier classify a collection of test data. Let

$$\begin{aligned}\text{TP} &= \text{Number of true positives}\\ \text{TN} &= \text{Number of true negatives}\\ \text{FP} &= \text{Number of false positives}\\ \text{FN} &= \text{Number of false negatives}\end{aligned}$$

The *precision P* is defined as

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

The *recall R* is defined as

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

Problem 1

Suppose a computer program for recognizing dogs in photographs identifies eight dogs in a picture containing 12 dogs and some cats. Of the eight dogs identified, five actually are dogs while the rest are cats. Compute the precision and recall of the computer program.

Solution

We have:

$$\begin{aligned}\text{TP} &= 5\\ \text{FP} &= 3\\ \text{FN} &= 7\end{aligned}$$

The *precision P* is

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{5}{5+3} = \frac{5}{8}$$

The *recall R* is

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{5}{5+7} = \frac{5}{12}$$

Problem 2

Let there be 10 balls (6 white and 4 red balls) in a box and let it be required to pick up the red balls from them. Suppose we pick up 7 balls as the red balls of which only 2 are actually red balls. What are the values of precision and recall in picking red ball?

Solution

Obviously we have:

$$\begin{aligned}\text{TP} &= 2\\ \text{FP} &= 7-2=5\\ \text{FN} &= 4-2=2\end{aligned}$$

The *precision P* is

$$P = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{2}{2+5} = \frac{2}{7}$$

The *recall R* is

$$R = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{2}{2+2} = \frac{1}{2}$$

Problem 3

Assume the following: A database contains 80 records on a particular topic of which 55 are relevant to a certain investigation. A search was conducted on that topic and 50 records were retrieved. Of the 50 records retrieved, 40 were relevant. Construct the confusion matrix for the search and calculate the precision and recall scores for the search.

Solution

Each record may be assigned a class label “relevant" or “not relevant”. All the 80 records were tested for relevance. The test classified 50 records as “relevant”. But only 40 of them were actually relevant. Hence we have the following confusion matrix for the search:

	Actual ”relevant”	Actual “not relevant”
Predicted “relevant”	40	10
Predicted “not relevant”	15	25

Table 5.2: Example for confusion matrix

$$\text{TP} = 40$$
$$\text{FP} = 10$$
$$\text{FN} = 15$$

The *precision* P is

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{40}{40 + 10} = \frac{4}{5}$$

The *recall* R is

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{40}{40 + 15} = \frac{40}{55}$$

Other measures of performance

Using the data in the confusion matrix of a classifier of two-class dataset, several measures of performance have been defined. A few of them are listed below.

1. Accuracy $= \dfrac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$
2. Error rate $= 1 - \text{Accuracy}$
3. Sensitivity $= \dfrac{\text{TP}}{\text{TP} + \text{FN}}$
4. Specificity $= \dfrac{\text{TN}}{\text{TN} + \text{FP}}$
5. F-measure $= \dfrac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}$

Receiver Operating Characteristic (ROC)

The acronym ROC stands for Receiver Operating Characteristic, a terminology coming from signal detection theory. The ROC curve was first developed by electrical engineers and radar engineers during World War II for detecting enemy objects in battlefields. They are now increasingly used in machine learning and data mining research.

TPR and FPR

Let a binary classifier classify a collection of test data. Let, as before,

$$\begin{aligned} TP &= \text{Number of true positives} \\ TN &= \text{Number of true negatives} \\ FP &= \text{Number of false positives} \\ FN &= \text{Number of false negatives} \end{aligned}$$

Now we introduce the following terminology:

$$\begin{aligned} TPR &= \text{True Positive Rate} \\ &= \frac{TP}{TP + FN} \\ &= \text{Fraction of positive examples correctly classified} \\ &= \text{Sensitivity} \\ FPR &= \text{False Positive Rate} \\ &= \frac{FP}{FP + TN} \\ &= \text{Fraction of negative examples incorrectly classified} \\ &= 1 - \text{Specificity} \end{aligned}$$

ROC space

We plot the values of FPR along the horizontal axis (that is , x-axis) and the values of TPR along the vertical axis (that is, y-axis) in a plane. For each classifier, there is a unique point in this plane with coordinates (FPR, TPR). The ROC space is the part of the plane whose points correspond to (FPR, TPR). Each prediction result or instance of a confusion matrix represents one point in the ROC space.

The position of the point (FPR, TPR) in the ROC space gives an indication of the performance of the classifier. For example, let us consider some special points in the space.

Special points in ROC space

1. **The left bottom corner point $(0, 0)$: Always negative prediction**

   A classifier which produces this point in the ROC space never classifies an example as positive, neither rightly nor wrongly, because for this point $TP = 0$ and $FP = 0$. It always makes negative predictions. All positive instances are wrongly predicted and all negative instances are correctly predicted. It commits no false positive errors.

2. **The right top corner point $(1, 1)$: Always positive prediction**

   A classifier which produces this point in the ROC space always classifies an example as positive because for this point $FN = 0$ and $TN = 0$. All positive instances are correctly predicted and all negative instances are wrongly predicted. It commits no false negative errors.

3. **The left top corner point $(0, 1)$: Perfect prediction**

   A classifier which produces this point in the ROC space may be thought as a perfect classifier. It produces no false positives and no false negatives.

4. **Points along the diagonal: Random performance**

   Consider a classifier where the class labels are randomly guessed, say by flipping a coin. Then, the corresponding points in the ROC space will be lying very near the diagonal line joining the points $(0, 0)$ and $(1, 1)$.

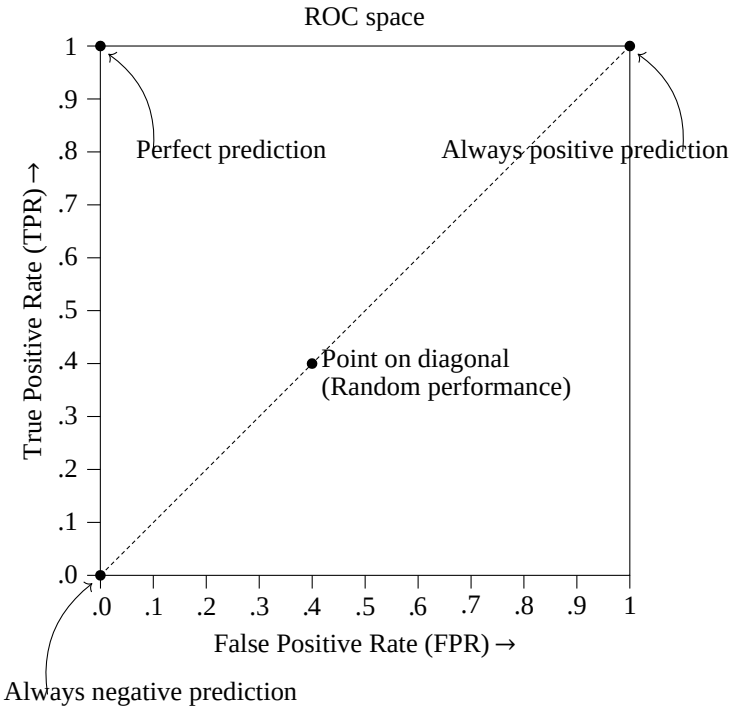


Figure 5.2: The ROC space and some special points in the space

ROC curve

In the case of certain classification algorithms, the classifier may depend on a parameter. Different values of the parameter will give different classifiers and these in turn give different values to TPR and FPR. The ROC curve is the curve obtained by plotting in the ROC space the points (TPR, FPR) obtained by assigning all possible values to the parameter in the classifier.

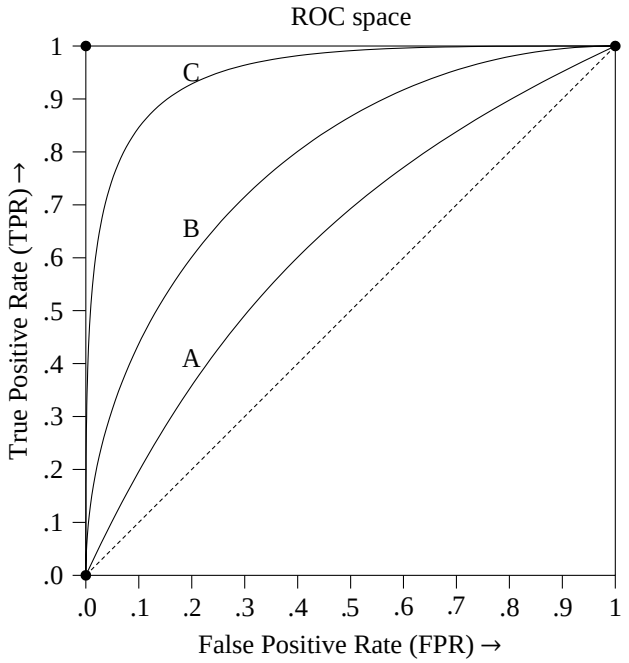


Figure 5.3: ROC curves of three different classifiers A, B, C

The closer the ROC curve is to the top left corner$(0, 1)$of the ROC space, the better the accuracy of the classifier. Among the three classifiers A, B, C with ROC curves as shown in Figure 5.3, the classifier C is closest to the top left corner of the ROC space. Hence, among the three, it gives the best accuracy in predictions.

Example

Cut-off value of BMI	Breast cancer		Normal persons		TPR	FPR
	TP	FN	FP	TN		
18	100	0	200	0	1.00	1.000
20	100	0	198	2	1.00	0.990
22	99	1	177	23	0.99	0.885
24	95	5	117	83	0.95	0.585
26	85	15	80	120	0.85	0.400
28	66	34	53	147	0.66	0.265
30	47	53	27	173	0.47	0.135
32	34	66	17	183	0.34	0.085
34	21	79	14	186	0.21	0.070
36	17	83	6	194	0.17	0.030
38	7	93	4	196	0.07	0.020
40	1	99	1	199	0.01	0.005

Table 5.3: Data on breast cancer for various values of BMI

The body mass index (BMI) of a person is defined as (weight(kg)/height(m)^2). Researchers have established a link between BMI and the risk of breast cancer among women. The higher the BMI the higher the risk of developing breast cancer. The critical threshold value of BMI may depend on several parameters like food habits, socio-cultural-economic background, life-style, etc. Table 5.3

gives real data of a breast cancer study with a sample having 100 patients and 200 normal persons.[2] The table also shows the values of TPR and FPR for various cut-off values of BMI. The ROC curve of the data in Table 5.3 is shown in Figure 5.4.

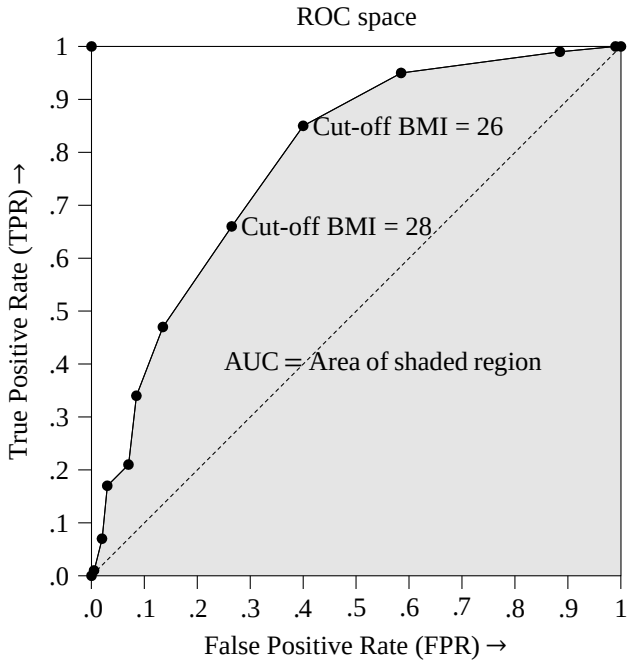


Figure 5.4: ROC curve of data in Table 5.3 showing the points closest to the perfect prediction point $(0, 1)$

Area under the ROC curve

The measure of the area under the ROC curve is denoted by the acronym AUC (see Figure 5.4). The value of AUC is a measure of the performance of a classifier. For the perfect classifier, AUC = 1.0.

Sample questions

(a) Short answer questions

1. What is cross-validation in machine learning?
2. What is meant by 5 2x cross-validation?
3. What is meant by leave-one-out cross validation?
4. What is meant by the confusion matrix of a binary classification problem.
5. Define the following terms: precision, recall, sensitivity, specificity.
6. What is ROC curve in machine learning?
7. What are true positive rates and false positive rates in machine learning?
8. What is AUC in relation to ROC curves?

[2] https://www.ncbi.nlm.nih.gov/ pmc/articles/PMC3755824/

(b) Long answer questions

1. Explain cross-validation in machine learning. Explain the different types of cross-validations.

2. What is meant by true positives etc.? What is meant by confusion matrix of a binary classification problem? Explain how this can be extended to multi-class problems.

3. What are ROC space and ROC curve in machine learning? In ROC space, which points correspond to perfect prediction, always positive prediction and always negative prediction? Why?

4. Consider a two-class classification problem of predicting whether a photograph contains a man or a woman. Suppose we have a test dataset of 10 records with expected outcomes and a set of predictions from our classification algorithm.

	Expected	Predicted
1	man	woman
2	man	man
3	woman	woman
4	man	man
5	woman	man
6	woman	woman
7	woman	woman
8	man	man
9	man	woman
10	woman	woman

   (a) Compute the confusion matrix for the data.

   (b) Compute the accuracy, precision, recall, sensitivity and specificity of the data.

5. Suppose 10000 patients get tested for flu; out of them, 9000 are actually healthy and 1000 are actually sick. For the sick people, a test was positive for 620 and negative for 380. For the healthy people, the same test was positive for 180 and negative for 8820. Construct a confusion matrix for the data and compute the accuracy, precision and recall for the data.

6. Given the following data, construct the ROC curve of the data. Compute the AUC.

Threshold	TP	TN	FP	FN
1	0	25	0	29
2	7	25	0	22
3	18	24	1	11
4	26	20	5	3
5	29	11	14	0
6	29	0	25	0
7	29	0	25	0

7. Given the following hypothetical data at various cut-off points of mid-arm circumference of mid-arm circumference to detect low birth-weight construct the ROC curve for the data.

Mid-arm circumference (cm)	Normal birth-weight TP	Low birth-weight TN
8.3	13	867
≤8.4	24	844
8.5	73	826
≤8.6	90	800
8.7	113	783
≤8.8	119	735
8.9	121	626
9.0	125	505
≤9.1	127	435
≤ 9.2 and above	130	0

Bayesian classifier and ML estimation

The Bayesian classifier is an algorithm for classifying multiclass datasets. This is based on the Bayes' theorem in probability theory. Bayes in whose name the theorem is known was an English statistician who was known for having formulated a specific case of a theorem that bears his name. The classifier is also known as "naive Bayes Algorithm" where the word "naive" is an English word with the following meanings: simple, unsophisticated, or primitive. We first explain Bayes' theorem and then describe the algorithm. Of course, we require the notion of conditional probability.

Conditional probability

The probability of the occurrence of an event A given that an event B has already occurred is called the *conditional probability of A given B* and is denoted by $P(A|B)$. We have

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \text{if} \quad P(B) \neq 0.$$

Independent events

1. Two events A and B are said to be *independent* if

$$P(A \cap B) = P(A)P(B).$$

2. Three events A, B, C are said to be *pairwise independent* if

$$P(B \cap C) = P(B)P(C)$$
$$P(C \cap A) = P(C)P(A)$$
$$P(A \cap B) = P(A)P(B)$$

3. Three events A, B, C are said to be *mutually independent* if

$$P(B \cap C) = P(B)P(C) \tag{6.1}$$
$$P(C \cap A) = P(C)P(A) \tag{6.2}$$
$$P(A \cap B) = P(A)P(B) \tag{6.3}$$
$$P(A \cap B \cap C) = P(A)P(B)P(C) \tag{6.4}$$

4. In general, a family of k events $A_1, A_2, \ldots, A_k$ is said to be mutually independent if for any subfamily consisting of $A_{i_1}, \ldots A_{i_m}$ we have

$$P(A_{i_1} \cap \ldots \cap A_{i_m}) = P(A_{i_1}) \ldots P(A_{i_m}).$$

Remarks

Consider events and respective probabilities as shown in Figure 6.1. It can be seen that, in this case, the conditions Eqs.(6.1)–(6.3) are satisfied, but Eq.(6.4) is not satisfied. But if the probabilities are as in Figure 6.2, then Eq.(6.4) is satisfied but all the conditions in Eqs.(6.1)–(6.2) are not satisfied.

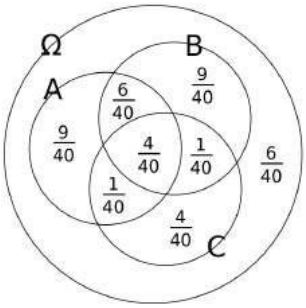


Figure 6.1: Events A, B, C which are not mutually independent: Eqs.(6.1)–(6.3) are satisfied, but Eq.(6.4) is not satisfied.

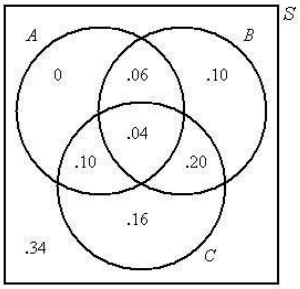


Figure 6.2: Events A, B, C which are not mutually independent: Eq.(6.4) is satisfied but Eqs.(6.1)–(6.2) are not satisfied.

Bayes' theorem

Theorem

Let A and B any two events in a random experiment. If $P(A) \neq 0$, then

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

Remarks

1. The importance of the result is that it helps us to "invert" conditional probabilities, that is, to express the conditional probability $P(A|B)$ in terms of the conditional probability $P(B|A)$.
2. The following terminology is used in this context:
   - A is called the *proposition* and B is called the *evidence*.
   - $P(A)$ is called the *prior probability* of proposition and $P(B)$ is called the prior probability of evidence.

- $P(A|B)$ is called the *posterior probability* of A given B.
- $P(B|A)$ is called the *likelihood* of B given A.

Generalisation

Let the sample space be divided into disjoint events $B_1, B_2, \ldots, B_n$ and A be any event. Then we have

$$P(B_k|A) = \frac{P(A|B_k)P(B_k)}{\sum_{i=1}^{n} P(A|B_i)P(B_i)}$$

Examples

Problem 1

Consider a set of patients coming for treatment in a certain clinic. Let A denote the event that a "Patient has liver disease" and B the event that a "Patient is an alcoholic." It is known from experience that 10% of the patients entering the clinic have liver disease and 5% of the patients are alcoholics. Also, among those patients diagnosed with liver disease, 7% are alcoholics. Given that a patient is alcoholic, what is the probability that he will have liver disease?

Solution

Using the notations of probability, we have

$$\begin{aligned} P(A) &= 10\% = 0.10 \\ P(B) &= 5\% = 0.05 \\ P(B|A) &= 7\% = 0.07 \\ P(A|B) &= \frac{P(B|A)P(A)}{P(B)} \\ &= \frac{0.07 \times 0.10}{0.05} \\ &= 0.14 \end{aligned}$$

Problem 2

Three factories A, B, C of an electric bulb manufacturing company produce respectively 35%. 35% and 30% of the total output. Approximately 1.5%, 1% and 2% of the bulbs produced by these factories are known to be defective. If a randomly selected bulb manufactured by the company was found to be defective, what is the probability that the bulb was manufactures in factory A?

Solution

Let A, B, C denote the events that a randomly selected bulb was manufactured in factory A, B, C respectively. Let D denote the event that a bulb is defective. We have the following data:

$$P(A) = 0.35, \quad P(B) = 0.35, \quad P(C) = 0.30$$
$$P(D|A) = 0.015, \quad P(D|B) = 0.010, \quad P(D|C) = 0.020$$

We are required to find $P(A|D)$. By the generalisation of the Bayes' theorem we have:

$$P(A|D) = \frac{P(D|A)P(A)}{P(D|A)P(A) + P(D|B)P(B) + P(D|C)P(C)}$$
$$= \frac{0.015 \times 0.35}{0.015 \times 0.35 + 0.010 \times 0.35 + 0.020 \times 0.30}$$
$$= 0.356.$$

Naive Bayes algorithm

Assumption

The naive Bayes algorithm is based on the following assumptions:

- All the *features are independent* and are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature.
- The data has *class-conditional independence*, which means that events are independent so long as they are conditioned on the same class value.

These assumptions are, in general, true in many real world problems. It is because of these assumptions, the algorithm is called a *naive* algorithm.

Basic idea

Suppose we have a training data set consisting of N examples having n features. Let the features be named as $(F_1, \ldots, F_n)$. A feature vector is of the form $(f_1, f_2, \ldots, f_n)$. Associated with each example, there is a certain class label. Let the set of class labels be $\{c_1, c_2, \ldots, c_p\}$.

Suppose we are given a test instance having the feature vector

$$X = (x_1, x_2, \ldots, x_n).$$

We are required to determine the most appropriate class label that should be assigned to the test instance. For this purpose we compute the following conditional probabilities

$$P(c_1|X), P(c_2|X), \ldots, P(c_p|X). \tag{6.5}$$

and choose the maximum among them. Let the maximum probability be $P(c_i|X)$. Then, we choose c_i as the most appropriate class label for the training instance having X as the feature vector.

The direct computation of the probabilities given in Eq.(6.5) are difficult for a number of reasons. The Bayes' theorem can b applied to obtain a simpler method. This is explained below.

Computation of probabilities

Using Bayes' theorem, we have:

$$P(c_k|X) = \frac{P(X|c_k)P(c_k)}{P(X)} \tag{6.6}$$

Since, by assumption, the data has class-conditional independence, we note that the events "$x_1|c_k$", "$x_2|c_k$", ..., $x_n|c_k$ are independent (because they are all conditioned on the same class label c_k). Hence we have

$$\begin{aligned} P(X|c_k) &= P((x_1, x_2, \ldots, x_n)|c_k) \\ &= P(x_1|c_k)P(x_2|c_k)\ldots P(x_n|c_k) \end{aligned}$$

Using this in Eq,(6.6) we get

$$P(c_k|X) = \frac{P(x_1|c_k)P(x_2|c_k)\ldots P(x_n|c_k)P(c_k)}{P(X)}.$$

Since the denominator $P(X)$ is independent of the class labels, we have

$$P(c_k|X) \propto P(x_1|c_k)P(x_2|c_k)\ldots P(x_n|c_k)P(c_k).$$

So it is enough to find the maximum among the following values:

$$P(x_1|c_k)P(x_2|c_k)\ldots P(x_n|c_k)P(c_k), \quad k = 1, \ldots, p.$$

Remarks

The various probabilities in the above expression are computed as follows:

$$P(c_k) = \frac{\text{No. of examples with class label } c_k}{\text{Total number of examples}}$$

$$P(x_j|c_k) = \frac{\text{No. of examples with } j\text{th feature equal to } x_j \text{ and class label } c_k}{\text{No. of examples with class label } c_k}$$

The algorithm

Algorithm: Naive Bayes

Let there be a training data set having n features $F_1, \ldots, F_n$. Let f_1 denote an arbitrary value of F_1, f_2 of F_2, and so on. Let the set of class labels be $\{c_1, c_2, \ldots, c_p\}$. Let there be given a test instance having the feature vector

$$X = (x_1, x_2, \ldots, x_n).$$

We are required to determine the most appropriate class label that should be assigned to the test instance.

Step 1.Compute the probabilities $P(c_k)$ for $k = 1, \ldots, p$.

Step 2.Form a table showing the conditional probabilities

$$P(f_1|c_k), \quad P(f_2|c_k), \quad \ldots \quad, P(f_n|c_k)$$

for all values of $f_1, f_2, \ldots, f_n$ and for $k = 1, \ldots, p$.

Step 3.Compute the products

$$q_k = P(x_1|c_k)P(x_2|c_k)\ldots P(x_n|c_k)P(c_k)$$

for $k = 1, \ldots, p$.

Step 4.Find j such $q_j = \max\{q_1, q_2, \ldots, q_p\}$.

Step 5.Assign the class label c_j to the test instance X.

Remarks

In the above algorithm, Steps 1 and 2 constitute the learning phase of the algorithm. The remaining steps constitute the testing phase. For testing purposes, only the table of probabilities is required; the original data set is not required.

Example

Problem

Consider a training data set consisting of the fauna of the world. Each unit has three features named "Swim", "Fly" and "Crawl". Let the possible values of these features be as follows:

Swim	Fast, Slow, No
Fly	Long, Short, Rarely, No
Crawl	Yes, No

For simplicity, each unit is classified as "Animal", "Bird" or "Fish". Let the training data set be as in Table 6.1. Use naive Bayes algorithm to classify a particular species if its features are (Slow, Rarely, No)?

Sl. No.	Swim	Fly	Crawl	Class
1	Fast	No	No	Fish
2	Fast	No	Yes	Animal
3	Slow	No	No	Animal
4	Fast	No	No	Animal
5	No	Short	No	Bird
6	No	Short	No	Bird
7	No	Rarely	No	Animal
8	Slow	No	Yes	Animal
9	Slow	No	No	Fish
10	Slow	No	Yes	Fish
11	No	Long	No	Bird
12	Fast	No	No	Bird

Table 6.1: Sample data set for naive Bayes algorithm

Solution

In this example, the features are

$$F_1 = \text{“Swim”}, \qquad F_2 = \text{“Fly”}, \qquad F_3 = \text{“Crawl”}.$$

The class labels are

$$c_1 = \text{“Animal”}, \qquad c_2 = \text{“ Bird”}, \qquad c_3 = \text{“Fish”}.$$

The test instance is (Slow, Rarely, No) and so we have:

$$x_1 = \text{“Slow”}, \qquad x_2 = \text{“Rarely”}, \qquad x_3 = \text{“No”}.$$

We construct the frequency table shown in Table 6.2 which summarises the data. (It may be noted that the construction of the frequency table is not part of the algorithm.)

Class	Features									Total
	Swim (F_1)			Fly (F_2)				Crawl (F_3)		
	Fast	Slow	No	Long	Short	Rarely	No	Yes	No	
Animal (c_1)	2	2	1	0	0	1	4	2	3	5
Bird (c_2)	1	0	3	1	2	0	1	1	3	4
Fish (c_3)	1	2	0	0	0	0	3	0	3	3
Total	4	4	4	1	2	1	8	4	8	12

Table 6.2: Frequency table for the data in Table 6.1

Step 1.We compute following probabilities.

$$P(c_1) = \frac{\text{No. of records with class label “Animal”}}{\text{Total number of examples}}$$

$$= 5/12$$

$$P(c_2) = \frac{\text{No. of records with class label “Bird”}}{\text{Total number of examples}}$$

$$= 4/12$$

$$P(c_3) = \frac{\text{No of records with class label “Fish”}}{\text{Total number of examples}}$$

$$= 3/12$$

Step 2.We construct the following table of conditional probabilities:

Class	Features								
	Swim (F_1) f_1			Fly (F_2) f_2				Crav (F_3) f_3	
	Fast	Slow	No	Long	Short	Rarely	No	Yes	No
Animal (c_1)	2/5	2/5	1/5	0/5	0/5	1/5	4/5	2/5	3/5
Bird (c_2)	1/4	0/4	3/4	1/4	2/4	0/4	1/4	0/4	4/4
Fish (c_3)	1 3	2/3	0/3	0/3	0/3	0/3	3/3	0/3	3/3

Table 6.3: Table of the conditional probabilities $P(f_i|c_k)$

Note: The conditional probabilities are calculated as follows:

$$P((F_1 = \text{Slow})|c_1) = \frac{\text{No. of records with } F_1 = \text{Slow and class label } c_1}{\text{No. of records with class label } c_1} = 2/5.$$

Step 3.We now calculate the following numbers:

$$\begin{aligned} q_1 &= P(x_1|c_1)P(x_2|c_1)P(x_3|c_1)P(c_1) \\ &(2/5)\times(1/5)\times(3/5)\times(5/12) \\ &=0.02 \\ q_2 &= P(x_1|c_2)P(x_2|c_2)P(x_3|c_2)P(c_2) \\ &(0/4)\times(0/4)\times(3/4)\times(4/12) \\ &=0 \\ q_3 &= P(x_1|c_3)P(x_2|c_3)P(x_3|c_3)P(c_3) \\ &(2/3)\times(0/3)\times(3/3)\times(3/12) \\ &=0 \end{aligned}$$

Step 4.Now

$$\max\{q_1, q_2, q_3\} = 0.05.$$

Step 5.The maximum is q_1 an it corresponds to the class label

c_1 = " Animal".

So we assign the class label "Animal" to the test instance "(Slow, Rarely, No)".

Using numeric features with naive Bayes algorithm

The naive Bayes algorithm can be applied to a data set only if the features are categorical. This is so because, the various probabilities are computed using the various frequencies and the frequencies can be counted only if each feature has a limited set of values.

If a feature is numeric, it has to be *discretized* before applying the algorithm. The discretization is effected by putting the numeric values into categories known as *bins*. Because of this *discretization* is also known as *binning*. This is ideal when there are large amounts of data.

There are several different ways to discretize a numeric feature.

1. If there are natural categories or *cut points* in the distribution of values, use these cut points to create the bins. For example, let the data consists of records of times when certain activities were carried out. The the categories, or bins, may be created as in Figure 6.3.

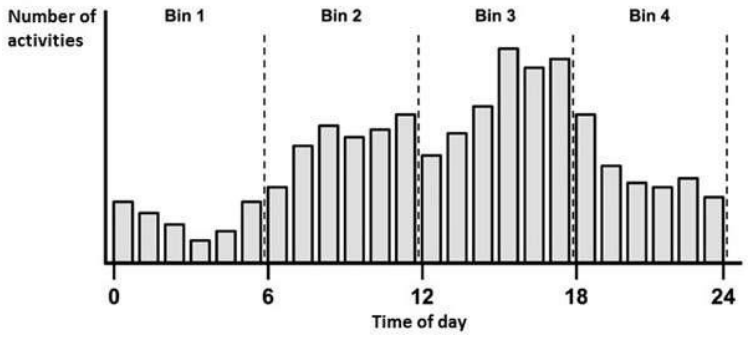


Figure 6.3: Discretization of numeric data: Example

2. If there are no obvious cut points, we may discretize the feature using quantiles. We may divide the data into three bins with tertiles, four bins with quartiles, or five bins with quintiles, etc.

Maximum likelihood estimation (ML estimation)

To develop a Bayesian classifier, we need the probabilities $P(x|c_k)$ for the class labels $c_1, \ldots, c_k$. These probabilities are estimated from the given data. There is need to know whether the sample is truly random so that the computed probabilities are good approximations to true probabilities. If they are good approximations of true probabilities, then there would be an underlying probability distribution. Suppose we have reasons to believe that the underlying distribution has a particular form, say binomial, Poisson or normal. These forms are defined by probability functions or probability density functions. There are parameters which define these functions, and these parameters are to be estimated to test whether a given data follow some particular distribution. Maximum likelihood estimation is particular method to estimate the parameters of a probability distribution.

Definition

Maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model, given observations. MLE attempts to find the parameter values that *maximize the likelihood function*, given the observations. The resulting estimate is called a *maximum likelihood estimate*, which is also abbreviated as MLE.

The general MLE method

Suppose we have a random sample $X = \{x_1, \ldots, x_n\}$ taken from a probability distribution having the probability mass function or probability density function $p(x|\theta)$ where x denotes a value of the random variable and θ denotes the set of parameters that appear in the function.

The *likelihood* of sample X is a function of the parameter θ and is defined as

$$l(\theta) = p(x_1|\theta)\, p(x_2|\theta) \ldots p(x_n|\theta) .$$

In maximum likelihood estimation, we find the value of θ that makes the value of the likelihood function maximum. For computation convenience, we define the *log likelihood function* as the logarithm of the likelihood function:

$$L(\theta) \quad \log l(\theta)$$
$$= \log p(x_1|\theta) + \log p(x_2|\theta) + \ldots + \log p(x_n|\theta).$$

A value of θ that maximizes $L(\theta)$ will also maximise $l(\theta)$ and vice-versa. Hence, in maximum likelihood estimation, we find θ that maximizes the log likelihood function. Sometimes the maximum likelihood estimate of θ is denoted by $\hat{\theta}$.

Special cases

1. Bernoulli density

In a Bernoulli distribution there are two outcomes: An event occurs or it does not, for example, an instance is a positive example of the class, or it is not. The event occurs and the Bernoulli random variable X takes the value 1 with probability p, and the nonoccurrence of the event has probability $1 - p$ and this is denoted by X taking the value 0.

The probability function of X is given by

$$f(x|p) = p^x(1-p)^{1-x}, \quad x = 0, 1.$$

In this function, the probability p is the only parameter.

Estimation of p

Consider a random sample $X = \{x_1, \ldots, x_n\}$ taken from a Bernoulli distribution with the probability function $f(x|p)$. The log likelihood function is

$$\begin{aligned} L(p) &= \log f(x_1|p) + \ldots + \log f(x_n|p) \\ &= \log p^{x_1}(1-p)^{1-x_1} + \ldots + \log p^{x_n}(1-p)^{1-x_n} \\ &= [x_1 \log p + (1-x_1)\log(1-p)] + \ldots + [x_n \log p + (1-x_n)\log(1-p)] \end{aligned}$$

To find the value of p that maximizes $L(p)$ we set up the equation

$$\frac{dL}{dp} = 0,$$

that is,

$$\left[\frac{x_1}{p} - \frac{1-x_1}{1-p}\right] + \ldots + \left[\frac{x_n}{p} - \frac{1-x_n}{1-p}\right] = 0.$$

Solving this equation, we have the maximum likelihood estimate of p as

$$\hat{p} = \frac{1}{n}(x_1 + \ldots + x_n).$$

2. Multinomial density

Suppose that the outcome of a random event is one of K classes, each of which has a probability of occurring p_i with

$$p_1 + \ldots + p_K = 1.$$

We represent each outcome by an ordered K-tuple $\mathbf{x} = (x_1, \ldots, x_K)$ where exactly one of $x_1, \ldots, x_K$ is 1 and all others are 0. $x_i = 1$ if the outcome in the i-th class occurs. The probability function can be expressed as

$$f(\mathbf{x}|p_1, \ldots, p_K) = p_1^{x_1} \ldots p_K^{x_K}.$$

Here, $p_1, \ldots, p_K$ are the parameters.

We choose n random samples. The i-the sample may be represented by

$$\mathbf{x}_i = (x_{1i}, \ldots, x_{Ki}).$$

The values of the parameters that maximizes the likelihood function can be shown to be

$$\hat{p}_k = \frac{1}{n}(x_{k1} + x_{k2} + \ldots + x_{kn}).$$

(We leave the details of the derivation as an exercise.)

3. Gaussian (normal) density

A continuous random variable X has the Gaussian or normal distribution if its density function is

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad -\infty < x < \infty.$$

Here μ and σ are the parameters.

Given a sample $x_1, x_2, \ldots, x_n$ from the distribution. the log likelihood function is

$$L(\mu,\sigma) = -\frac{n}{2}\log(2\pi) - n\log\sigma - \frac{1}{2\sigma^2}\left[(x_1-\mu)^2 + \ldots + (x_n-\mu)^2\right].$$

Setting up the equations

$$\frac{dL}{d\mu} = 0, \quad \frac{dL}{d\sigma} = 0$$

and solving for μ and σ we get the maximum likelihood estimates of μ and σ as

$$\hat{\mu} = \frac{1}{n}(x_1 + \ldots + x_n)$$
$$\hat{\sigma}^2 = \frac{1}{n}((x_1-\hat{\mu})^2 + \ldots + (x_n-\hat{\mu})^2)$$

(We leave the details of the derivation as an exercise.)

Sample questions

(a) Short answer questions

1. What are the assumptions under the naive Bayes algorithm?
2. Why is naive Bayes algorithm "naive"?
3. Given an instance X of a feature vector and a class label c_k, explain how Bayes theorem is used to compute the probability $P(c_k|X)$.
4. What does a naive Bayes classifier do?
5. What is naive Bayes used for?
6. Is naive Bayes supervised or unsupervised? Why?
7. What is meant by the likelihood of a random sample taken from population?
8. How do we use numeric features in naive Bayes algorithm?

(b) Long answer questions

1. State Bayes theorem and illustrate it with an example.
2. Explain naive Bayes algorithm.
3. Use naive Bayes algorithm to determine whether a red domestic SUV car is a stolen car or not using the following data:

Example no.	Colour	Type	Origin	Whether stolen
1	red	sports	domestic	yes
2	red	sports	domestic	no
3	red	sports	domestic	yes
4	yellow	sports	domestic	no
5	yellow	sports	imported	yes
6	yellow	SUV	imported	no
7	yellow	SUV	imported	yes
8	yellow	SUV	domestic	no
9	red	SUV	imported	no
10	red	sports	imported	yes

4. Based on the following data determine the gender of a person having height 6 ft., weight 130 lbs. and foot size 8 in. (use naive Bayes algorithm).

person	height (feet)	weight (lbs)	foot size (inches)
male	6.00	180	10
male	6.00	180	10
male	5.50	170	8
male	6.00	170	10
female	5.00	130	8
female	5.50	150	6
female	5.00	130	6
female	6.00	150	8

5. Given the following data on a certain set of patients seen by a doctor, can the doctor conclude that a person having chills, fever, mild headache and without running nose has the flu?

chills	running nose	headache	fever	has flu
Y	N	mild	Y	N
Y	Y	no	N	Y
Y	N	strong	Y	Y
N	Y	mild	Y	Y
N	N	no	N	N
N	Y	strong	Y	Y
N	Y	strong	N	N
Y	Y	mild	Y	Y

6. Explain the general MLE method for estimating the parameters of a probability distribution.

7. Find the ML estimate for the parameter p in the binomial distribution whose probability function is

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, 2, \ldots, n$$

8. Compute the ML estimate for the parameter λ in the Poisson distribution whose probability function is

$$f(x) = e^{-\lambda} \frac{\lambda^x}{x!}, \qquad x = 0, 1, 2, \ldots$$

Find the ML estimate of the parameter p in the geometric distribution defined by the probability mass function

$$f(x) = (1-p)p^x, \qquad x = 1, 2, 3, \ldots$$

Regression

We have seen in Section 1.5.3 that regression is a supervised learning problem where there is an input x an output y and the task is to learn the mapping from the input to the output. We have also seen that the approach in machine learning is that we assume a model, that is, a relation between x and y containing a set of parameters, say, θ in the following form:

$$y = g(x, \theta).$$

$g(x, \theta)$ is the regression function. The machine learning program optimizes the parameters θ such that the approximation error is minimized, that is, our estimates are as close as possible to the correct values given in the training set. In this chapter we discuss a method, known as ordinary least squares method, to estimate the parameters. In fact this method can be derived from the maximum likelihood estimation method discussed in Section 6.5.

Definition

A *regression problem* is the problem of determining a relation between one or more independent variables and an output variable which is a real continuous variable, given a set of observed values of the set of independent variables and the corresponding values of the output variable.

Examples

1. Let us say we want to have a system that can predict the price of a used car. Inputs are the car attributes âĂŤ brand, year, engine capacity, mileage, and other information âĂŤ that we believe affect a car's worth. The output is the price of the car.
2. Consider the navigation of a mobile robot, say an autonomous car. The output is the angle by which the steering wheel should be turned at each time, to advance without hitting obstacles and deviating from the route. Inputs are provided by sensors on the car like a video camera, GPS, and so forth.
3. In finance, the capital asset pricing model uses regression for analyzing and quantifying the systematic risk of an investment.
4. In economics, regression is the predominant empirical tool. For example, it is used to predict consumption spending, inventory investment, purchases of a country's exports, spending on imports, labor demand, and labor supply.

Different regression models

The different regression models are defined based on type of functions used to represent the relation between the dependent variable y and the independent variables.

1. **Simple linear regression**

   Assume that there is only one independent variable x. If the relation between x and y is modeled by the relation

   $$y = a + bx$$

   then we have a simple linear regression.

2. **Multiple regression**

   Let there be more than one independent variable, say $x_1, x_2, \ldots, x_n$, and let the relation between y and the independent variables be modeled as

   $$y = a_0 + a_1 x_1 + \ldots + a_n x_n$$

   then it is case of multiple linear regression or multiple regression.

3. **Polynomial regression**

   Let there be only one variable x and let the relation between x y be modeled as

   $$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

   for some positive integer $n > 1$, then we have a polynomial regression.

4. **Logistic regression**

   Logistic regression is used when the dependent variable is binary (0/1, True/False, Yes/No) in nature. Even though the output is a binary variable, what is being sought is a probability function which may take any value from 0 to 1.

Criterion for minimisation of error

In regression, we would like to write the numeric output y, called the dependent variable, as a function of the input x, called the independent variable. We assume that the output is the sum of a function $f(x)$ of the input and some random error denoted by s:

$$y = f(x) + s.$$

Here the function $f(x)$ is unknown and we would like to approximate it by some estimator $g(x, \theta)$ containing a set of parameters θ. We assume that the random error s follows normal distribution with mean 0.

Let $x_1, \ldots, x_n$ be a random sample of observations of the input variable x and $y_1, \ldots, y_n$ the corresponding observed values of the output variable y.

Using the assumption that the error s follows normal distribution, we can apply the method of maximum likelihood estimation to estimate the values of the parameter θ. It can be shown that the values of θ which maximizes the likelihood function are the values of θ that minimizes the following sum of squares:

$$E(\theta) = (y_1 - g(x_1, \theta))^2 + \ldots + (y_n - g(x_n, \theta))^2.$$

The method of finding the value of θ as that value of θ that minimizes $E(\theta)$ is known as the *ordinary least squares method*.

The full details of the derivation of the above result are beyond the scope of these notes.

x	x_1	x_2	x_n
y	y_1	y_2	y_n

Table 7.1: Data set for simple linear regression

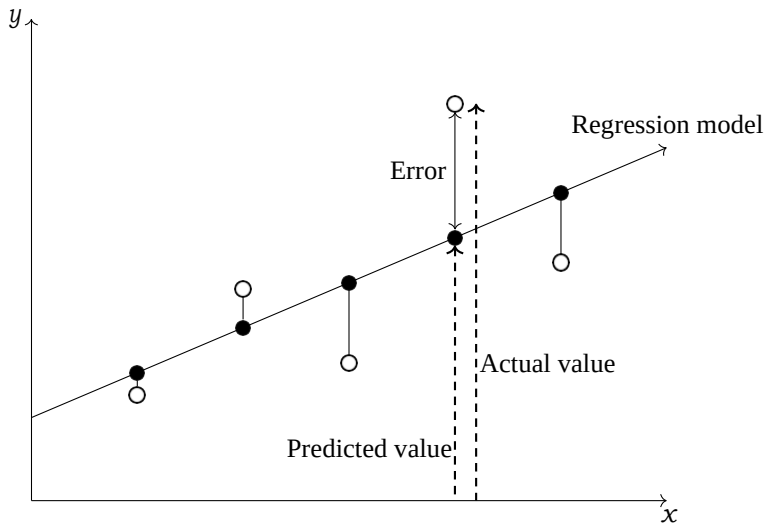


Figure 7.1: Errors in observed values

Simple linear regression

Let x be the independent predictor variable and y the dependent variable. Assume that we have a set of observed values of x and y:

A simple linear regression model defines the relationship between x and y using a line defined by an equation in the following form:

$$y = \alpha + \beta x$$

In order to determine the optimal estimates of α and β, an estimation method known as *Ordinary Least Squares* (OLS) is used.

The OLS method

In the OLS method, the values of y-intercept and slope are chosen such that they minimize the sum of the squared errors; that is, the sum of the squares of the vertical distance between the predicted y-value and the actual y-value (see Figure 7.1). Let $\hat{y}_i$ be the predicted value of y_i. Then the sum of squares of errors is given by

$$E = \sum_{i=1}^{n} (y - \hat{y})_i^2$$
$$= \sum_{i=1}^{n} [y_i - (\alpha + \beta x_i)]^2$$

So we are required to find the values of α and β such that E is minimum. Using methods of calculus, we can show that the values of α and b, which are respectively the values of α and β for which E is minimum, can be obtained by solving the following equations.

$$\sum_{i=1}^{n} y_i = na + b\sum_{i=1}^{n} x_i$$

$$\sum_{i=1}^{n} x_i y_i = a \sum_{i=1}^{n} x_i + b \sum_{i=1}^{n} x_i^2$$

Formulas to find a and b

Recall that the means of x and y are given by

$$\bar{x} = \frac{1}{n} \sum x_i$$
$$\bar{y} = \frac{1}{n} \sum y_i$$

and also that the variance of x is given by

$$\text{Var}(x) = \frac{1}{n-1} \Sigma (x_i - \bar{x}_i)^2.$$

The *covariance of x and y*, denoted by Cov(x, y) is defined as

$$\text{Cov}(x, y) = \frac{1}{n-1} \Sigma (x_i - \bar{x})(y_i - \bar{y})$$

It can be shown that the values of a and b can be computed using the following formulas:

$$b = \frac{\text{Cov}(x, y)}{\text{Var}(x)}$$
$$a = \bar{y} - b\bar{x}$$

Remarks

It is interesting to note why the least squares method discussed above is christened as "ordinary" least squares method. Several different variants of the least squares method have been developed over the years. For example, in the *weighted least squares* method, the coefficients a and b are estimated such that the weighted sum of squares of errors

$$E = \sum_{i=1}^{n} w_i [y_i - (a + bx_i)]^2,$$

for some positive constants $w_1, \ldots, w_n$, is minimum. There are also methods known by the names *generalised least squares* method, *partial least squares* method, *total least squares* method, etc. The reader may refer to *Wikipedia*, a free online encyclopedia, to obtain further information about these methods.

The OLS method has a long history. The method is usually credited to Carl Friedrich Gauss (1795), but it was first published by Adrien-Marie Legendre (1805).

Example

Obtain a linear regression for the data in Table 7.2 assuming that y is the independent variable.

x	1.0	2.0	3.0	4.0	5.0
y	1.00	2.00	1.30	3.75	2.25

Table 7.2: Example data for simple linear regression

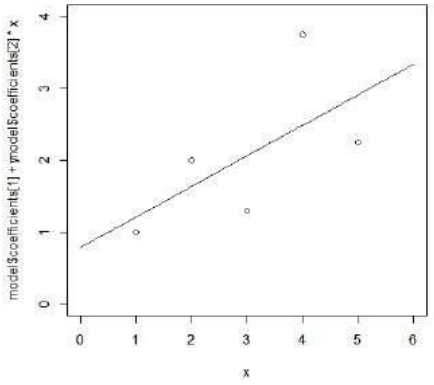


Figure 7.2: Regression model for Table 7.2

Solution

In the usual notations of simple linear regression, we have

$$n = 5$$

$$\bar{x} = \frac{1}{5}(1.0 + 2.0 + 3.0 + 4.0 + 5.0)$$

$$= 3.0$$

$$\bar{y} = \frac{1}{5}(1.00 + 2.00 + 1.30 + 3.75 + 2.25)$$

$$= 2.06$$

$$\text{Cov}(x, y) = \frac{1}{4}[(1.0 - 3.0)(1.00 - 2.06) + ... + (5.0 - 3.0)(2.25 - 2.06)]$$

$$= 1.0625$$

$$\text{Var}(x) = \frac{1}{4}[(1.0 - 3.0)^2 + ... + (5.0 - 3.0)^2]$$

$$= 2.5$$

$$b = \frac{1.0625}{2.5}$$

$$= 0.425$$

$$a \quad 2.06 \; 0.425 \; 3x0$$

$$= 0.785$$

Therefore, the linear regression model for the data is

$$y = 0.785 + 0.425x. \tag{7.1}$$

Remark

Figure 7.2 in page 76 shows the data in Table 7.2 and the line given by Eq. (7.1). The figure was created using R.

Polynomial regression

Let x be the independent predictor variable and y the dependent variable. Assume that we have a set of observed values of x and y as in Table 7.1 in page 74.

A polynomial regression model defines the relationship between x and y by an equation in the following form:

$$y = a_0 + a_1 x + a_2 x_2 + \ldots + a_k x^k.$$

To determine the optimal values of the parameters $a_0, a_1, \ldots, a_k$ the method of ordinary least squares is used. The values of the parameters are those values which minimizes the sum of squares:

$$E = \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + a_2 x_{2i} + \ldots + a_k x_i^k)]^2.$$

The optimal values of the parameters are obtained by solving the following system of equations:

$$\frac{\partial E}{\partial a_i} = 0, \quad i = 0, 1, \ldots, k. \tag{7.2}$$

Let the values of values of the parameters which minimizes E be

$$a_i = a_i, \quad i = 0, 1, 2, \ldots, n. \tag{7.3}$$

Simplifying Eq. (7.2) and using Eq. (7.3), we can see that the values of a_i can be obtained by solving the the following system of $(k+1)$ linear equations:

$$\begin{aligned}
\Sigma\, y_i &= a_0 n + a_1 (\Sigma\, x_i) + \ldots + a_k (\Sigma\, x_i^k) \\
\Sigma\, y_i x_i &= a_0 (\Sigma\, x_i) + a_1 (\Sigma\, x_i^2) + \ldots + a_k (\Sigma\, x_i^{k+1}) \\
\Sigma\, y_i x_i^2 &= a_0 (\Sigma\, x_i^2) + a_1 (\Sigma\, x_i^3) + \ldots + a_k (\Sigma\, x_i^{k+2}) \\
&\vdots \\
\Sigma\, y_i x_i^k &= a_0 (\Sigma\, x_i^k) + a_1 (\Sigma\, x_i^{k+1}) + \ldots + a_k (\Sigma\, x_i^{2k})
\end{aligned}$$

Solving this system of linear equations we get the optimal values for the parameters.

Remarks

The linear system of equations to find a_i's, has a compact matrix representation. We write:

$$D = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^k \\ 1 & x_2 & x_2^2 & \ldots & x_2^k \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \ldots & x_n^k \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \vec{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix}$$

Then we have

$$\vec{a} = (D^T D)^{-1} D^T \vec{y},$$

where the superscript T denotes the transpose of the matrix.

Example

Find a quadratic regression model for the following data:

x	3	4	5	6	7
y	2.5	3.2	3.8	6.5	11.5

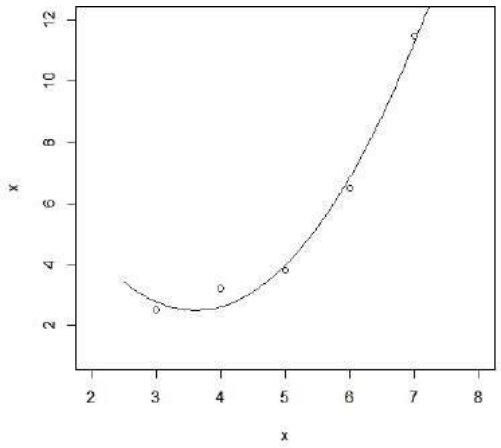


Figure 7.3: Plot of quadratic polynomial model

Solution

Let the quadratic regression model be

$$y = a_0 + a_1 x + a_2 x_2.$$

The values of a_0, a_1 and a_2 which minimises the sum of squares of errors are a_0, a_1 and a_2 which satisfy the following system of equations:

$$\begin{aligned}
\Sigma y &= na + a(\quad x) + a(x \Sigma_i^2 \\
\Sigma y_i x_i &= a_0(\Sigma x_i) + a_2(\Sigma x_2) + a_2(\Sigma x_3) \\
\Sigma y_i x_i^2 &= a_0(\Sigma x_i^2) + a_1(\Sigma x_i^3) + a_2(\Sigma x_i^4)
\end{aligned}$$

Using the given data we have

$$\begin{aligned}
27.5 &= 5a_0 + 25a_1 + 135a_2 \\
158.8 &= 25a + 135a + 775a_1 \\
966.2 &= 135a_0 + 775a_1 + 4659a_2
\end{aligned}$$

Solving this system of equations we get

$$\begin{aligned}
a_0 &= 12.4285714 \\
a_1 &= -5.5128571 \\
a_2 &= 0.7642857
\end{aligned}$$

The required quadratic polynomial model is

$$y = 12.4285714 - 5.5128571x + 0.7642857x_2.$$

Figure 7.3 shows plots of the data and the quadratic polynomial model.

Multiple linear regression

We assume that there are N independent variables x_1, x_2, ..., x_N. Let the dependent variable be y. Let there also be n observed values of these variables:

Variables (features)	Values (examples)			
			...	
x_1	x_{11}	x_{12}		x_{1n}
x_2	x_{21}	x_{22}	...	x_{2n}
...				
x_N	x_{N1}	x_{N2}	...	x_{Nn}
y (outcomes)	y_1	y_2		y_n

Table 7.3: Data for multiple linear regression

The multiple linear regression model defines the relationship between the N independent variables and the dependent variable by an equation of the following form:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_N x_N$$

As in simple linear regression, here also we use the ordinary least squares method to obtain the optimal estimates of $\beta_0, \beta_1, .., \beta_N$. The method yields the following procedure for the computation of these optimal estimates. Let

$$X = \begin{pmatrix} 1 & x_{11} & x_{21} & \dots & x_{N1} \\ 1 & x_{12} & x_{22} & \dots & x_{N2} \\ \vdots & & & & \\ 1 & x_{1n} & x_{2n} & \dots & x_{Nn} \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad B = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_N \end{pmatrix}$$

Then it can be shown that the regression coefficients are given by

$$B = (X^T X)^{-1} X^T Y$$

Example

Example

Fit a multiple linear regression model to the following data:

x_1	1	1	2	0
x_2	1	2	2	1
y	3.25	6.5	3.5	5.0

Table 7.4: Example data for multi-linear regression

Solution

In this problem, there are two independent variables andfour sets of values of the variables. Thus, in the notations used above, we have $n = 2$ and $N = 4$. The multiple linear regression model for this problem has the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

The computations are shown below.

$$X = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \\ 1 & 0 & 1 \end{pmatrix}, \quad Y = \begin{pmatrix} 3.25 \\ 6.5 \\ 3.5 \\ 5.0 \end{pmatrix}, \quad B = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix}$$

$$X^T X = \begin{bmatrix} 4 & 4 & 6 \\ 4 & 6 & 7 \\ 6 & 7 & 10 \end{bmatrix}$$

$$(X^T X)^{-1} = \begin{bmatrix} \frac{11}{4} & \frac{1}{2} & -2 \\ \frac{1}{4} & 1 & -1 \\ -2 & -1 & 2 \end{bmatrix}$$

$$B = (X^T X)^{-1} X^T Y$$

$$= \begin{bmatrix} 2.0625 \\ -2.3750 \\ 3.2500 \end{bmatrix}$$

The required model is

$$y = 2.0625 - 2.3750x_1 + 3.2500x_2.$$

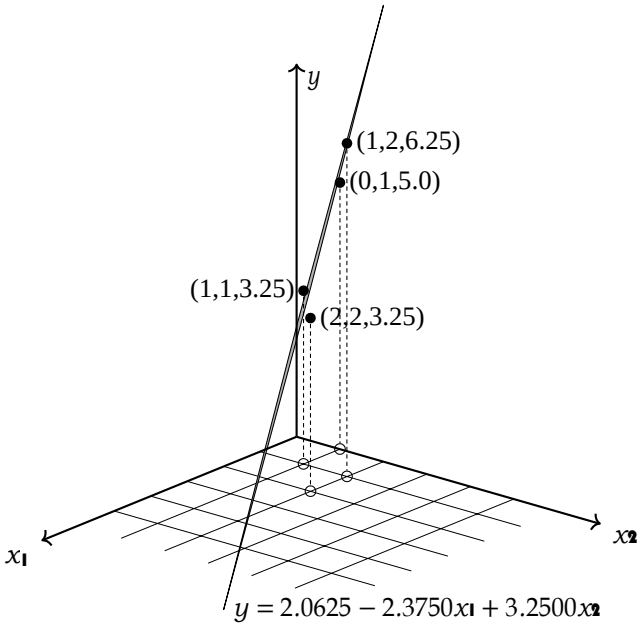


Figure 7.4: The regression plane for the data in Table 7.4

Sample questions

(a) Short answer questions

1. What are the different types of regression.
2. Is regression a supervised learning? Why?
3. Explain the ordinary least squares method for regression.
4. What are linear, multinomial and polynomial regressions.
5. If model used for regression is

$$y = a + (b - x)x_1 x_2,$$

is it a multinomial regression? If not, what type of regression is it?

6. What does the line of regression tell you?

(b) Long answer questions

1. Discuss linear regression with an example.

2. In the table below, the x_i row shows scores in an aptitude test. Similarly, the y_i row shows statistics grades. If a student made an 80 on the aptitude test, what grade would we expect her to make in statistics?

Student i	1	2	3	4	5
x_i	95	85	80	70	60
y_i	85	95	70	65	70

3. Use the following data to construct a linear regression model for the auto insurance premium as a function of driving experience.

Driving experience (in years)	5	2	12	9	15	6	25	16
Monthly auto insurance premium ($)	64	87	50	71	44	56	42	60

4. Determine the regression equation by finding the regression slope coefficient and the intercept value using the following data.

x	55	60	65	70	80
y	52	54	56	58	62

5. The following table contains measurements of yield from an experiment done at five different temperature levels. The variables are y = yield and x = temperature in degrees Fahrenheit. Compute a second degree polynomial regression model to predict the yield given the temperature.

Temperature (x)	Yield (y)
50	3.0
70	2.7
80	2.6
90	2.9
100	3.3

6. An experiment was done to assess how moisture content and sweetness of a pastry product affect a tasterâĂŹs rating of the product. The following table summarises the findings.

Rating	Moisture	Sweetness
64	4	2
73	4	4
61	4	2
76	4	4
72	6	2
80	6	4
71	6	2
83	6	4
83	8	2
89	8	4
86	8	2
93	8	4
88	10	2
95	10	4
94	10	2
100	10	4

Compute a linear regression model to predict the rating of the pastry product.

7. The following data contains the Performance IQ scores (PIQ) (in appropriate scales), brain sizes (in standard units), heights (in inches) and weights (in pounds) of 15 American college students. Obtain a linear regression model to predict the PIQ given the values of the other features.

PIQ	Brain	Height	Weight
124	81.69	64.5	118
150	103.84	73.3	143
128	96.54	68.8	172
134	95.15	65.0	147
110	92.88	69.0	146
131	99.13	64.5	138
98	85.43	66.0	175
84	90.49	66.3	134
147	95.55	68.8	172
124	83.39	64.5	118
128	107.95	70.0	151
124	92.41	69.0	155
147	85.65	70.5	155
90	87.89	66.0	146
96	86.54	68.0	135

8. Use the following data to generate a linear regression model for annual salary as function of GPA and number of months worked.

Example no.	Annual salary ($)	GPA	Months worked
1	20000	2.8	48
2	24500	3.4	24
3	23000	3.2	24
4	25000	3.8	24
5	20000	3.2	48
6	22500	3.4	36
7	27500	4.0	24
8	19000	2.6	48
9	24000	3.2	36
10	28500	3.8	12

Module 4

Decision trees

"Decision tree learning is a method for approximating discrete valued target functions, in which the learned function is represented by a decision tree. Decision tree learning is one of the most widely used and practical methods for inductive inference." ([4] p.52)

Decision tree: Example

Consider the following situation. Somebody is hunting for a job. At the very beginning, he decides that he will consider only those jobs for which the monthly salary is at least Rs.50,000. Our job hunter does not like spending much time traveling to place of work. He is comfortable only if the commuting time is less than one hour. Also, he expects the company to arrange for a free coffee every morning! The decisions to be made before deciding to accept or reject a job offer can be schematically represented as in Figure 8.6. This figure represents a *decision tree*[1].

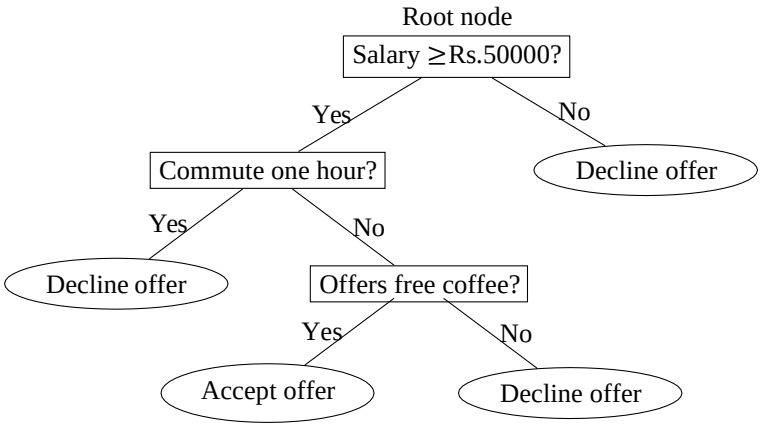


Figure 8.1: Example for a decision tree

Here, the term "tree" refers to the concept of a tree in graph theory in mathematics[2]. *In graph theory, a tree is defined as an undirected graph in which any two vertices are connected by exactly one path.* Using the conventions of graph theory, the decision tree shown in Figure 8.6 can be represented as a graph-theoretical tree as in Figure 8.2. Since a decision tree is a graph-theoretical tree, all terminology related to graph-theoretical trees can be applied to describe decision trees also. For example, in Figure 8.6, the nodes or vertices shown as ellipses are called the *leaf nodes*. All other nodes, except the root node, are called the *internal nodes*.

[1]In such diagrams, the "tree" is shown upside down with the root node at the top and all the leaves at the bottom.

[2]The term "tree" was coined in 1857 by the British mathematician Arthur Cayley (see Wikipedia).

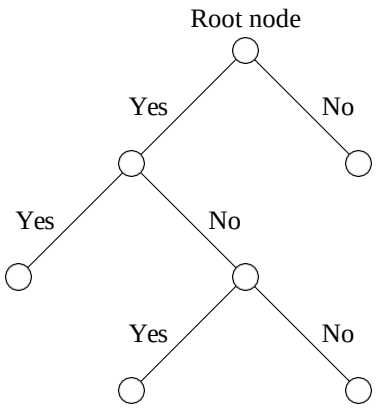


Figure 8.2: The graph-theoretical representation of the decision tree in Figure 8.6

Two types of decision trees

There are two types of decision trees.

1. **Classification trees**

   Tree models where the target variable can take a discrete set of values are called *classification trees*. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

2. **Regression trees**

   Decision trees where the target variable can take continuous values (real numbers) like the price of a house, or a patient's length of stay in a hospital, are called *regression trees*.

Classification trees

We illustrate the concept with an example.

Example

Data

Nam	Features				Class label
	gives birth	aquatic animal	aerial animal	has legs	
human	yes	no	no	yes	mammal
python	no	no	no	no	reptile
salmon	no	yes	no	no	fish
frog	no	semi	no	yes	amphibian
bat	yes	no	yes	yes	bird
pigeon	no	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Table 8.1: The vertebrate data set

Consider the data given in Table 8.1 which specify the features of certain vertebrates and the class to which they belong. For each species, four features have been identified: “gives birth”, ”aquatic animal”, “aerial animal” and “has legs”. There are five class labels, namely, “amphibian”, “bird”, “fish”, “mammal” and “reptile”. The problem is how to use this data to identify the class of a newly discovered vertebrate.

Construction of the tree

Step 1

We split the set of examples given in Table 8.1 into disjoint subsets according to the values of the feature “gives birth”. Since there are only two possible values for this feature, we have only two subsets: One subset consisting of those examples for which the value of “gives birth” is “yes” and one subset for which the value is “no”. The former is given in Table 8.2 and the latter in Table 8.3. This stage of the classification can be represented as in Figure 8.3.

Name	Gives birth	Aquatic animal	Aerial animal	Has legs	Class label
human	yes	no	no	yes	mammal
bat	yes	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish

Table 8.2: The subset of Table 8.1 with “gives birth” = ”yes"

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
python	no	no	no	no	reptile
salmon	no	yes	no	no	fish
frog	no	semi	no	yes	amphibian
pigeon	no	no	yes	yes	bird
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Table 8.3: The subset of Table 8.1 with “gives birth” = ”no"

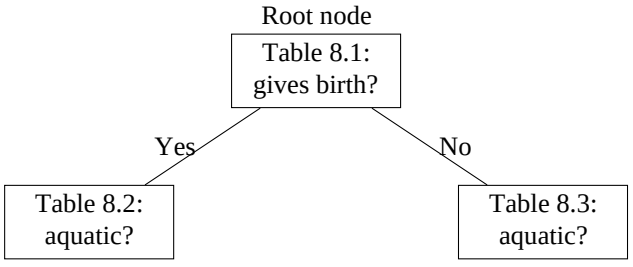


Figure 8.3: Classification tree

Step 2

We now consider the examples in Table 8.2. We split these examples based on the values of the feature “aquatic animal”. There are three possible values for this feature. However, only two of

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
human	yes	no	no	yes	mammal
bat	yes	no	yes	yes	bird
cat	yes	no	no	yes	mammal

Table 8.5: The vertebrate data set

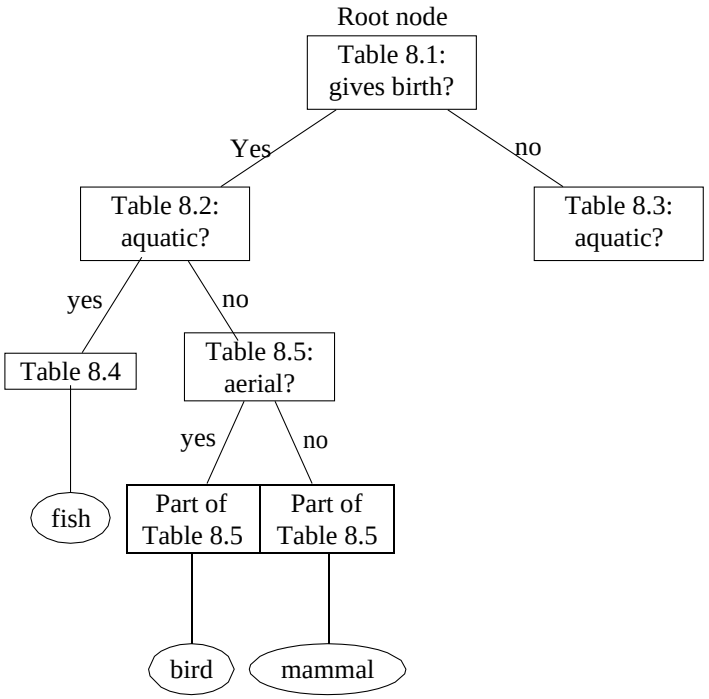


Figure 8.4: Classification tree

these appear in Table 8.2. Accordingly, we need consider only two subsets. These are shown in Tables 8.4 and 8.5.

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
shark	yes	yes	no	no	fish

Table 8.4: The vertebrate data set

- Table 8.4 contains only one example and hence no further splitting is required. It leads to the assignment of the class label “fish”.
- The examples in Table 8.5 need to be split into subsets based on the values of “aerial animal”. It can be seen that these subsets immediately lead to unambiguous assignment of class labels: The value of “no” leads to “mammal” and the value “yes” leads to ”bird”.

At this stage, the classification tree is as shown in Figure 8.4

Step 3

Next we consider the examples in Table 8.3 and split them into disjoint subsets based on the values of "aquatic animal". We get the examples in Table 8.6 for "yes", the examples in Table **??** for "no" and the examples in Table **??** for "semi". We now split the resulting subsets based on the values of

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
salmon	no	yes	no	no	fish

Table 8.6: The vertebrate data set

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
frog	no	semi	no	yes	amphibian
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Table 8.7: The vertebrate data set

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
python	no	no	no	no	reptile
pigeon	no	no	yes	yes	bird

Table 8.8: The vertebrate data set

"has legs", etc. Putting all these together, we get the the diagram in Figure 8.5 as the classification tree for the data in Table 8.1.

Classification tree in rule format

The classification tree shown in Figure 8.5 can be presented as a set of rules in the form of an algorithm.

Algorithm for classification of vertebrates

```
1.  if give birth = "yes" then
2.      if aquatic = "yes" then
3.          return class = "fish"
4.      else
5.          if aerial = "yes" then
6.              return class = "bird"
7.          else
8.              return class = "mammal"
9.          end if
10.     end if
11. else
12.     if aquatic = "yes" then
13.         return class = "fish"
```

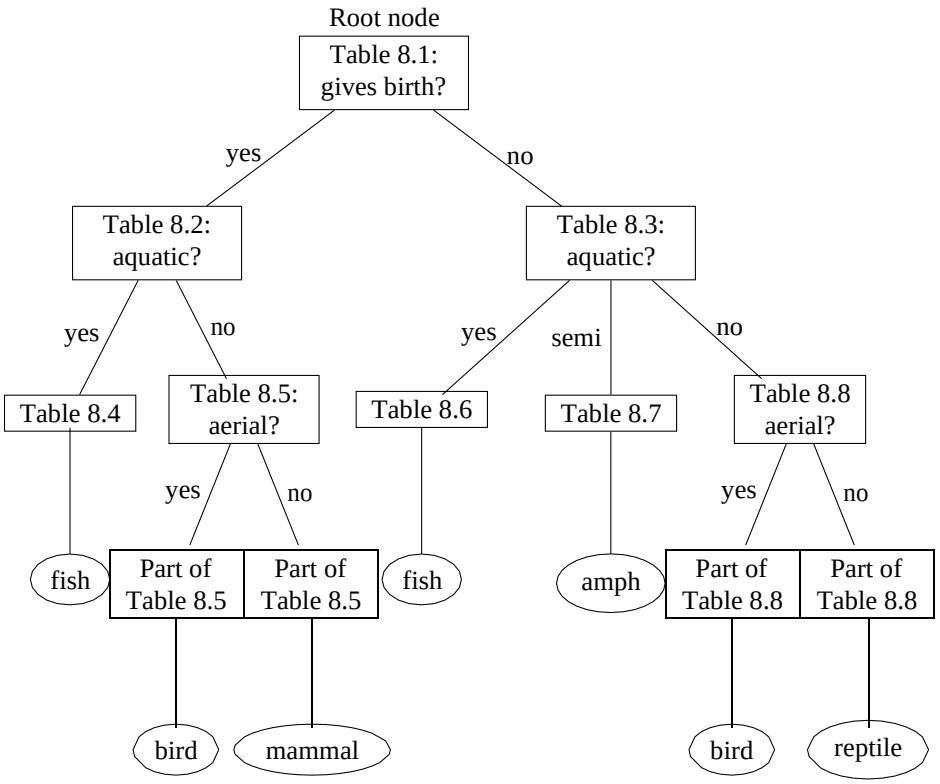


Figure 8.5: Classification tree

```
14.       end if
15.       if aquatic = "semi" then
16.             return class = "amphibian"
17.       else
18.             if aerial = "yes" then
19.                   return class = "amphibian"
20.             else
21.                   return class = "reptile"
22.             end if
23.       end if
24. end  if
```

Some remarks

1. On the elements of a classification tree

The various elements in a classification tree are identified as follows.

- Nodes in the classification tree are identified by the feature names of the given data.
- Branches in the tree are identified by the values of features.
- The leaf nodes identified by are the class labels.

2. On the order in which the features are selected

In the example discussed above, initially we chose the feature "gives birth" to split the data set into disjoint subsets and then the feature "aquatic animal", and so on. There was no theoretical justification for this choice. We could as well have chosen the feature "aquatic animal", or any other feature, as the initial feature for splitting the data. The classification tree depends on the order in which the features are selected for partitioning the data.

3. Stopping criteria

A real-world data will contain much more example record than the example we considered earlier. In general, there will be a large number of features each feature having several possible values. Thus, the corresponding classification trees will naturally be more complex. In such cases, it may not be advisable to construct all branches and leaf nodes of the tree. The following are some of commonly used criteria for stopping the construction of further nodes and branches.

- All (or nearly all) of the examples at the node have the same class.
- There are no remaining features to distinguish among the examples.
- The tree has grown to a predefined size limit.

Feature selection measures

If a dataset consists of n attributes then deciding which attribute is to be to placed at the root or at different levels of the tree as internal nodes is a complicated problem. It is not enough that we just randomly select any node to be the root. If we do this, it may give us bad results with low accuracy.

The most important problem in implementing the decision tree algorithm is deciding which features are to be considered as the root node and at each level. Several methods have been developed to assign numerical values to the various features such that the values reflect the relative importance of the various features. These are called the *feature selection measures*. Two of the popular feature selection measures are *information gain* and *Gini index*. These are explained in the next section.

Entropy

The degree to which a subset of examples contains only a single class is known as *purity*, and any subset composed of only a single class is called a *pure* class. Informally, entropy[3] is a measure of "impurity" in a dataset. Sets with high entropy are very diverse and provide little information about other items that may also belong in the set, as there is no apparent commonality.

Entropy is measured in bits. If there are only two possible classes, entropy values can range from 0 to 1. For n classes, entropy ranges from 0 to $\log_2(n)$. In each case, the minimum value indicates that the sample is completely homogeneous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

Definition

Consider a segment S of a dataset having c number of class labels. Let p_i be the proportion of examples in S having the i th class label. The entropy of S is defined as

$$\text{Entropy } (S) = \sum_{i=1}^{c} -p_i \log_2(p_i) .$$

[3]From German Entropie "measure of the disorder of a system," coined in 1865 (on analogy of Energie) by German physicist Rudolph Clausius (1822-1888), in his work on the laws of thermodynamics, from Greek entropia "a turning toward," from en "in" + trope "a turning, a transformation,"

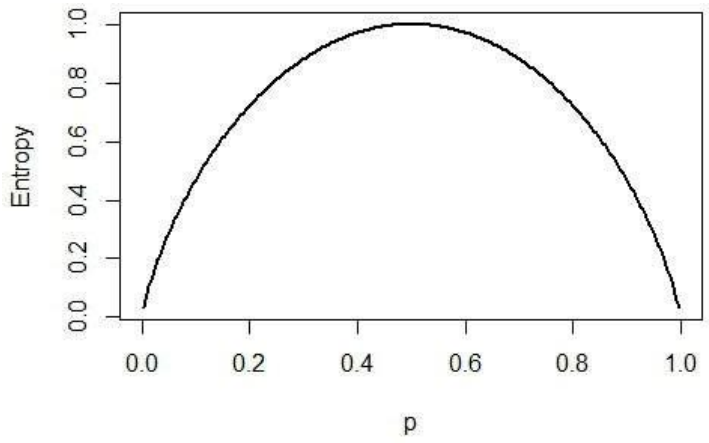


Figure 8.6: Plot of p vs. Entropy

Remark

In the expression for entropy, the value of $0 \log_2 0$ is taken as zero.

Special case

Let the data segment S has only two class labels, say, "yes" and "no". If p is the proportion of examples having the label "yes" then the proportion of examples having label "no" will be $1 - p$. In this case, the entropy of S is given by

$$\text{Entropy}(S) = -p \log_2(p) - (1-p) \log_2(1-p) .$$

If we plot the values of graph of Entropy(S) for all possible values of p, we get the diagram shown in Figure 8.6[4].

Examples

Let "xxx" be some class label. We denote by p_{xxx} the proportion of examples with class label "xxx".

1. **Entropy of data in Table 8.1**

   Let S be the data in Table 8.1. The class labels are "amphi", "bird", "fish", "mammal" and "reptile". In S we have the following numbers.

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

   Therefore, we have:

$$\text{Entropy}(S) = \sum_{\text{for all classes "xxx"}} -p_{xxx} \log_2(p_{xxx})$$

[4]Plot created using R language.

$$
\begin{aligned}
&= p_{\text{amphi}} \log_2(p_{\text{amphi}}) - p_{\text{bird}} \log_2(p_{\text{bird}}) \\
&\quad - p_{\text{fish}} \log_2(p_{\text{fish}}) - p_{\text{mammal}} \log_2(p_{\text{mammal}}) \\
&\quad - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\
&= -(3/10)\log_2(3/10) - (2/10)\log_2(2/10) \\
&\quad - (2/10)\log_2(2/10) - (2/10)\log_2(2/10) \\
&\quad - (1/10)\log_2(1/10) \\
&= 2.2464
\end{aligned}
$$

2. Entropy of data in Table 8.2

Consider the segment S of the data in Table 8.1 given in Table 8.2. For quick reference, the table has been reproduced below:

Name	Gives birth	Aquatic animal	Aerial animal	Has legs	Class label
human	yes	no	no	yes	mammal
bat	yes	no	yes	yes	bird
cat	yes	no	no	yes	mammal
shark	yes	yes	no	no	fish

Three class labels appear in this segment, namely, “bird”, “fish” and “mammal”. We have:

Number of examples with class label “bird”	1
Number of examples with class label “fish”	1
Number of examples with class label “mammal”	2
Total number of examples	4

Therefore we have

$$
\begin{aligned}
\text{Entropy}(S) &= \sum_{\text{for all classes “xxx”}} -p_{\text{xxx}} \log_2(p_{\text{xxx}}) \\
&= p_{\text{bird}} \log_2(p_{\text{bird}}) - p_{\text{fish}} \log_2(p_{\text{fish}}) \\
&\quad - p_{\text{mammal}} \log_2(p_{\text{mammal}}) \\
&= -(1/4)\log_2(1/4) - (1/4)\log_2(1/4) - (2/4)\log_2(2/4) \\
&\quad - (1/4) \times (-2) - (1/4) \times (-2) - (2/4) \times (-1) \\
&= 1.5
\end{aligned} \tag{8.1}
$$

3. Entropy of data in Table 8.3

Consider the segment S of the data in Table 8.1 given in Table 8.3. For quick reference, the table has been reproduced below:

Name	gives birth	aquatic animal	aerial animal	has legs	Class label
python	no	no	no	no	reptile
salmon	no	yes	no	no	fish
frog	no	semi	no	yes	amphibian
pigeon	no	no	yes	yes	bird
turtle	no	semi	no	yes	amphibian
salamander	no	semi	no	yes	amphibian

Four class labels appear in this segment, namely, “amphi”, “bird”, “fish” and “reptile”. We have:

Number of examples with class label "amphi"	3
Number of examples with class label "bird"	1
Number of examples with class label "fish"	1
Number of examples with class label "reptile"	1
Total number of examples	6

Therefore, we have:

$$\begin{aligned}\text{Entropy}(S) &= \sum_{\text{for all classes "xxx"}} -p_{\text{xxx}} \log_2(p_{\text{xxx}}) \\ &= p_{\text{amphi}} \log_2(p_{\text{amphi}}) - p_{\text{bird}} \log_2(p_{\text{bird}}) - p_{\text{fish}} \log_2(p_{\text{fish}}) \\ &\quad - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\ &= -(3/6)\log_2(3/6) - (1/6)\log_2(1/6) - (1/6)\log_2(1/6) \\ &\quad - (1/6)\log_2(1/6) \\ &= 1.7925 \end{aligned} \tag{8.2}$$

Information gain

Definition

Let S be a set of examples, A be a feature (or, an attribute), S_v be the subset of S with $A = v$, and Values (A) be the set of all possible values of A. Then the *information gain of an attribute A relative to the set S*, denoted by Gain (S, A), is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

where $|S|$ denotes the number of elements in S.

Example 1

Consider the data S given in Table 8.1. We have have already seen that

$$|S| = 10$$
$$\text{Entropy}(S) = 2.2464.$$

We denote the information gain corresponding to the feature "xxx" by Gain$(S,$ xxx$)$.

1. **Computation of Gain** $(S,$ **gives birth**$)$

$$\begin{aligned} A_1 &= \text{gives birth} \\ \text{Values of } A_1 &= \{\text{"yes", "no"}\} \\ S_{A_1=\text{yes}} &= \text{Data in Table 8.2} \\ |S_{A_1=\text{yes}}| &= 4 \\ \text{Entropy}(S_{A_1=\text{yes}}) &= 1.5 \text{ (See Eq.(8.1))} \\ S_{A_1=\text{no}} &= \text{Data in Table 8.3} \\ |S_{A_1=\text{no}}| &= 6 \\ \text{Entropy}(S_{A_1=\text{no}}) &= 1.7925 \text{ (See Eq.(8.2))} \end{aligned}$$

Now we have

$$\text{Gain}(S, A_1) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A_1)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

$$= \text{Entropy}(S) - \frac{|S_{A_1=\text{yes}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{yes}}) - \frac{|S_{A_1=\text{no}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{no}})$$

$$= 2.2464 - (4/10) \times 1.5 - (6/10) \times 1.7925$$

$$= 0.5709$$

2. **Computation of Gain(S, aquatic)**

A_2 = aquatic

Values of A_2 = {"yes", "no", "semi"}

$S_{A_2=\text{yes}}$ = See Table 8.1

$|S_{A_2=\text{yes}}| = 2$

$$\text{Entropy}(S_{A_2=\text{yes}}) = -p_{\text{fish}} \log_2(p_{\text{fish}}) - (2/2)\log_2(2/2) = 0$$

$S_{A_2=\text{no}}$ = See Table 8.1

$|S_{A_2=\text{no}}| = 5$

$$\text{Entropy}(S_{A_2=\text{no}}) = -p_{\text{mammal}} \log_2(p_{\text{mammal}}) - p_{\text{reptile}} \log_2(p_{\text{reptile}}) - p_{\text{bird}} \log(p_{\text{bird}})$$

$$= -(2/5) \times \log_2(2/5) - (1/5) \times \log_2(1/5) - (2/5) \times \log_2(2/5) = 1.5219$$

$S_{A_2=\text{semi}}$ = See Table 8.1

$|S_{A_2=\text{semi}}| = 3$

$$\text{Entropy}(S_{A_2=\text{semi}}) = -p_{\text{amphi}} \log_2(p_{\text{amphi}}) - (3/3) \times \log_2(3/3) = 0$$

$$\text{Gain}(S, A_2) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A_2)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v)$$

$$= \text{Entropy}(S) - \frac{|S_{A_1=\text{yes}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{yes}}) - \frac{|S_{A_1=\text{no}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{no}}) - \frac{|S_{A_1=\text{semi}}|}{|S|} \times \text{Entropy}(S_{A_1=\text{semi}})$$

$$= 2.2464 - (2/10) \times 0 - (5/10) \times 1.5219 - (3/3) \times 0$$

$$= 1.48545$$

3. **Computations of Gain(S, aerial animal) and Gain(S, has legs)**

These are left as exercises.

Gini indices

The Gini split index of a data set is another feature selection measure in the construction of classification trees. This measure is used in the CART algorithm.

8.7.1 Gini index

Consider a data set S having r class labels $c_1, \ldots, c_r$. Let p_i be the proportion of examples having the class label c_i. The Gini index of the data set S, denoted by Gini(S), is defined by

$$\text{Gini}(S) = 1 - \sum_{i=1}^{r} p_i^2.$$

Example

Let S be the data in Table 8.1. There are four class labels "amphi", "bird", "fish", "mammal" and "reptile". The numbers of examples having these class labels are as follows:

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

The Gini index of S is given by

$$\begin{aligned}\text{Gini}(S) &= 1 - \sum_{i=1}^{r} p_i^2 \\ &= 1 - (3/10)^2 - (2/10)^2 - (2/10)^2 - (2/10)^2 - (1/10)^2 \\ &= 0.78\end{aligned}$$

8.7.2 Gini split index

Let S be a set of examples, A be a feature (or, an attribute), S_v be the subset of S with $A = v$, $v \in$ Values (A) and Values (A) be the set of all possible values of A. Then the *Gini split index of A relative to S*, denoted by $\text{Gini}_{\text{split}}(S, A)$, is defined as

$$\text{Gini}_{\text{split}}(S, A) = \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Gini}(S_v).$$

where $|S|$ denotes the number of elements in S.

Gain ratio

The *gain ratio* is a third feature selection measure in the construction of classification trees.

Let S be a set of examples, A a feature having c different values and let the set of values of A be denoted by Values(A). We defined the information gain of A relative to S, denoted by Gain(S, A), by

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

We now define the *split information* of A relative to S, dented by SplitInformation(S, A), by

$$\text{SplitInformation}(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where $S_1, \ldots S_c$ are the c subsets of examples resulting from partitioning S into the c values of the attribute A. The *gain ratio* of A relative to S, denoted by GainRatio(S, A), by

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}.$$

8.8.1 Example

Consider the data S given in Table 8.1. Let A denote the attribute "gives birth".We have have already seen that

$$|S| = 10$$
$$\text{Entropy}\ (S) = 2.2464$$
$$\text{Gain}(S, A) = 0.5709$$

Now we have

$$\begin{aligned}\text{SplitInformation}(S,A) &= -\frac{|S_{\text{yes}}|}{|S|}\log_2\frac{|S_{\text{yes}}|}{|S|} - \frac{|S_{\text{no}}|}{|S|}\log_2\frac{|S_{\text{no}}|}{|S|} \\ &= -\frac{4}{10}\times\log_2\frac{4}{10} - \frac{6}{10}\times\log_2\frac{6}{10} \\ &= 0.9710 \\ \text{GainRatio} &= \frac{0.5709}{0.9710} \\ &= 0.5880\end{aligned}$$

In a similar way we can compute the gain ratios Gain(S, "aquatic"), Gain(S, "aerial") and Gain(S, "has legs").

Decision tree algorithms

Outline

Decision tree algorithm: Outline

1. Place the "best" feature (or, attribute) of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for a feature.
3. Repeat Step 1 and Step 2 on each subset until we find leaf nodes in all the branches of the tree.

Some well-known decision tree algorithms

1. ID3 (Iterative Dichotomiser 3) developed by Ross Quinlan
2. C4.5 developed by Ross Quinlan
3. C5.0 developed by Ross Quinlan
4. CART (Classification And Regression Trees)
5. 1R (One Rule) developed by Robert Holte in 1993.
6. RIPPER (Repeated Incremental Pruning to Produce Error Reduction) Introduced in 1995 by William W. Cohen.

As an example of decision tree algorithms, we discuss the details of the ID3 algorithm and illustrate it with an example.

The ID3 algorithm

Ross Quinlan, while working at University of Sydney, developed the ID3 (Iterative Dichotomiser 3)[5] algorithm and published it in 1975.

Assumptions

- The algorithm uses information gain to select the most useful attribute for classification.
- We assume that there are only two class labels, namely, " +" and "−". The examples with class labels "+" are called positive examples and others negative examples.

The algorithm

Notations

The following notations are used in the algorithm:

S	The set of examples
C	The set of class labels
F	The set of features
A	An arbitrary feature (attribute)
Values(A)	The set of values of the feature A
v	An arbitrary value of A
S_v	The set of examples with $A = v$
Root	The root node of a tree

Algorithm ID3(S, F , C)

1. Create a root node for the tree.
2. **if** (all examples in S are positive) **then**
3. **return** single node tree Root with label "+"
4. **end if**
5. **if** (all examples are negative) **then**
6. **return** single node tree Root with label "–"
7. **end if**
8. **if** (number of features is 0) **then**
9. **return** single node tree Root with label equal to the most common class label.
10. **else**
11. Let A be the feature in F with the highest information gain.
12. Assign A to the Root node in decision tree.
13. **for all** (values v of A) **do**
14. Add a new tree branch below Root corresponding to v.
15. **if** (S_v is empty) **then**
16. Below this branch add a leaf node with label equal to the most common class label in the set S.
17. **else**
18. Below this branch add the subtree formed by applying the same algorithm ID3 with the values ID3(S_v, C, $F - \{A\}$)
19. **end if**
20. **end for**
21. **end if**

---

[5] *dichotomy*: A division into two parts or classifications especially when they are sharply distinguished or opposed

Example

Problem

Use ID3 algorithm to construct a decision tree for the data in Table 8.9.

Day	outlook	temperature	humidity	wind	playtennis
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D3	overcast	hot	high	weak	yes
D4	rain	mild	high	weak	yes
D5	rain	cool	normal	weak	yes
D6	rain	cool	normal	strong	no
D7	overcast	cool	normal	strong	yes
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D10	rain	mild	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes
D14	rain	mild	high	strong	no

Table 8.9: Training examples for the target concept "PlayTennis"

Solution

Note that, in the given data, there are four features but only two class labels (or, target variables), namely, "yes" and "no".

Step 1

We first create a root node for the tree (see Figure 8.7).

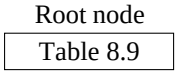


Figure 8.7: Root node of the decision tree for data in Table 8.9

Step 2

Note that not all examples are positive (class label "yes") and not all examples are negative (class label "no"). Also the number of features is not zero.

Step 3

We have to decide which feature is to be placed at the root node. For this, we have to calculate the information gains corresponding to each of the four features. The computations are shown below.

(i) Calculation of Entropy (S)

$$\begin{aligned}\text{Entropy}(S) &= -p_{\text{yes}}\log_2(p_{\text{yes}}) - p_{\text{no}}\log_2(p_{\text{no}}) \\ &\quad -(9/14)\times\log_2(9/14) - (5/14)\times\log_2(5/14) \\ &= 0.9405\end{aligned}$$

(ii) Calculation of Gain $(S, \text{outlook})$

The values of the attribute "outlook" are "sunny", " overcast" and "rain". We have to calculate Entropy (S_v) for $v = \text{sunny}$, $v = \text{overcast}$ and $v = \text{rain}$.

$$\begin{aligned}
\text{Entropy}(S_{\text{sunny}}) &= -(3/5) \times \log_2(3/5) - (2/5) \times \log_2(2/5) \\
&= 0.9710 \\
\text{Entropy}(S_{\text{overcast}}) &= -(4/4) \times \log_2(4/4) \\
&= 0 \\
\text{Entropy}(S_{\text{rain}}) &= -(3/5) \times \log_2(3/5) - (2/5) \times \log_2(2/5) \\
&= 0.9710
\end{aligned}$$

$$\begin{aligned}
\text{Gain}(S, \text{outlook}) &= \text{Entropy}(S) - \frac{|S_{\text{sunny}}|}{|S|} \times \text{Entropy}(S_{\text{sunny}}) \\
&\quad - \frac{|S_{\text{overcast}}|}{|S|} \times \text{Entropy}(S_{\text{overcast}}) \\
&\quad - \frac{|S_{\text{rain}}|}{|S|} \times \text{Entropy}(S_{\text{rain}}) \\
&= 0.9405 - (5/14) \times 0.9710 - (4/14) \times 0 \\
&\quad - (5/14) \times 0.9710 \\
&= 0.2469
\end{aligned}$$

(iii) Calculation of Gain $(S, \text{temperature})$

The values of the attribute "temperature" are "hot", "mild" and "cool". We have to calculate Entropy (S_v) for $v = \text{hot}$, $v = \text{mild}$ and $v = \text{cool}$.

$$\begin{aligned}
\text{Entropy}(S_{\text{hot}}) &= -(2/4) \times \log_2(2/4) - (2/4) \times \log_2(2/4) \\
&= 1.0000 \\
\text{Entropy}(S_{\text{mild}}) &= -(4/6) \times \log_2(4/6) - (2/6) \times \log_2(2/6) \\
&= 0.9186 \\
\text{Entropy}(S_{\text{cool}}) &= -(3/4) \times \log_2(3/4) - (1/4) \times \log_2(1/4) \\
&= 0.8113
\end{aligned}$$

$$\begin{aligned}
\text{Gain}(S, \text{temperature}) &= \text{Entropy}(S) - \frac{|S_{\text{hot}}|}{|S|} \times \text{Entropy}(S_{\text{hot}}) \\
&\quad - \frac{|S_{\text{mild}}|}{|S|} \times \text{Entropy}(S_{\text{mild}}) \\
&\quad - \frac{|S_{\text{cool}}|}{|S|} \times \text{Entropy}(S_{\text{cool}}) \\
&= 0.9405 - (4/14) \times 1.0000 - (6/14) \times 0.9186 \\
&\quad - (4/14) \times 0.8113 \\
&= 0.0293
\end{aligned}$$

(iv) Calculation of Gain $(S, \text{humidity})$ and Gain (S, wind)

The following information gains can be calculated in a similar way:

$$\text{Gain}(S, \text{humidity}) = 0.151$$
$$\text{Gain}(S, \text{wind}) = 0.048$$

Step 4

We find the highest information gain whic is the maximum among Gain(S, outlook), Gain(S, temperature), Gain(S, humidity) and Gain(S, wind). Therefore, we have:

$$\begin{aligned}\text{highest information gain} &= \max\{0.2469,\ 0.0293,\ 0.151, 0.048\} \\ &= 0.2469\end{aligned}$$

This corresponds to the feature "outlook". Therefore, we place "outlook" at the root node. We now split the root node in Figure 8.7 into three branches according to the values of the feature "outlook" as in Figure 8.8.

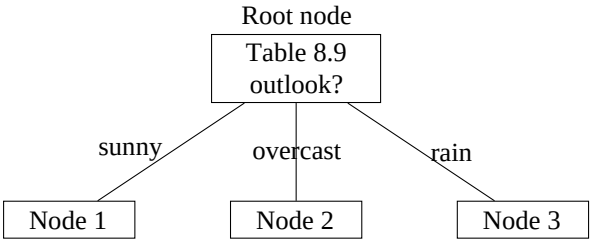


Figure 8.8: Decision tree for data in Table 8.9, after selecting the branching feature at root node

Step 5

Let $S^{(1)} = S_{\text{outlook=sunny}}$. We have $|S^{(1)}| = 5$. The examples in $S^{(1)}$ are shown in Table 8.10.

Day	outlook	temperature	humidity	wind	playtennis
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D11	sunny	mild	normal	strong	yes

Table 8.10: Training examples with outlook = "sunny"

$$\begin{aligned}\text{Gain}(S^{(1)}, \text{temp}) = \text{Entropy}(S^{(1)}) &- \frac{|S^{(1)}_{\text{temp = hot}}|}{|S^{(1)}|} \times \text{Entropy}(S^{(1)}_{\text{temp = hot}}) \\ &- \frac{|S^{(1)}_{\text{temp = mild}}|}{|S^{(1)}|} \times \text{Entropy}(S^{(1)}_{\text{temp = mild}}) \\ &- \frac{|S^{(1)}_{\text{temp = cool}}|}{|S^{(1)}|} \times \text{Entropy}(S^{(1)}_{\text{temp = cool}}) \\ = [-(2/5)\log_2(2/5) &- (3/5)\log_2(3/5)] \\ &- (2/5) \times [-(2/2)\log(2/2))] \\ &- (2/5) \times [-(1/2)\log(1/2) - (1/2)\log_2(1/2)] \\ &- (1/5) \times [-(1/1)\log(1/1)] \\ = 0.5709\end{aligned}$$

$$\text{Gain}(S^{(1)}, \text{hum}) = \text{Entropy}(S^{(1)}) - \frac{|S^{(1)}_{\text{hum = high}}|}{|S^{(1)}|} \times \text{Entropy}(S^{(1)}_{\text{hum = high}}) - \frac{|S^{(1)}_{\text{hum = normal}}|}{|S^{(1)}|} \times \text{Entropy}(S^{(1)}_{\text{hum = normal}})$$

$$= [-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)] - (3/5) \times [-(3/3)\log(3/3))] - (2/5) \times [-(2/2)\log(2/2)]$$

$$= 0.9709$$

$$\text{Gain}(S^{(1)}, \text{wind}) = \text{Entropy}(S^{(1)}) - \frac{|S^{(1)}_{\text{wind = weak}}|}{|S^{(1)}|} \times \text{Entropy}(S^{(1)}_{\text{wind = weak}}) - \frac{|S^{(1)}_{\text{wind = strong}}|}{|S^{(1)}|} \times \text{Entropy}(S^{(1)}_{\text{wind = strong}})$$

$$= [-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)] - (3/5) \times [-(2/3)\log(2/3) - (1/3)\log_2(1/3)] - (2/5) \times [-(1/2)\log(1/2) - (1/2)\log(1/2)]$$

$$= 0.0110$$

The maximum of $\text{Gain}(S^{(1)}, \text{temp})$, $\text{Gain}(S^{(1)}, \text{hum})$ and $\text{Gain}(S^{(1)}, \text{wind})$ is $\text{Gain}(S^{(1)}, \text{hum})$. Hence we place "humidity" at Node 1 and split this node into two branches according to the values of the feature "humidity" to get the tree in Figure 8.9.

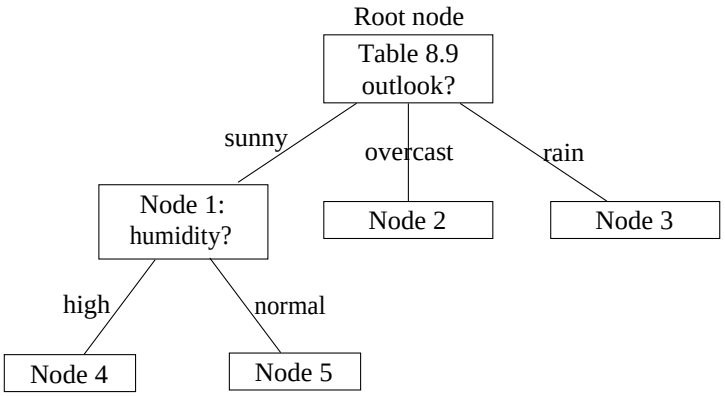


Figure 8.9: Decision tree for data in Table 8.9, after selecting the branching feature at Node 1

Step 6

It can be seen that all the examples in the the data set corresponding to Node 4 in Figure 8.9 have the same class label "no" and all the examples corresponding to Node 5 have the same class label "yes". So we represent Node 4 as a leaf node with value "no" and Node 5 as a leaf node with value "yes". Similarly, all the examples corresponding to Node 2 have the same class label "yes". So we convert Node 2 as a leaf node with value " yes. Finally, let $S^{(2)} = S_{\text{outlook = rain}}$. The highest information gain for this data set is $\text{Gain}(S^{(2)}, \text{humidity})$. The branches resulting from splitting this node corresponding to the values "high" and "normal" of "humidity" lead to leaf nodes with class labels "no" and "yes". With these changes, we get the tree in Figure 8.10.

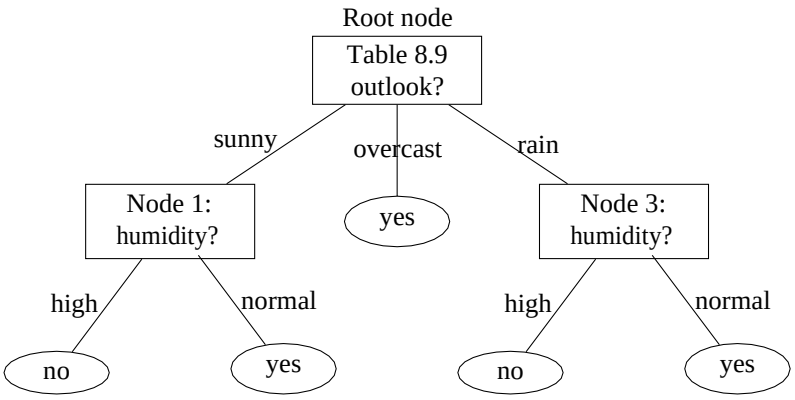


Figure 8.10: Decision tree for data in Table 8.9

Regression trees

A *regression problem* is the problem of determining a relation between one or more independent variables and an output variable which is a real continuous variable and then using the relation to predict the values of the dependent variables. Regression problems are in general related to prediction of numerical values of variables. Trees can also be used to make such predictions. A tree used for making predictions of numerical variables is called a *prediction tree* or a *regression tree*.

Example

Using the data in Table 8.11, construct a tree to predict the values of y.

x_1	1	3	4	6	10	15	2	7	16	0
x_2	12	23	21	10	27	23	35	12	27	17
y	10.1	15.3	11.5	13.9	17.8	23.1	12.7	43.0	17.6	14.9

Table 8.11: Data for regression tree

Solution

We shall construct a *raw decision tree* (a tree constructed without using any standard algorithm) to predict the value of y corresponding to any untabulated values of x_1 and x_2.

Step 1.We arbitrarily split the values of x_1 into two sets: One set defined by $x_1 < 6$ and the other set defined by $x \geq 6$. This splits the data into two parts. This yields the tree in Figure **??.1**

x_1	1	3	4	2	0
x_2	12	23	21	35	17
y	10.1	15.3	11.5	12.7	14.9

Table 8.12: Data for regression tree

Step 2.In Figure 8.12, consider the node specified by Table 8.12. We arbitrarily split the values of x_2 into two sets: one specified by $x_2 < 21$ and one specified by $x_2 \geq 21$. Similarly, the node specified by Table 8.13, we split the values of x_2 into sets: one specified by $x_2 < 23$

x_1	6	10	15	7	16
x_2	10	27	23	12	27
y	13.9	17.8	23.1	43.0	17.6

Table 8.13: Data for regression tree

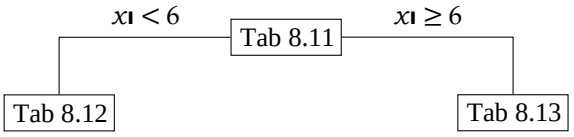


Figure 8.11: Part of a regression tree for Table 8.11

and one specified by $x_2 \geq 23$. The split data are given in Table 8.14(a) - (d). This gives us the tree in Figure 8.12.

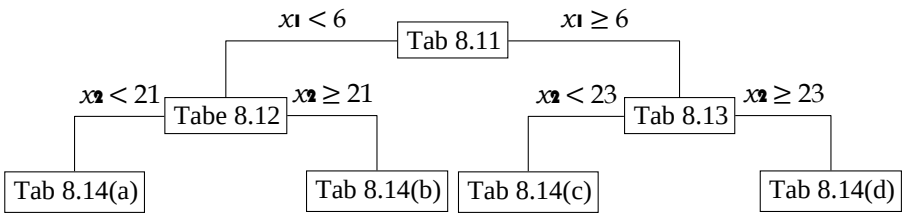


Figure 8.12: Part of regression tree for Table 8.11

Step 3.We next make the nodes specified by Table 8.14(a), . . . , Tab 8.14(d) into leaf nodes. In each of these leaf nodes, we write the average of the values in the corresponding table (this is a standard procedure). For, example, at Table 8.14(a), we write $\frac{1}{2}(10.1 + 14.9) = 12.5$. Then we get Figure 8.13.

x_1	1	0
x_2	12	17
y	10.1	14.9

(a)

x_1	3	4	2
x_2	23	21	35
y	15.3	11.5	12.7

(b)

x_1	6	7
x_2	10	12
y	13.9	43.0

(c)

x_1	10	15	16
x_2	27	23	27
y	17.8	23.1	17.6

(d)

Table 8.14: Data for regression tree

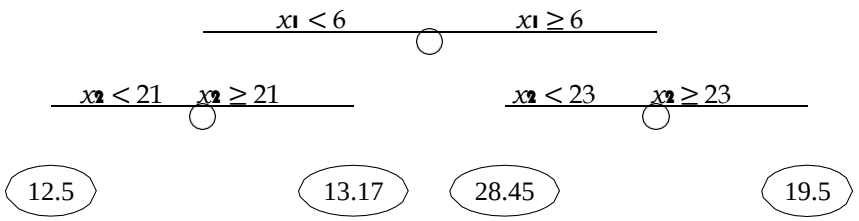


Figure 8.13: A regression tree for Table 8.11

Step 4.Figure 8.13 is the final raw regression tree for predicting the values of y based on the data in Table 8.11.

An algorithm for constructing regression trees

Starting with a learning sample, three elements are necessary to determine a regression tree:

1.A way to select a split at every intermediate node

2.A rule for determining when a node is terminal

3.A rule for assigning a value for the output variable to every terminal node

Notations

$x_1, x_2, \ldots, x_n$	:	The input variables
N	:	Number of samples in the data set
$y_1, y_2, \ldots, y_N$	:	The values of the output variables
T	:	A tree
c	:	A leaf of T
n_c	:	Number of data elements in the leaf c
C	:	The set of indices of data elements which are in the leaf c
m_c	:	The mean of the values of y which are in the leaf c
S_T	:	Sum of squares of errors in T

We have

$$m_c = \frac{1}{n_c} \sum_{i \in C} y_i$$

$$S_T = \sum_{c \in \text{leaves}(T)} \sum_{i \in C} (y_i - m_c)^2$$

Algorithm

Step 1. Start with a single node containing all data points. Calculate m_c and S_T .

Step 1. If all the points in the node have the same value for all the independent variables, stop.

Step 1. Otherwise, search over all binary splits of all variables for the one which will reduce S_T as much as possible.

(a) If the largest decrease in S_T would be less than some threshold δ, or one of the resulting nodes would contain less than q points, stop and if c is a node where we have stopped, then assign the value m_c to the node.

(b) Otherwise, take that split, creating two new nodes.

Step 1. In each new node, go back to Step 1.

Remarks

1. We have seen entropy and information defined for discrete variables. We can define them for continuous variables also. But in the case of regression trees, it is more common to use the sum of squares. The above algorithm is based on sum of squares of errors.

2. The CART algorithm mentioned below searches every distinct value of every predictor variable to find the predictor variable and split value which will reduce S_T as much as possible.

3. In the above algorithm, we have given the simplest criteria for stopping growing of trees. More sophisticated criteria which produce much less error have been developed.

Example

Consider the data given in Table 8.11.

1. Computation of S_T for the entire data set. Initially, there is only one node. So, we have:

$$m_c = \frac{1}{n} \sum_{\in} y_i$$

$$= \frac{1}{10}(10.1 + 15.3 + ... + 14.9)$$

$$= 17.99$$

$$S_T = \sum_{c \in \text{leaves } T} \sum_{i \in C} (y_i - m_c)^2$$

$$= (10.1 - 17.99)^2 + (15.3 - 17.99)^2 + ... + (14.9 - 17.99)^2$$

$$= 817.669$$

2. As suggested in the remarks above, we have to search every distinct value of x_1 and x_2 to find the predictor variable and split value which will reduce S_T as much as possible.

3. Let us consider the value 6 of x_1. This splits the data set into two parts c_1 and c_2. Let c_1 be the part defined by $x_1 < 6$ and c_2 the part defined by $x_1 \geq 6$. S_1 is given in Table 8.12 and S_2 by Table 8.13.Now

$$\text{leaves}(T) = \{c_1, c_2\}.$$

Let T_1 be the tree corresponding to this partition. Then

$$S_{T_1} = \sum_{c \in \text{leaves } T_1} \sum_{i \in C} (y_i - m_c)^2$$

$$= \sum_{i \in C_1} (y_i - m_{c_1})^2 + \sum_{i \in C_2} (y_i - m_{c_2})^2$$

$$m_{c_1} = \frac{1}{n_{c_1}} \sum_{i \in C_1} y_i$$

$$= \frac{1}{5}(10.1 + 15.3 + 11.5 + 12.7 + 14.9)$$

$$= 12.9$$

$$m_{c_2} = \frac{1}{n_{c_2}} \sum_{i \in C_2} y_i$$
$$= \frac{1}{5}(13.9 + 17.8 + 23.1 + 43.0 + 17.6)$$
$$= 23.08$$
$$S_{T_1} = [(10.1 - 12.9)^2 + ... + (14.9 - 12.9)^2] +$$
$$[(\quad ^2$$
$$= 558.588 \quad ^2 19 - 23.08) + ... +$$
$$(17.6 - 23.08)]$$

The reduction in sum of squares of errors is

$$S_T - S_{T_1} = 817.669 - 558.588 = 259.081.$$

4. In this way, we have compute the reduction in the sum of squares of errors corresponding to all other values of x_1 and each of the values of x_2 and choose the one for which the reduction is maximum.

5. The process has be continued. (Software package may be required to complete the problem.)

CART algorithm

We have seen how decision trees can be used to create a model that predicts the value of a target (or dependent variable) based on the values of several input or independent variables.

The CART, or *Classification And Regression Trees* methodology, was introduced in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone as an umbrella term to refer to the following types of decision trees:

- *Classification trees* where the target variable is categorical and the tree is used to identify the "class" within which a target variable would likely fall into.
- *Regression trees* where the target variable is continuous and tree is used to predict it's value.

The main elements of CART are:

- Rules for splitting data at a node based on the value of one variable
- Stopping rules for deciding when a branch is terminal and can be split no more
- A prediction for the target variable in each terminal node

Other decision tree algorithms

The C4.5 algorithm

The C4.5 algorithm is an algorithm developed by Ross Quinlan as an improvement of the ID3 algorithm. The following are some of the improvements incorporated in C4.5.

- Handling both continuous and discrete attributes
- Handling training data with missing attribute values
- Handling attributes with differing costs
- Pruning trees after creation

The C5.0 algorithm

The C5.0 algorithm represents a further improvement on the C4.5 algorithm. This was also developed by Ross Quinlan.

- Speed - C5.0 is significantly faster than C4.5.
- Memory usage - C5.0 is more memory efficient than C4.5.
- C5.0 gets similar results to C4.5 with considerably smaller decision trees.

The C5.0 algorithm is one of the most well-known implementations of the the decision tree algorithm. The source code for a single-threaded version of the algorithm is publicly available, and it has been incorporated into programs such as R. The C5.0 algorithm has become the industry standard to produce decision trees.

Issues in decision tree learning

In thie next feww sections, we discuss some of the practical issues in learning decision trees.

Avoiding overfitting of data

When we construct a decision tree, the various branches are grown (that is, sub-branches are constructed) just deeply enough to perfectly classify the training examples. This leads to difficulties when there is noise in the data or when the number of training examples are too small. In these cases the algorithm can produce trees that overfit the training examples.

Definition

We say that a hypothesis *overfits* the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances, including instances beyond the training set.

Impact of overfitting

Figure 8.14 illustrates the impact of overfitting in a typical decision tree learning. From the figure, we can see that the accuracy of the tree over training examples increases monotonically whereas the accuracy measured over independent test samples first increases then decreases.

Approaches to avoiding overfitting

The main approach to avoid overfitting is *pruning*. Pruning is a technique that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

- We may apply pruning earlier, that is, before it reaches the point where it perfectly classifies the training data.
- We may allow the tree to overfit the data, and then post-prune the true.

Now there is the problem of what criterion is to be used to determine the correct final tree size. One commonly used criterion is to use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.

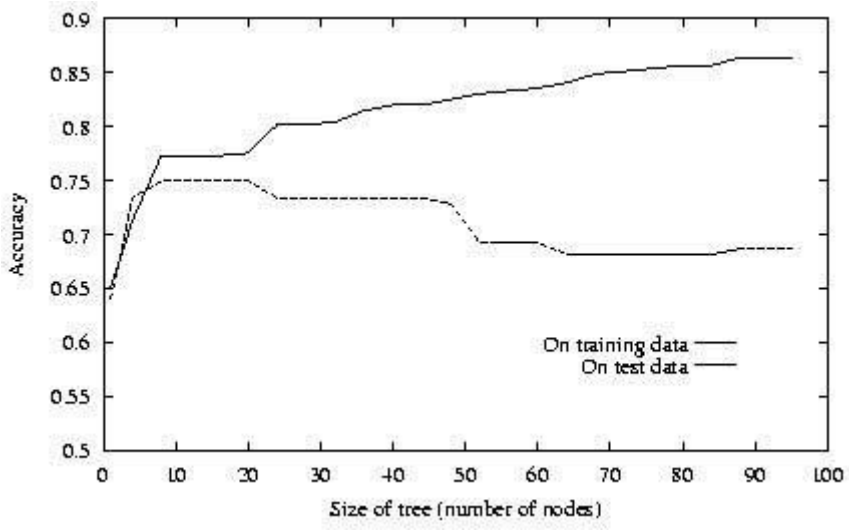


Figure 8.14: Impact of overfitting in decision tree learning

Case	Temperature	Headache	Nausea	Decision (Flue)
1	high	?	no	yes
2	very high	yes	no	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	?	yes	?	yes

Table 8.15: A dataset with missing attribute values

Reduced error pruning

In *reduced-error pruning*, we consider each of the decision tress to be a candidate for pruning. Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated to that node. Nodes are removed only if the resulting pruned tree performs no worse than the original over validation set. Nodes are pruned iteratively, always choosing the node whose removal most increases the accuracy over the validation set. Pruning of nodes is continued until further pruning decreases the accuracy over the validation set.

Problem of missing attributes

Table 8.15 shows a dataset with missing attribute values. the missing values are indicated by "?"s.

The following are some of the methods used to handle the problem of missing attributes.

- Deleting cases with missing attribute values
- Replacing a missing attribute value by the most common value of that attribute

- Assigning all possible values to the missing attribute value
- Replacing a missing attribute value by the mean for numerical attributes
- Assigning to a missing attribute value the corresponding value taken from the closest t cases, or replacing a missing attribute value by a new value

---

Sample questions

(a) Short answer questions

1. Explain the concept of a decision tree with an example.
2. What are the different types of decision trees?
3. Define the entropy of a dataset.
4. Write a formula to compute the entropy of a two-class dataset.
5. Define information gain and Gini index.
6. Give the names of five different decision-tree algorithms.
7. Can decision tree be used for regression? If yes, explain how. If no, explain why.
8. What is the difference between classification and regression trees?

(b) Long answer questions

1. Explain classification tree using an example.
2. Consider the following set of training examples:

Instance	Classification	a_1	a_2
1	+	T	T
2	+	T	T
3	−	T	F
4	+	F	F
5	−	F	T
6	−	F	T

   (a) What is the entropy of this collection of training examples with respect to the target function "classification"?

   (b) What is the information gain of a_2 relative to these training examples?

3. Explain the ID3 algorithm for learning decision trees.
4. Explain CART algorithm.
5. What are issues in decision tree learning? How are they overcome?
6. Describe an algorithm to construct regression trees.
7. What do you mean by information gain and entropy? How is it used to build the decision trees? Illustrate using an example.
8. Use ID3 algorithm to construct a decision tree for the data in the following table.

Instance no.	Class label	x_1	x_2
1	1	T	T
2	1	T	T
3	0	T	F
4	1	F	F
5	0	F	T
6	0	F	T

9. Use ID3 algorithm to construct a decision tree for the data in the following table.

Gender	Car ownership	Travel cost	Income level	Class (mode of transportation)
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

10. Use ID3 algorithm to construct a decision tree for the data in the following table.

Age	Competition	Type	Class (profit)
Old	Yes	Software	Down
Old	No	Software	Down
Old	No	Hardware	Down
Mid	Yes	Software	Down
Mid	Yes	Hardware	Down
Mid	No	Hardware	Up
Mid	No	Software	Up
New	Yes	Software	Up
New	No	Hardware	Up
New	No	Software	Up

11. Construct a decision tree for the following data.

Class label (risk)	Collateral	Income	Debt	Credit history
high	none	low	high	bad
high	none	middle	high	unknown
moderate	none	middle	low	unknown
high	none	low	low	unknown
low	none	upper	low	unknown
low	adequate	upper	low	unknown
high	none	low	low	bad
moderate	adequate	upper	low	bad
low	none	upper	low	good
low	adequate	upper	high	good
high	none	low	high	good
moderate	none	middle	high	good
low	none	upper	high	good
high	none	middle	high	bad

Neural networks

Introduction

An *Artificial Neural Network* (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs. Just as a brain uses a network of interconnected cells called neurons to create a massive parallel processor, ANN uses a network of artificial neurons or nodes to solve learning problems.

Biological motivation

Let us examine how a biological neuron functions. Figure 9.2 gives a schematic representation of the functioning of a biological neuron.

In the cell, the incoming signals are received by the cell's *dendrites* through a biochemical process. The process allows the impulse to be weighted according to its relative importance or frequency. As the cell body begins accumulating the incoming signals, a threshold is reached at which the cell fires and the output signal is transmitted via an electrochemical process down the *axon*. At the axon's terminals, the electric signal is again processed as a chemical signal to be passed to the neighboring neurons across a tiny gap known as a *synapse*.[1]

Biological learning systems are built of very complex webs of interconnected neurons. The human brain has an interconnected network of approximately 10^{11} neurons, each connected, on an average, to 10^{4} other neurons. Even though the neuron switching speeds are much slower than than

[1]Neuron. (2018, February 15). In Wikipedia, The Free Encyclopedia. Retrieved 01:44, February 23, 2018.

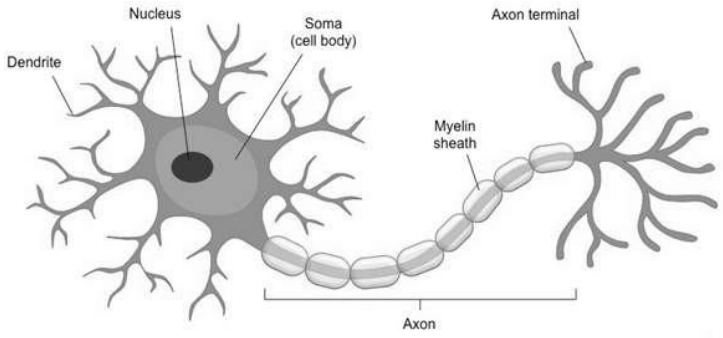


Figure 9.1: Anatomy of a neuron

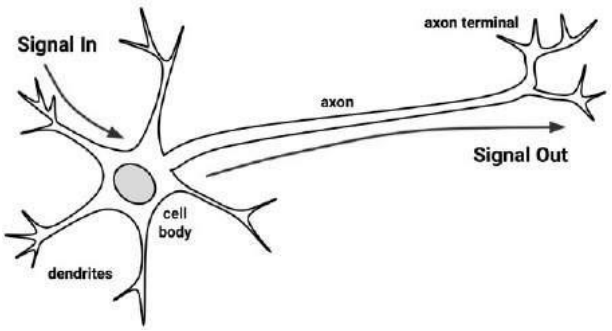


Figure 9.2: Flow of signals in a biological neuron

computer switching speeds, we are able to take complex decisions relatively quickly. Because of this, it is believed that the information processing capabilities of biological neural systems is a consequence of the ability of such systems to carry out a huge number of parallel processes distributed over many neurons. The developments in ANN systems are motivated by the desire to implement this kind of highly parallel computation using distributed representations.

Artificial neurons

Definition

An *artificial neuron* is a mathematical function conceived as a model of biological neurons. Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs (representing excitatory postsynaptic potentials and inhibitory postsynaptic potentials at neural dendrites) and sums them to produce an output. Each input is separately weighted, and the sum is passed through a function known as an *activation function* or *transfer function*.

Schematic representation of an artificial neuron

The diagram shown in Figure **??** gives a schematic representation of a model of an artificial neuron. The notations in the diagram have the following meanings:

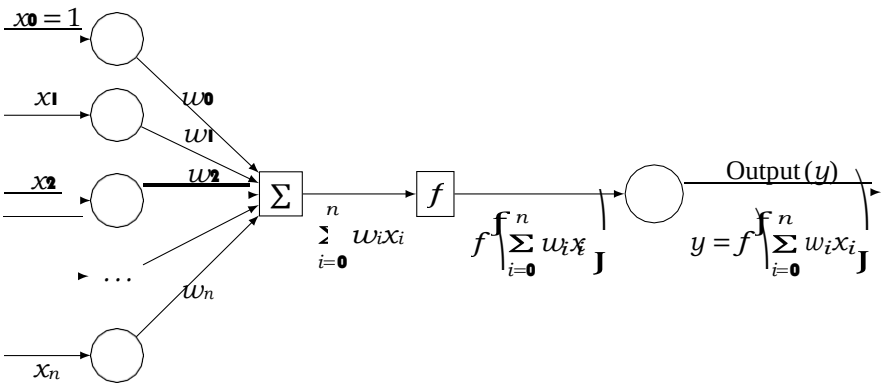


Figure 9.3: Schematic representation of an artificial neuron

$x_1, x_2, \ldots x_n$: input signals

$w_1, w_2, \ldots w_n$: weights associated with input signals

x_0 : input signal taking the constant value 1
w_0 weight associated with x_0 (called bias)
Σ : indicates summation of input signals
f : function which produces the output
y : output signal

The function f can be expressed in the following form:

$$y = f\left(\sum_{i=0}^{n} w_i x_i\right) \tag{9.1}$$

Remarks

The small circles in the schematic representation of the artificial neuron shown in Figure 9.3 are called the *nodes* of the neuron. The circles on the left side which receives the values of $x_0, x_1, \ldots, x_n$ are called the *input nodes* and the circle on the right side which outputs the value of y is called *output node*. The squares represent the processes that are taking place before the result is outputted. They need not be explicitly shown in the schematic representation. Figure 9.4 shows a simplified representation of an artificial neuron.

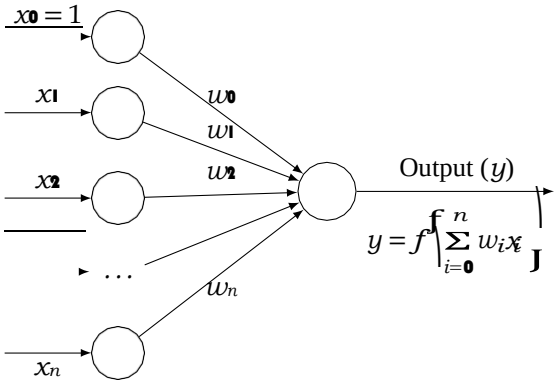


Figure 9.4: Simplified representation of an artificial neuron

Activation function

Definition

In an artificial neural network, the function which takes the incoming signals as input and produces the output signal is known as the *activation function*.

Remark

Eq.(9.1) represents the activation function of the ANN model shown in Figure **??**.

Some simple activation functions

The following are some of the simple activation functions.

1. Threshold activation function

The *threshold activation function* is defined by

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases}$$

The graph of this function is shown in Figure 9.5.

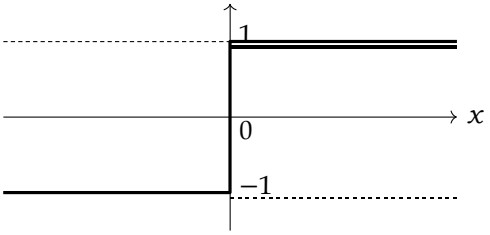


Figure 9.5: Threshold activation function

2. Unit step functions

Sometimes, the threshold activation function is also defined as a unit step function in which case it is called a *unit-step activation function*. This is defined as follows:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

The graph of this function is shown in Figure 9.6.

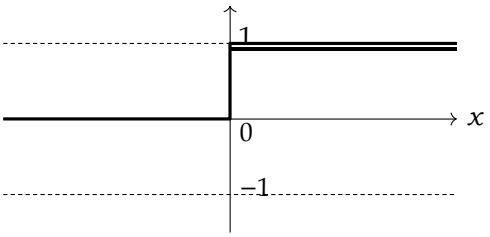


Figure 9.6: Unit step activation function

3. Sigmoid activation function (logistic function)

One of the must commonly used activation functions is the sigmoid activation function. It is defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}}$$

The graph of the function is shown in Figure 9.7.

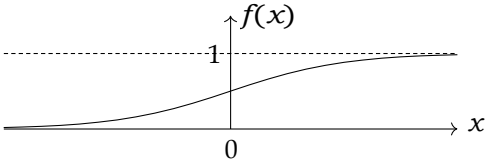


Figure 9.7: The sigmoid activation function

4. Linear activation function

The linear activation function is defined by

$$F(x) = mx + c.$$

This defines a straight line in the xy-plane.

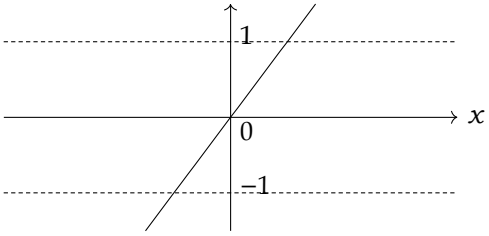


Figure 9.8: Linear activation function

5. Piecewise (or, saturated) linear activation function
This is defined by

$$f(x) = \begin{cases} 0 & \text{if } x < x_{\min} \\ mx + c & \text{if } x_{\min} \le x \le x_{\max} \\ 0 & \text{if } x > x_{\max} \end{cases}$$

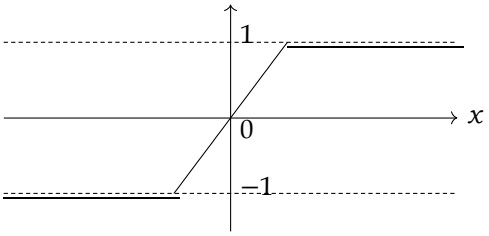


Figure 9.9: Piecewise linear activation function

6. Gaussian activation function

This is defined by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

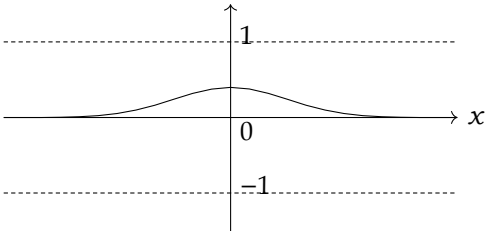


Figure 9.10: Gaussian activation function

7. Hyperbolic tangential activation function

This is defined by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

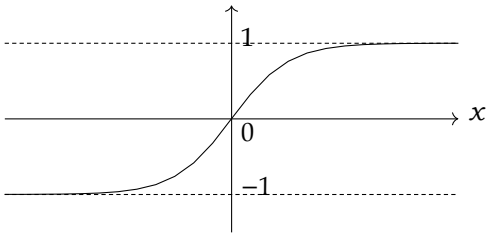


Figure 9.11: Hyperbolic tangent activation function

Perceptron

The perceptron is a special type of artificial neuron in which thee activation function has a special form.

Definition

A perceptron is an artificial neuron in which the activation function is the threshold function.

Consider an artificial neuron having $x_1, x_2, \ldots, x_n$ as the input signals and $w_1, w_2, \ldots, w_n$ as the associated weights. Let w_0 be some constant. The neuron is called a perceptron if the output of the neuron is given by the following function:

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots + w_n x_n > 0 \\ -1 & \text{if } w_0 + w_1 x_1 + \ldots + w_n x_n \leq 0 \end{cases}$$

Figure 9.12 shows the schematic representation of a perceptron.

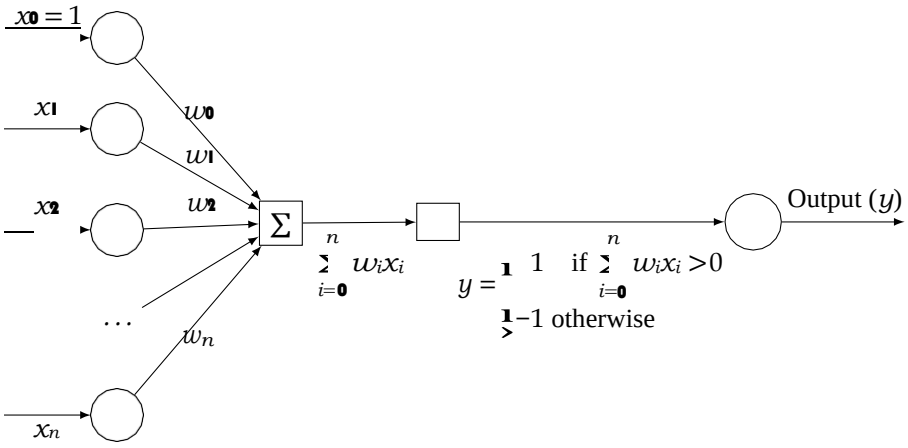


Figure 9.12: Schematic representation of a perceptrn

Remarks

1. The quantity $-w_0$ can be looked upon as a "threshold" that should be crossed by the weighted sum $w_1 x_1 + ... + w_n x_n$ in order for the neuron to output a "1".

Representations of boolean functions by perceptrons

In this section we examine whether simple boolean functions like x_1 AND x_2 can be represented by perceptrons. To be consistent with the conventions in the definition of a perceptron we assume that the values -1 and 1 represent the boolean constants "false" and "true" respectively.

Representation of x_1 AND x_2

Let x_1 and x_2 be two boolean variables. Then the boolean function x_1 AND x_2 is represented by Table 9.1. It can be easily verified that the perceptron shown in Figure 9.13 represents the function

x_1	x_2	x_1 AND x_2
−1	−1	−1
−1	1	−1
1	−1	−1
1	1	1

Table 9.1: The boolean function x_1 AND x_2

x_1 AND x_2.

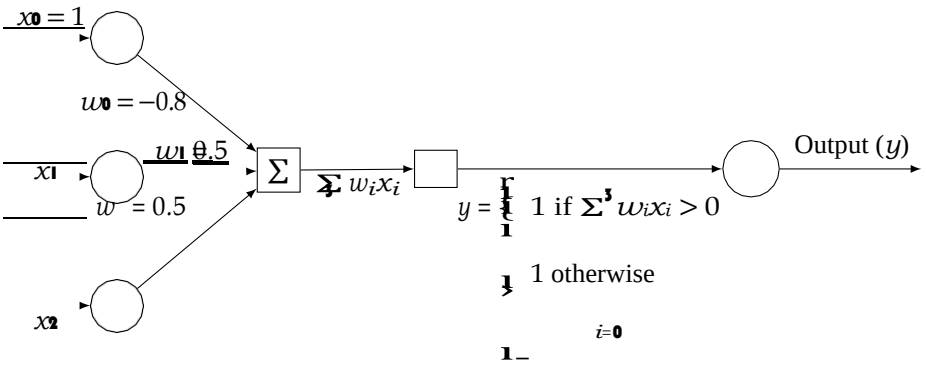


Figure 9.13: Representation of x_1 AND x_2 by a perceptron

In the perceptron shown in Figure 9.13, the output is given by

$$y = \begin{cases} 1 \text{ if } \sum_{i=0}^{2} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$$
$$= \begin{cases} 1 \text{ if } -0.8 + 0.5x_1 + 0.5x_2 > 0 \\ -1 \text{ otherwise} \end{cases}$$

Representations of OR, NAND and NOR

The functions x_1 OR x_2, x_1 NAND x_2 and x_1 NOR x_2 can also be represented by perceptrons. Table 9.2 shows the values to be assigned to the weights w_0, w_1, w_2 for getting these boolean functions.

Boolean function	w_0	w_1	w_2
x_1 AND x_2	−0.8	0.5	0.5
x_1 OR x_2	−0.3	0.5	0.5
x_1 NAND x_2	0.8	−0.5	−0.5
x_1 NOR x_2	0.3	−0.5	−0.5

Table 9.2: Representations of boolean functions by perceptrons

Remarks

Not all boolean functions can be represented by perceptrons. For example, the boolean function x_1 XOR x_2 cannot be represented by a perceptron. This means that we cannot assign values to w_0, w_1, w_2 such that the expression $w_0 + w_1 x_1 + w_2 x_2$ takes the values of x_1 XOR x_2, and that this is the case can be easily verified also.

Learning a perceptron

By "learning a perceptron" we mean the process of assigning values to the weights and the threshold such that the perceptron produces correct output for each of the given training examples. The following are two algorithms to solve this learning problem:

Perceptron learning algorithm

Definitions

In the algorithm, we use the following notations:

n	:	Number of input variables
$y = f(\mathbf{z})$	:	Output from the perceptron for an input vector $\mathbf{z}$
$D = \{(\mathbf{x}_1, d_1), \ldots, (\mathbf{x}_s, d_s)\}$	:	Training set of s samples
$\mathbf{x}_j = (x_{j0}, x_{j1}, \ldots, x_{jn})$	:	The n-dimensional input vector
d_j	:	Desired output value of the perceptron for the input $\mathbf{x}_j$
x_{ji}	:	Value of the i-th feature of the j-th training input vector
x_{j0}	:	1
w_i	:	Weight of the i-th input variable
$w_i(t)$	:	Weight i at the t-th iteration

Algorithm

Step 1. Initialize the weights and the threshold. Weights may be initialized to 0 or to a small random value.

Step 2. For each example j in the training set D, perform the following steps over the input $\mathbf{x}_j$ and desired output d_j:

a) Calculate the actual output:

$$y_j(t) = f[w_0(t)x_{j0} + w_1(t)x_{j1} + w_2(t)x_{j2} + \ldots + w_n(t)x_{jn}]$$

b)Update the weights:

$$w_i(t+1) = w_i(t) + (d_j - y_j(t))x_{ji}$$

for all features $0 \leq i \leq n$.

Step 3.Step 2 is repeated until the iteration error $\frac{1}{s}\sum_{j=1}^{s}|d_j - y_j(t)|$ is less than a user-specified error threshold γ, or a predetermined number of iterations have been completed, where s is again the size of the sample set.

Remarks

The above algorithm can be applied only if the training examples are *linearly separable*.

Artificial neural networks

An *artificial neural network* (ANN) is a computing system inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units called artificial neurons. Each connection between artificial neurons can transmit a signal from one to another. The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.

each connection between artificial neurons has a weight attached to it that get adjusted as learning proceeds. Artificial neurons may have a threshold such that only if the aggregate signal crosses that threshold the signal is sent. Artificial neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the input layer to the output layer, possibly after traversing the layers multiple times.

Characteristics of an ANN

An ANN can be defined and implemented in several different ways. The way the following characteristics are defined determines a particular variant of an ANN.

- **The activation function**

  This function defines how a neuron's combined input signals are transformed into a single output signal to be broadcasted further in the network.

- **The network topology (or architecture)**

  This describes the number of neurons in the model as well as the number of layers and manner in which they are connected.

- **The training algorithm**

  This algorithm specifies how connection weights are set in order to inhibit or excite neurons in proportion to the input signal.

Activation functions

The activation function is the mechanism by which the artificial neuron processes incoming information and passes it throughout the network. Just as the artificial neuron is modeled after the biological version, so is the activation function modeled after nature's design.

Let $x_1, x_2, \ldots, x_n$ be the input signals, $w_1, w_2, \ldots, w_n$ be the associated weights and $-w_0$ the threshold. Let

$$x = w_0 + w_1x_1 + \ldots + w_nx_n.$$

The activation function is some function of x. Some of the simplest and commonly used activations are given in Section 9.4.

Network topology

By "network topology" we mean the patterns and structures in the collection of interconnected nodes. The topology determines the complexity of tasks that can be learned by the network. Generally, larger and more complex networks are capable of identifying more subtle patterns and complex decision boundaries. However, the power of a network is not only a function of the network size, but also the way units are arranged.

Different forms of forms of network architecture can be differentiated by the following characteristics:

- The number of layers
- Whether information in the network is allowed to travel backward
- The number of nodes within each layer of the network

1. The number of layers

In an ANN, the *input nodes* are those nodes which receive unprocessed signals directly from the input data. The *output nodes* (there may be more than one) are those nodes which generate the final predicted values. A *hidden node* is a node that processes the signals from the input nodes (or other such nodes) prior to reaching the output nodes.

The nodes are arranged in *layers*. The set of nodes which receive the unprocessed signals from the input data constitute the *first layer* of nodes. The set of hidden nodes which receive the outputs from the nodes in the first layer of nodes constitute the *second layer* of nodes. In a similar way we can define the third, fourth, etc. layers. Figure 9.14 shows an ANN with only one layer of nodes. Figure 9.15 shows an ANN with two layers.

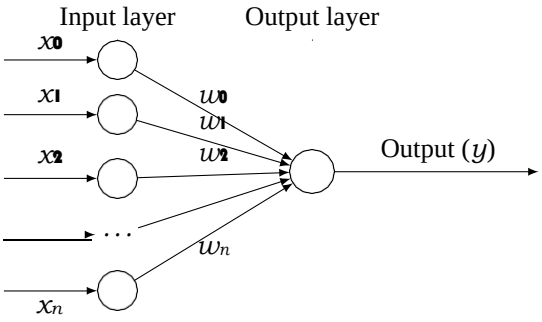


Figure 9.14: An ANN with only one layer

2. The direction of information travel

Networks in which the input signal is fed continuously in one direction from connection to connection until it reaches the output layer are called *feedforward networks*. The network shown in Figure 9.15 is a feedforward network.

Networks which allows signals to travel in both directions using loops are called *recurrent networks* (or, *feedback networks*).

In spite of their potential, recurrent networks are still largely theoretical and are rarely used in practice. On the other hand, feedforward networks have been extensively applied to real-world problems. In fact, the multilayer feedforward network, sometimes called the Multilayer Perceptron (MLP), is the de facto standard ANN topology. If someone mentions that they are fitting a neural network, they are most likely referring to a MLP.

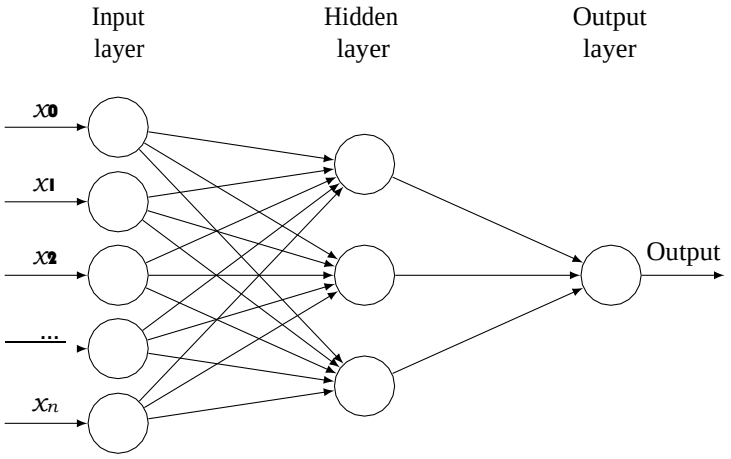


Figure 9.15: An ANN with two layers

3. The number of nodes in each layer

The number of input nodes is predetermined by the number of features in the input data. Similarly, the number of output nodes is predetermined by the number of outcomes to be modeled or the number of class levels in the outcome. However, the number of hidden nodes is left to the user to decide prior to training the model. Unfortunately, there is no reliable rule to determine the number of neurons in the hidden layer. The appropriate number depends on the number of input nodes, the amount of training data, the amount of noisy data, and the complexity of the learning task, among many other factors.

The training algorithm

There are two commonly used algorithms for learning a single perceptron, namely, the perceptron rule and the delta rule. The former is used when the training data set is linearly separable and the latter when the training data set is not linearly separable.

The algorithm which is now commonly used to train an ANN is known simply as *backpropagation*.

The cost function

Definition

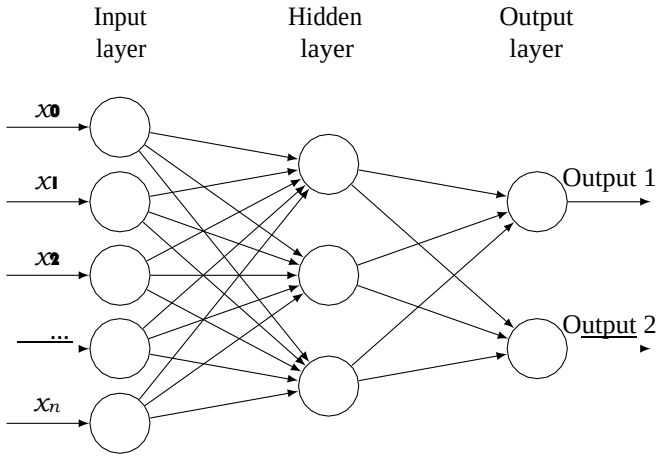
In a machine learning algorithm, the *cost function* is a function that measures how well the algorithm maps the target function that it is trying to guess or a function that determines how well the algorithm performs in an optimization problem.

Remaarks

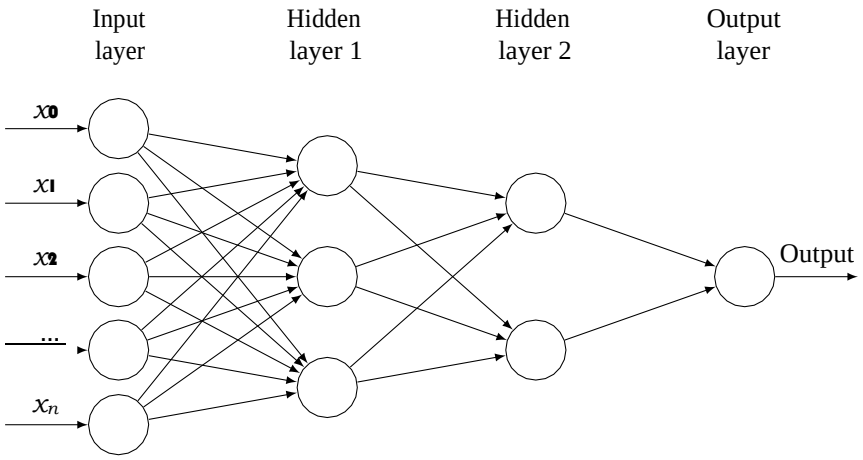
The cost function is also called the *loss function*, the *objective function*, the *scoring function*, or the *error function*.

Example

Let y be the the output variable. Let $y_1, \ldots, y_n$ be the actual values of y in n examples and $\hat{y}_1, \ldots, \hat{y}_n$ be the values predicted by an algorithm.



(a) Network with one hidden layer and two output nodes



(b) Network with two hidden layers

Figure 9.16: Examples of different topologies of networks

1. The sum of squares of the differences between the predicted and actual values of y, denoted by SSE and defined below, can be taken as a cost function for the algorithm.

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

2. The mean of the sum of squares of the differences between the predicted and actual values of y, denoted by MSE and defined below, can be taken as a cost function for the algorithm.

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

Backpropagation

The backpropagation algorithm was discovered in 1985-86. Here is an outline of the algorithm.

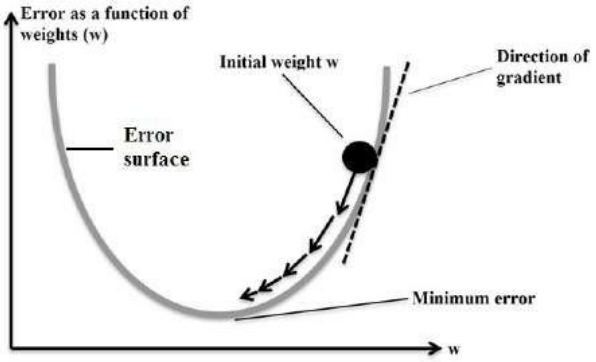


Figure 9.17: A simplified model of the error surface showing the direction of gradient

Outline of the algorithm

1. Initially the weights are assigned at random.
2. Then the algorithm iterates through many cycles of two processes until a stopping criterion is reached. Each cycle is known as an *epoch*. Each epoch includes:
   (a) A *forward phase* in which the neurons are activated in sequence from the input layer to the output layer, applying each neuron's weights and activation function along the way. Upon reaching the final layer, an output signal is produced.
   (b) A *backward phase* in which the network's output signal resulting from the forward phase is compared to the true target value in the training data. The difference between the network's output signal and the true value results in an error that is propagated backwards in the network to modify the connection weights between neurons and reduce future errors.
3. The technique used to determine how much a weight should be changed is known as *gradient descent method*. At every stage of the computation, the error is a function of the weights. If we plot the error against the wights, we get a higher dimensional analog of something like a curve or surface. At any point on this surface, the gradient suggests how steeply the error will be reduced or increased for a change in the weight. The algorithm will attempt to change the weights that result in the greatest reduction in error (see Figure 9.17).

Illustrative example

To illustrate the various steps in the backpropagation algorithm, we consider a small network with two inputs, two outputs and one hidden layer as shown in Figure 9.18.[2]

We assume that there are two observations:

Sample	Input 1 i_1	Input 2 i_2	Output target 1 T_1	Output target 2 T_2
1	0.05	0.10	0.01	0.99
2	0.25	0.18	0.23	0.79

We are required to estimate the optimal values of the weights $w_1, \ldots, w_8, b_1, b_2$. Here b_1 and b_2 are the biases. For simplicity, we have assigned the same biases to both nodes in the same layer.

Step 1. We initialise the connection weights to small random values. These initial weights are shown in Figure 9.19.

[2] Thanks to `https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/` for this example.

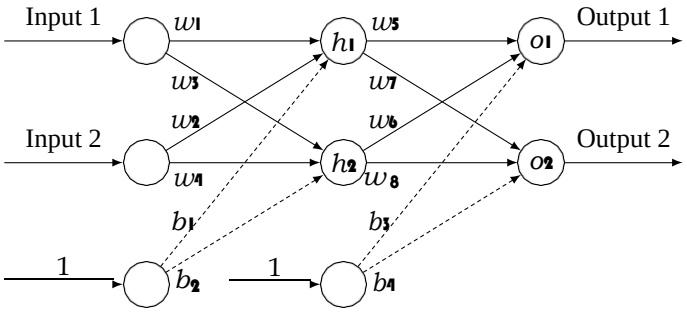


Figure 9.18: ANN for illustrating backpropagation algorithm

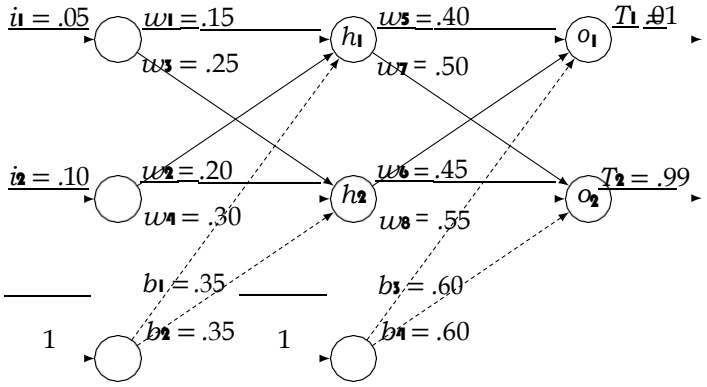


Figure 9.19: ANN for illustrating backpropagation algorithm with initial values for weights

Step 2.Present the first sample inputs and the corresponding output targets to the network. This is shown in Figure 9.19.

Step 3.Pass the input values to the first layer (the layer with nodes h_1 and h_2).

Step 4.We calculate the outputs from h_1 and h_2. We use the logistic activation function

$$f(x) = \frac{1}{1 + e^{-x}}.$$

$$\begin{aligned}
\text{out}_{h_1} &= f(w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1) \\
&= f(0.15 \times 0.05 + 0.20 \times 0.10 + 0.35 \times 1) \\
&= f(0.3775) \\
&= \frac{1}{1 + e^{-0.3775}} \\
&= 0.59327 \\
\text{out}_{h_2} &= f(w_3 \times i_1 + w_4 \times i_2 + b_2 \times 1) \\
&= f(0.25 \times 0.05 + 0.30 \times 0.10 + 0.35 \times 1) \\
&= f(0.3925) \\
&= \frac{1}{1 + e^{-0.3925}} \\
&= 0.59689
\end{aligned}$$

Step 5. We repeat this process for every layer. We get the outputs from the nodes in the output layer as follows:

$$\begin{aligned}
\text{out}_{o_1} &= f(w_5 \times \text{out}_{h_1} + w_6 \times \text{out}_{h_2} + b_3 \times 1) \\
&= f(0.40 \times 0.59327 + 0.45 \times 0.59689 + 0.60 \times 1) \\
&= f(1.10591) \\
&= \frac{1}{1 + e^{-1.10591}} \\
&= 0.75137 \\
\text{out}_{o_2} &= f(w_7 \times \text{out}_{h_1} + w_8 \times \text{out}_{h_2} + b_4 \times 1) \\
&= f(0.50 \times 0.59327 + 0.55 \times 0.59689 + 0.60 \times 1) \\
&= f(1.22492) \\
&= \frac{1}{1 + e^{-1.22492}} \\
&= 0.77293
\end{aligned}$$

The sum of the squares of the output errors is given by

$$\begin{aligned}
E &= \frac{1}{2}(T_1 - \text{out}_{o_1})^2 + \frac{1}{2}(T_2 - \text{out}_{o_2})^2 \\
&= (0.01 - 0.75137)^2 + (0.99 - 0.77293)^2 \\
&= 0.298371
\end{aligned}$$

Step 6. We begin backward phase. We adjust the weights. We first adjust the weights leading to the nodes o_1 and o_2 in the output layer and then the weights leading to the nodes h_1 and h_2 in the hidden layer. The adjusted values of the weights $w_1, \ldots, w_8, b_1, \ldots, b_4$ are denoted by $w_1^+, \ldots, w_8^+, b_1^+, \ldots, b_4^+$. The computations use a certain constant η called the *learning rate*. In the following we have taken $\eta = 0.5$.

(a) Computation of adjusted weights leading to o_1 and o_2:

$$\begin{aligned}
\delta_{o_1} &= (T_1 - \text{out}_{o_1}) \times \text{out}_{o_1} \times (1 - \text{out}_{o_1}) \\
&= (0.01 - 0.75137) \times 0.75137 \times (1 - 0.75137) \\
&= -0.13850 \\
w_5^+ &= w_5 + \eta \times \delta_{o_1} \times \text{out}_{h_1} \\
&= 0.40 + 0.5 \times (-0.13850) \times 0.59327 \\
&= 0.35892 \\
w_6^+ &= w_6 + \eta \times \delta_{o_1} \times \text{out}_{h_2} \\
&= 0.45 + 0.5 \times (-0.13850) \times 0.59689 \\
&= 0.40867 \\
b_3^+ &= b_3 + \eta \times \delta_{o_1} \times 1 \\
&= 0.60 + 0.5 \times (-0.13850) \times 1 \\
&= 0.53075
\end{aligned}$$

$$\begin{aligned}
\delta_{o_2} &= (T_2 - \text{out}_{o_2}) \times \text{out}_{o_2} \times (1 - \text{out}_{o_2}) \\
&= (0.99 - 0.77293) \times 0.77293 \times (1 - 0.77293) \\
&= 0.03810 \\
w_7^+ &= w_7 + \eta \times \delta_{o_2} \times \text{out}_{h_1} \\
&= 0.50 + 0.5 \times 0.03810 \times 0.59327
\end{aligned}$$

$$= 0.51130$$

$$w_8^+ = w_8 + \eta \times \delta_{o_2} \times \text{out}_{h_2}$$
$$= 0.55 + 0.5 \times 0.03810 \times 0.59689$$
$$= 0.56137$$

$$b_4^+ = b_4 + \eta \times \delta_{o_2} \times 1$$
$$= 0.60 + 0.5 \times 0.03810 \times 1$$
$$= 0.61905$$

(b) Computation of adjusted weights leading to h_1 and h_2:

$$\delta = (\delta \times w \text{ of } \delta \times \text{ of } w) \times \text{out} \times (1 - \text{out})_h$$
$$= (-0.13850 \times 0.40 + 0.03810 \times 0.50) \times 0.59327 \times (1 - 0.59327)$$
$$= -0.00877$$

$$w_1^+ = w_1 + \eta_{h_1} \times \delta_1 \times i$$
$$= 0.15 + 0.5 \times (-0.00877) \times 0.05$$
$$= 0.14978$$

$$w_2^+ = w_2 + \eta \times \delta_{h_1} \times i_2$$
$$= 0.20 + 0.5 \times (-0.00877) \times 0.10$$
$$= 0.19956$$

$$b_1^+ = b_1 + \eta_{h_1} \times \delta \times 1$$
$$= 0.35 + 0.5 \times (-0.00877) \times 1$$
$$= 0.34562$$

$$\delta = (\delta \times w \text{ of } \delta \times \text{ of } w) \times \text{out} \times (1 - \text{out})_h$$
$$= ((-0.13850) \times 0.45 + 0.03810 \times 0.55) \times 0.59689 \times (1 - 0.59689)$$
$$= -0.00995$$

$$w_3^+ = w_3 + \eta_{h_2} \times \delta_1 \times i$$
$$= 0.25 + 0.5 \times (-0.00995) \times 0.05$$
$$= 0.24975$$

$$w_4^+ = w_4 + \eta \times \delta_{h_2} \times i_2$$
$$= 0.30 + 0.5 \times (-0.00995) \times 0.10$$
$$= 0.29950$$

$$b_2^+ = b_2 + \eta_{h_2} \times \delta \times 1$$
$$= 0.35 + 0.5 \times (-0.00995) \times 1$$
$$= 0.34503$$

Step 7.Now we set:

$$w_1 = w_1^+, \quad w_2 = w_2^+, \quad w_3 = w_3^+, \quad w_4 = w_4^+$$
$$w_5 = w_5^+, \quad w_6 = w_6^+, \quad w_7 = w_7^+, \quad w_8 = w_8^+$$
$$b_1 = b_1^+, \quad b_2 = b_2^+, \quad b_3 = b_3^+, \quad b_4 = b_4^+$$

We choose the next sample input and the corresponding output targets to the network and repeat Steps 2 to 6.

Step 8.The process in Step 7 is repeated until the root mean square of output errors is minimised.

Remarks

1. The constant $\frac{1}{2}$ is included in the expression for E so that the exponent is cancelled when we differentiate it. The result has been multiplied by a learning rate $\eta = 0.5$ and so it doesnâĂŹt matter that we introduce the constant $\frac{1}{2}$ in E.

2. In the above computations, the method used to calculate the adjusted weights is known as the *delta rule*.

3. The rule for computing the adjusted weights can be succinctly stated as follows. Let w be a weight and w^+ its adjusted weight. Let E be the the total sum of squares of errors. Then w^+ is computed by

$$w^+ = w - \eta \frac{\partial E}{\partial w}.$$

Here $\frac{\partial E}{\partial w}$ is the gradient of E with respect to w; that is, the rate at which E is changing with respect to w. (The set of all such gradients specifies the direction in which E is decreasing the most rapidly, that is, the direction of quickest descent.) For example, it can be shown that

$$\begin{aligned}\frac{\partial E}{\partial w_5} &= -(T_1 - \text{out}_{o_1}) \times \text{out}_{o_1} \times (1 - \text{out}_{o_1}) \times \text{out}_{h_1} \\ &= -\delta_{o_1} \times \text{out}_{h_1}\end{aligned}$$

and so

$$\begin{aligned}w_5^+ &= w_5 - \eta \frac{\partial E}{\partial w_5} \\ &= w_5 + \eta \times \delta_{o_1} \times \text{out}_{h_1}\end{aligned}$$

The algorithm

The backpropagation algorithm trains a given feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted. The adjustments are based on the mean square error of the output response to the sample input and it is known as the *delta learning rule*. The set of these sample patterns are repeatedly presented to the network until the error value is minimized.

Notations

Figures 9.20 and 9.21 show the various notations used in the algorithm.

M	: Number of layers (excluding the input layer which is assigned the layer number 0)
N_j	: Number of neurons (nodes) in j-th layer
$\mathbf{X}_p = (X_{p1}, X_{p2}, \ldots, X_{pN_0})$	: p-th training sample
$\mathbf{T}_p = (T_{p1}, T_{p2}, \ldots, T_{pN_M})$	: Known output corresponding to the p-th training sample
$\mathbf{O}_p = (O_{p1}, O_{p2}, \ldots, O_{pN_M})$	: Actual output by the network corresponding to the p-th training sample
Y_{ji}	: Output from the i-th neuron in layer j
W_{jik}	: Connection weight from k-th neuron in layer $(j-1)$ to i-th neuron in layer j
δ_{ji}	: Error value associated with the i-th neuron in layer j

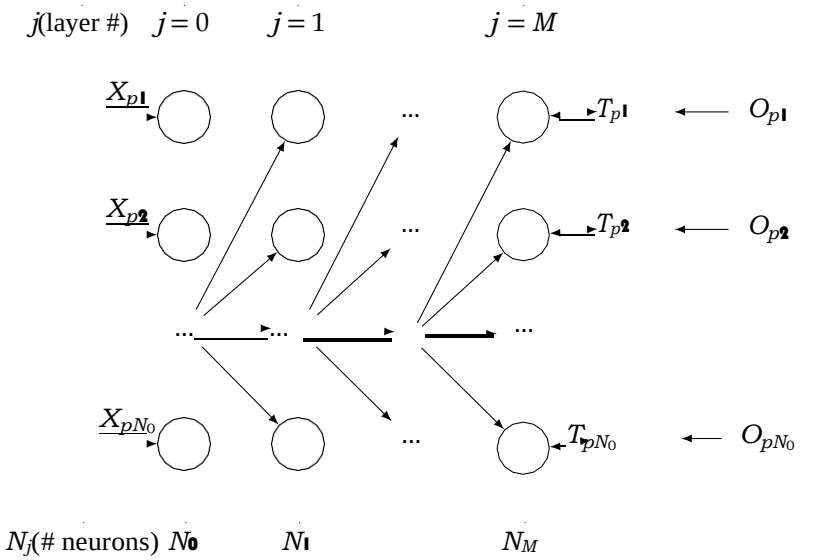


Figure 9.20: Notations of backpropagation algorithm

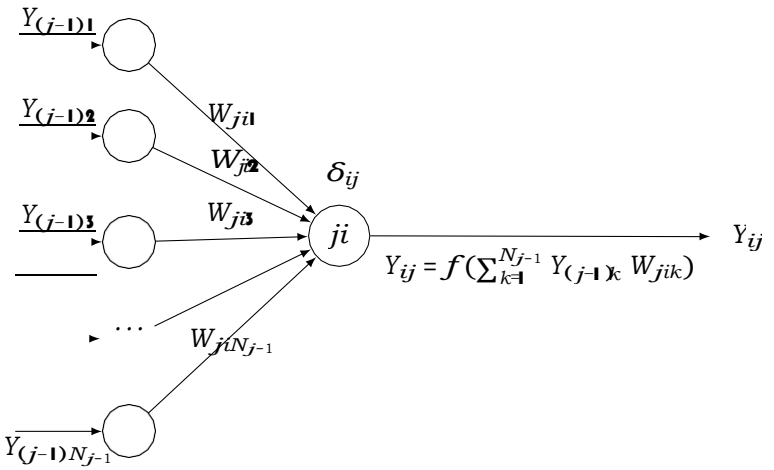


Figure 9.21: Notations of backpropagation algorithm: The i-th node in layer j

The algorithm

Step 1.Initialize connection weights into small random values.

Step 2.Present the pth sample input vector of pattern

$$\mathbf{X}_p = (X_{p1}, X_{p2}, \ldots, X_{pN_0})$$

and the corresponding output target

$$\mathbf{T}_p = (T_{p1}, T_{p2}, \ldots, T_{pN_M})$$

to the network.

Step 3.Pass the input values to the first layer, layer 1. For every input node i in layer 0, perform: $Y_{0i} = X_{pi}$.

Step 4.For every neuron i in every layer $j = 1, 2, \ldots, M$, find the output from the neuron:

$$Y_{ji} = f(\sum_{k=1}^{N_{j-1}} Y_{(j-1)k} W_{jik}),$$

where

$$f(x) = \frac{1}{1 + \exp(-x)}.$$

Step 5.Obtain output values. For every output node i in layer M, perform:

$$O_{pi} = Y_{Mi}.$$

Step 6.Calculate error value δ_{ji} for every neuron i in every layer in backward order $j = M, M - 1, \ldots, 2, 1$, from output to input layer, followed by weight adjustments. For the output layer, the error value is:

$$\delta_{Mi} = Y_{Mi}(1 - Y_{Mi})(T_{pi} - Y_{Mi}),$$

and for hidden layers:

$$\delta_{ji} = Y_{ji}(1 - Y_{ji}) \sum_{k=1}^{N_{j+1}} \delta_{(j+1)k} W_{(j+1)ki}.$$

The weight adjustment can be done for every connection from neuron k in layer $(j - 1)$ to every neuron j in every layer i:

$$W_{jik}^{+} = W_{jik} + \eta \delta_{ji} Y_{ji},$$

where η represents weight adjustment factor (called the *learning rate*) normalized between 0 and 1.

Step 7.The actions in steps 2 through 6 will be repeated for every training sample pattern p, and repeated for these sets until the sum of the squares of output errors is minimized.

Introduction to deep learning

Definition

A neural network with multiple hidden layers is called a *Deep Neural Network* (DNN) and the practice of training such network is referred to as *deep learning*.

Remarks

In the terminology “deep learning”, the term “deep” is a technical term. It refers to the number of layers in a neural network. A *shallow network* has one so-called hidden layer, and a deep network has more than one. Multiple hidden layers allow deep neural networks to learn features of the data in a so-called feature hierarchy, because simple features recombine from one layer to the next, to form more complex features. Networks with many layers pass input data (features) through more mathematical operations than networks with few layers, and are therefore more computationally intensive to train. Computational intensivity is one of the hallmarks of deep learning.

Figure 9.22 shows a shallow neural network and Figure 9.23 shows a deep neural network with three hidden layers.

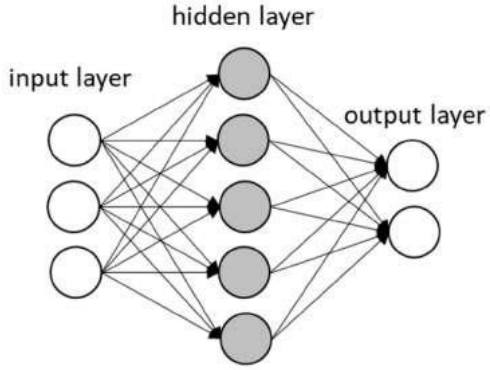


Figure 9.22: A shallow neural network

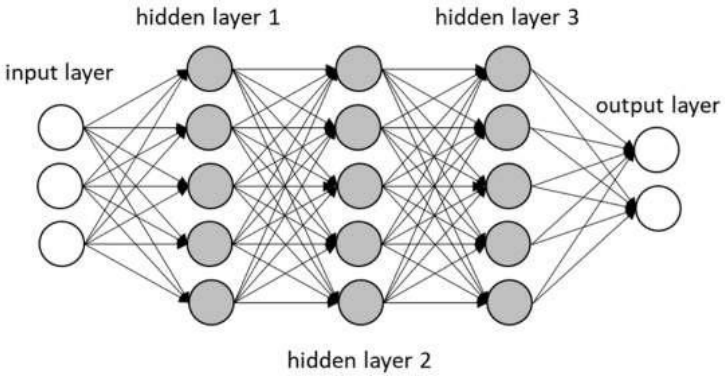


Figure 9.23: A deep neural network with three hidden layers

Some applications

Deep learning applications are used in industries from automated driving to medical devices.

1. **Automated driving**:

   Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

2. **Aerospace and defense**:

   Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.

3. **Medical research**:

   Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.

4. **Industrial automation**:

   Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.

5. **Electronics**:

Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

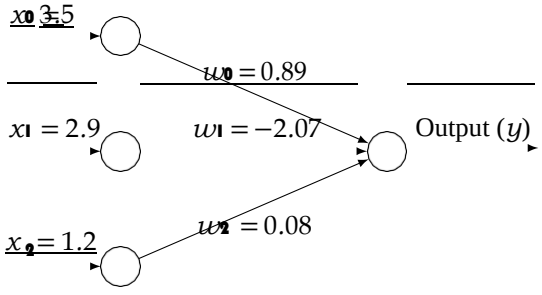
Sample questions

(a) Short answer questions

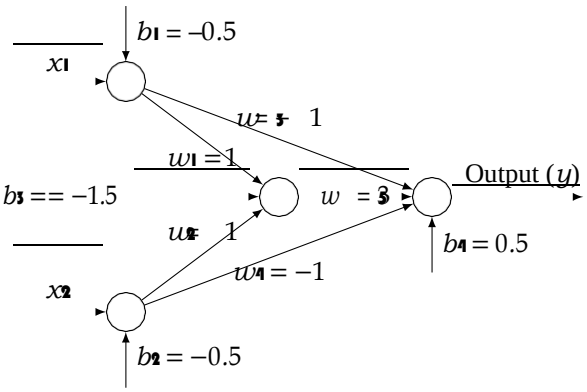
1. Explain the biological motivation for the formulation of the concept of artificial neural networks.
2. With the aid of a diagram, explain the concept of an artificial neuron.
3. What is an activation function in an artificial neuron? Give some examples.
4. Define a perceptron.
5. Is neural network supervised or unsupervised learning? Why?
6. Is deep learning supervised or unsupervised? Why?
7. What is the basic idea of the backpropagation algorithm?
8. In the context of ANNs, what is meant by network topology?
9. Explain the different types of layers in an ANN.
10. What is the gradient descent method? How is used in the backpropagation algorithm?
11. A neuron with 4 inputs has the weights 1, 2, 3, 4 and bias 0. The activation function is linear, say the function $f(x) = 2x$. If the inputs are 4, 8, 5, 6, compute the output. Draw a diagram representing the neuron.

(b) Long answer questions

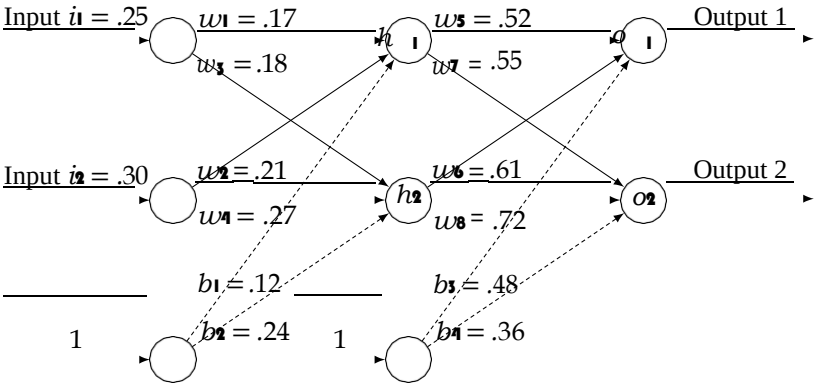
1. Design a two layer network of perceptrons to implement A XOR B.
2. Explain the backpropagation algorithm.
3. Describe the perceptron learning algorithm.
4. What are the characteristics of an artificial neural networks.
5. Explain the concept of deep learning. Give some real life problems where this concept has been successfully applied.
6. Compute the output of the following neuron if the activation function is (i) the threshold function (ii) the sigmoid function (iii) the hyperbolic tangent function (assume the same bias 0.5 for each node).



7. Which of the boolean functions AND, OR, XOR (or none of these) is represented by the following network of perceptrons (with unit step function as the activation function)?



8. Given the following network, compute the outputs from o_1 and o_2 (assume that the activation function is the sigmoid function).



9.(Assignment question) Given the following data, use ANN with one hidden layer, appropriate initial weights and biases to compute the optimal values of the weights. Perform one iteration of the forward and phases of the backpropagation algorithm for each samples.

Sample	Input 1	Input 2	Output target 1	Output target 2
1	1.20	2.30	0.53	0.76
2	0.23	0.37	1.17	2.09

Module 5

Support vector machines

We begin this chapter by illustrating the basic concepts and terminology of the theory of support vector machines by a simple example. We then introduce the necessary mathematical background, which is essentially an introduction to finite dimensional vector spaces, for describing the general concepts in the theory of support vector machines. The related algorithms without proofs are then presented.

An example

Problem statement

Suppose we want to develop some criteria for determining the weather conditions under which tennis can be played. To simplify the matters it has been decided to use the measures of temperature and humidity as the critical parameters for the investigation. We have some data as given in Table 10.1 regarding the values of the parameters and the decisions taken as to whether to play tennis or not. We are required to develop a criteria to know whether one would be playing tennis on a future date if we know the values of the temperature and humidity of that date in advance.

Discussion and solution

We shall now see the various steps that lead to a solution of the problem using the ideas of support vector machines.

temperature	humidity	play
85	85	no
60	70	yes
80	90	no
72	95	no
68	80	yes
74	73	yes
69	70	yes
75	85	no
83	78	no

Table 10.1: Example data with two class labels

1. Two-class data set

This is our first observation regarding the data in Table 10.1. In Table 10.1, the data are classified based on the values of the variable "play". This variable has only two values or labels, namely "yes" and "no". When there are only two class labels the data is said to be a "two-class data set". So the data in Table 10.1 is a two-class data set.

2. Scatter plot of the data

Since there are only two features or parameters, we may plot the values of one of the parameters, say "temperature", along the horizontal axis (that is, the x-axis) and the values of the other parameter "humidity", along the vertical axis (that is, the y-axis). The data can be plotted in a coordinate plane to get a scatter plot of the data. Figure 10.1 shows the scatter plot. In the figure the points which correspond to the decision "yes" on playing tennis has been plotted as filled squares (▪) and which correspond to the decision "no" has been marked as hollow circles (○).

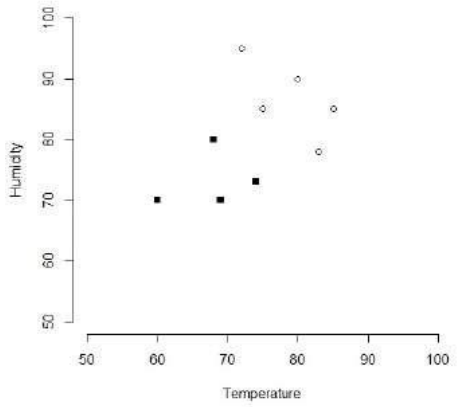


Figure 10.1: Scatter plot of data in Table 10.1 (filled circles represent "yes" and unfilled circles "no")

3. A separating line

If we examine the plot in Figure 10.1, we can see that we can draw a straight line in the plane separating the two types of points in the sense that all points plotted as filled squares are on one side of the line and all points marked as hollow circles are on the other side of the line. Such a line is called a "separating line" for the data. Figure 10.2 shows a separating line for the data in Table 10.1. The equation of the separating line shown in Figure 10.2 is

$$5x + 2y - 535 = 0. \tag{10.1}$$

It has the following property:

- If the data point with values (x', y') has the value "yes" for "play" (filled square), then

$$5x' + 2y' - 535 < 0. \tag{10.2}$$

- If the data point with values (x, y) has the value "no" for "play" (hollow circle), then

$$5x' + 2y' - 535 > 0. \tag{10.3}$$

If such a separating line exists for a given data then the data is said to be "linearly separable". Thus the data in table 10.1 is linearly separable. However note that not all data are linearly separable.

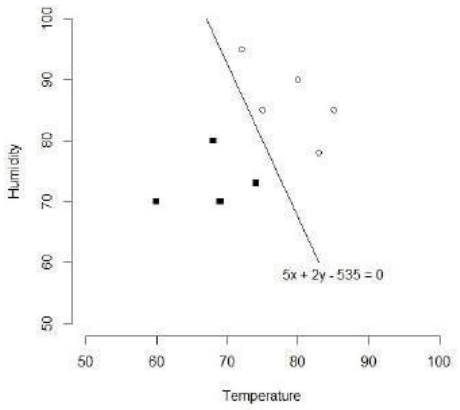


Figure 10.2: Scatter plot of data in Table 10.1 with a separating line

4. Several separating lines

Apparently, the conditions given in Eqs. (10.2) and (10.3) may be used as the criteria to know whether one would be playing tennis on a future date if we know the values of the temperature and humidity of that date in advance. But there are several separating lines and the problem of determining which one to choose arises. Figure 10.3 shows two separating lines for the given data.

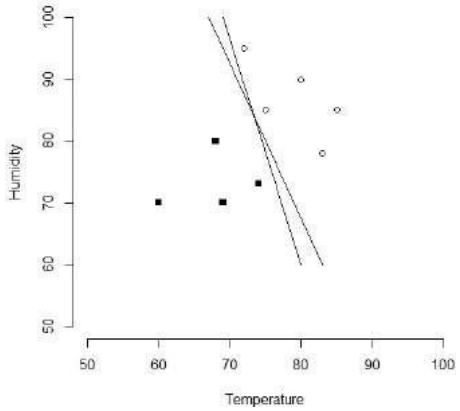


Figure 10.3: Two separating lines for the data in Table 10.1

4. Margin of a separating line

To choose the "best" separating line, we introduce the concept of the margin of a separating line.

Given a separating line for the data, we consider the perpendicular distances of the data points from the separating line. Th double of the shortest perpendicular distance is called the "margin of the separating line". Figure **??** shows some of the perpendicular distances and the shortest perpendicular distance for the data in Table 10.1 and for the separating line given by Eq. (10.1).

5. Maximal margin separating line

The "best" separating line is the one with the maximum margin.

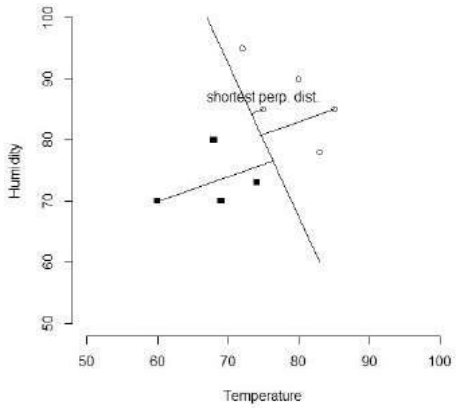


Figure 10.4: Shortest perpendicular distance of a separating line from data points

The separating line with the maximum margin is called the "maximum margin line" or the "optimal separating line". This line is also called the "support vector machine" for the data in Table 10.1.

Unfortunately, finding the equation of the maximum margin line is not a trivial problem. Figure 10.5 shows the maximum margin line for the data in Table 10.1. The equation of the maximum margin line can be shown to be

$$7x+6y-995.5=0. \tag{10.4}$$

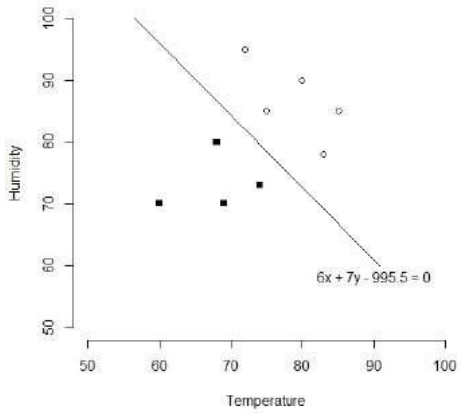


Figure 10.5: Maximum margin line for data in Table 10.1

6. Support vectors

The data points which are closest to the maximum margin line are called the "support vectors". The support vectors are shown in Figure 10.6.

7. The required criterion

As per theory of support vector machines, the equation of the maximum margin line is used to devise a criterion for taking a decision on whether to play tennis or not. Let x' and y' be the values

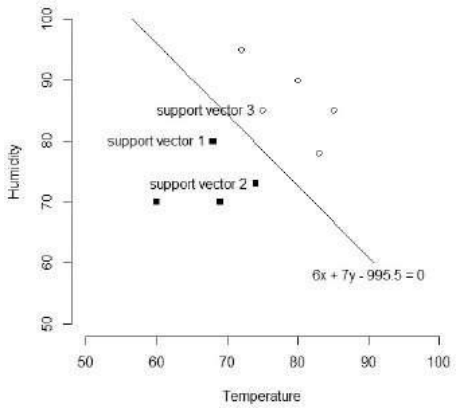


Figure 10.6: Support vectors for data in Table 10.1

of temperature and humidity on a given day. Then the decision as to whether play tennis on that day is "yes" if

$$7x+6y-995.5<0$$

and "no" if

$$7x+6y-995.5>0.$$

8. "Street" of maximum width separating "yes" points and "no" points

Considering Figure 10.6, we may draw a line through the support vectors 1 and 2 parallel to the maximum margin line, and a line through support vector 3 parallel to the maximum margin line. The two lines are shown as dashed lines in Figure 10.7. The region between these two dashed lines can be thought of as a "road" or a "street" of maximum width that separates the "yes" data points and the "no" data points.

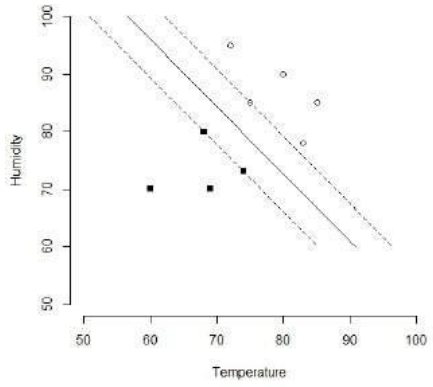


Figure 10.7: Boundaries of "street" of maximum width separating "yes" points and "no" points in Table 10.1

9. Final comments

i) Any line given an equation of the form

$$ax + by + c = 0$$

separates the coordinate plane into two halves. One half consists of all points for which $ax+by+c > 0$ and the other half consists of all points for which $ax+by+c < 0$. Which half is which depends the signs of the coefficients a, b, c.

ii) Figure 10.8 shows the plot of the maximum margin line produced using the R programming language.

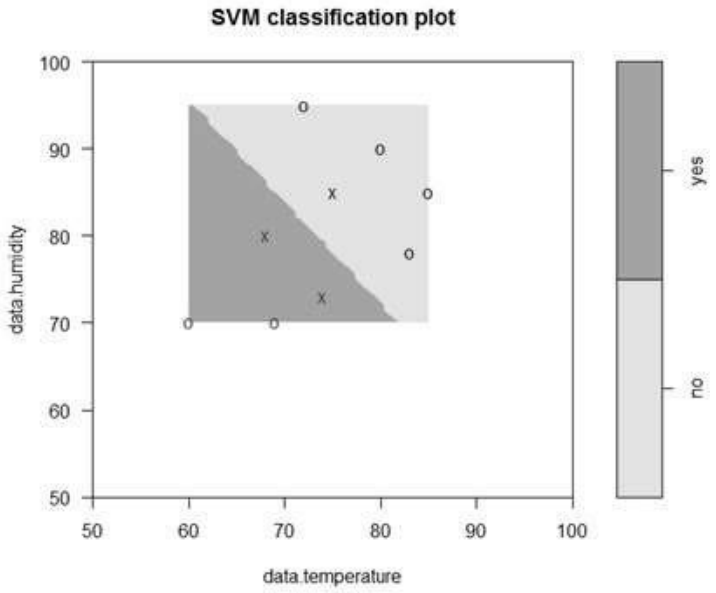


Figure 10.8: Plot of the maximum margin line of data in Table 10.1 produced by the R programming language

iii) In the sections below, we generalise the concepts introduced above to data sets having more than two features.

Finite dimensional vector spaces

In Section 10.1 we have geometrically examined in detail the concepts of the theory of support vector machines with an example having only two features. But, obviously, such a geometrical approach is infeasible if there are more than two features. In such cases we have to resort to formal algebraic/mathematical formalism to investigate the problem. The theory of what are known as "finite dimensional vector spaces" provides such a formalism. We present below the absolutely essential parts of this theory. Those who are interested in learning about the abstract concept of a vector space may refer to any well written book on linear algebra.

Definition

We give the definition of a finite dimensional vector space here. We once again warn the reader that we are introducing the terms with reference to a very special case of a finite dimensional vector

space and that all the terms given below have more general meanings.

Definition

Let n be a positive integer. By a *n-dimensional vector* we mean an ordered n-tuple of real numbers of the form $(x_1, x_2, \ldots, x_n)$. We denote vectors by $\vec{x}, \vec{y}$, etc. In the vector $\vec{x} = (x_1, x_2, \ldots, x_n)$, the numbers $x_1, x_2, \ldots x_n$ are called the *coordinates* or the *components* of $\vec{x}$. In the following, we call real numbers as *scalars*.

The set of all n-dimensional vectors with the operations of *addition of vectors* and *multiplication of a vector by a scalar* and with the definitions of the *zero vector* and the *negative of a vector* as defined below is a n-dimensional vector space. It is denoted by $\mathbb{R}^n$.

1. **Addition of vectors**

   Let $\vec{x} = (x_1, x_2, \ldots, x_n)$ and $\vec{y} = (y_1, y_2, \ldots, y_n)$ be two n-dimensional vectors. The sum of $\vec{x}$ and $\vec{y}$, denoted by $\vec{x}+\vec{y}$, is defined by

   $$\vec{x} + \vec{y} = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n).$$

2. **Multiplication by scalar**

   Let a be a scalar and $\vec{x} = (x_1, x_2, \ldots, x_n)$ be a n-dimensional vector. The product of $\vec{x}$ by a, denoted by $a\vec{x}$, is defined by

   $$a\vec{x} = (ax_1, ax_2, \ldots, ax_n).$$

   When we write the product of $\vec{x}$ by a, we always write the scalar a on the left side of the vector $\vec{x}$ as we have done above.

3. **The zero vector**

   The n-dimensional vector $(0, 0, \ldots, 0)$, which has all components equal to 0, is called the zero vector. It is also denoted by 0. From the context of the usage we can understand whether 0 denotes the scalar 0 or the zero vector.

4. **Negative of a vector**

   Let $\vec{x} = (x_1, x_2, \ldots, x_n)$ be any n-dimensional vector. The negative of $\vec{x}$ is a vector denoted by $-\vec{x}$ and is defined by

   $$-\vec{x} = (-x_1, -x_2, \ldots, -x_n)$$

   We write $\vec{x} + (-\vec{y})$ as $\vec{x} - \vec{y}$.

Properties

Let n be a positive integer. Let $\vec{x}, \vec{y}, \vec{z}$ be arbitrary vectors in $\mathbb{R}^n$ and let a, β, γ be arbitrary scalars.

1 Closure under addition: $\vec{x} + \vec{y}$ is also a n-dimensional vector.

2 Commutativity: $\vec{x} + \vec{y} = \vec{y} + \vec{x}$

3 Associativity: $\vec{x} + (\vec{y} + \vec{z}) = (\vec{x} + \vec{y}) + \vec{z}$

(Because of this property, we can write the sums $\vec{x} + (\vec{y} + \vec{z})$ and $(\vec{x} + \vec{y}) + \vec{z}$ in the form $\vec{x} + \vec{y} + \vec{z}$.)

4 Existence of identity for addition: $\vec{x} + 0 = \vec{x}$

5 Existence of inverse for addition: $\vec{x} + (-\vec{x}) = 0$

6 Closure under scalar multiplication: $a\vec{x}$ is also a n-dimensional vector.

7 Compatibility of multiplication of a vector by a scalar with multiplication of scalars: $a(\beta\dot{x}) = (a\beta)\dot{x}$

8 Distributivity of scalar multiplication over vector addition: $a(\dot{x} + \dot{y}) = a\dot{x} + a\dot{y}$

9.Distributivity of scalar multiplication over addition of scalars: $(a + \beta)\dot{x} = a\dot{x} + \beta\dot{x}$

10.Existence of identity element for scalar multiplication: $1\dot{x} = \dot{x}$

Example of computation

Let $n = 3$. Let $\dot{x} = (-1, 2, 3), \dot{y} = (2, 0, -1), \dot{z} = (1, 1, 0)$, $a = 2, \beta = -3, \gamma = 4$ and $\lambda = 5$. The expression $\lambda(a\dot{x} + \beta\dot{y} + \gamma\dot{z})$ can be computed in several different ways. One of the methods is shown below.

$$\begin{aligned}\lambda(a\dot{x} + \beta\dot{y} + \gamma\dot{z}) &= 5(2(-1, 2, 3) + (-3)(2, 0, -1) + 4(1, 1, 0)) \\ &= 5((-2, 4, 6) + (-6, 0, 3) + (4, 4, 0)) \\ &= 5((-8, 4, 9) + (4, 4, 0)) \\ &= 5(-4, 8, 9) \\ &= (-20, 40, 45)\end{aligned}$$

Norm and inner product

1. **Norm**

   The norm of the n-dimensional vector $\dot{x} = (x_1, x_2, \ldots, x_n)$, denoted by $\|\dot{x}\|$, is defined by

   $$\|\dot{x}\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}.$$

2. **Inner product**

   The inner product of $\dot{x} = (x_1, x_2, \ldots, x_n)$ and $\dot{y} = (y_1, y_2, \ldots, y_n)$, denoted by $\dot{x} \cdot \dot{y}$, is defined by

   $$\dot{x} \cdot \dot{y} = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n.$$

   Note that we have

   $$\|\dot{x}\| = \sqrt{\dot{x} \cdot \dot{x}}.$$

3. **Angle between two vectors**

   The angle θ between two vectors $\dot{x}$ and $\dot{y}$ is defined by

   $$\cos\theta = \frac{\dot{x} \cdot \dot{y}}{\|\dot{x}\| \|\dot{y}\|}.$$

4. **Perpendicularity**

   Two vectors $\dot{x} = (x_1, x_2, \ldots, x_n)$ and $\dot{y} = (y_1, y_2, \ldots, y_n)$ are said to be perpendicular (or, orthogonal) if

   $$\dot{x} \cdot \dot{y} = 0.$$

Example

Let $n = 4$ and let $\dot{x} = (-1, 2, 0, 3)$ and $\dot{y} = (2, 3, 1, -4)$.

$$\|\dot{x}\| = \sqrt{(-1)^2 + 2^2 + 0^2 + 3^2}$$
$$= \sqrt{14}$$
$$\|\dot{y}\| = \sqrt{2^2 + 3^2 + 1^2 + (-4)^2}$$
$$= \sqrt{30}$$
$$\dot{x} \cdot \dot{y} = (-1) \times 2 + 2 \times 3 + 0 \times 1 + 3 \times (-4)$$
$$= -8$$
$$\cos\theta = \frac{-8}{\sqrt{14}\sqrt{30}}$$
$$= -0.39036$$
$$\theta = 112.98 \text{ degrees}$$

Since $\dot{x} \cdot \dot{y} \neq 0$ the vectors $\dot{x}$ and $\dot{y}$ are not orthogonal.

Hyperplanes

Hyperplanes are certain subsets of finite dimensional vector spaces which are similar to straight lines in planes and planes in three-dimensional spaces.

Definition

Consider the n-dimensional vector space R^n. The set of all vectors

$$\dot{x} = (x_1, x_2, \ldots, x_n)$$

in R^n whose components satisfy an equation of the form

$$a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = 0, \tag{10.5}$$

where $a_0, a_1, a_2, \ldots, a_n$ are scalars, is called a **hyperplane** in the vector space R^n.

Remarks 1

Let $\dot{x} = (x_1, x_2, \ldots, x_n)$ and $\dot{a} = (a_1, a_2, \ldots, a_n)$, then using the notation of inner product, Eq.(10.5) can be written in the following form:

$$a_0 + \dot{a} \cdot \dot{x} = 0.$$

Remarks 2

The hyperplane in R^n defined by Eq.(10.5) divides the space R^n into two disjoint halves. One of the two halves consists of all vectors $\dot{x}$ for which

$$a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n > 0$$

and the other half consists of all vectors $\dot{x}$ for which

$$a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n < 0.$$

Special cases

Hyperplanes in 2-dimensional vector spaces: Straight lines

Consider the 2-dimensional vector space R^2. Vectors in this space are ordered pairs of the form (x_1, x_2). Choosing appropriate coordinate axes, such a vector can be represented by a point with coordinates $\dot{x} = (x_1, x_2)$ in the plane. So, the vector space R^2 can be identified with the set of points in a plane. In this special case, the norm $\|x\|$ is the distance of the point (x_1, x_2) in the plane from the origin. The angle between the vectors $\dot{x} = (x_1, x_2)$ and $\dot{y} = (y_1, y_2)$ is the angle between the lines joining the origin to the points (x_1, x_2) and (y_1, y_2).

Consider the set of all vectors $\dot{x} = (x_1, x_2)$ in R^2 which satisfy the following equation:

$$a_0 + a_1 x_1 + a_2 x_2 = 0$$

where a_0, a_1, a_2 are scalars. From elementary analytical geometry we can see that the corresponding set of points in the plane form a straight line in the plane. This straight line divides the plane into two disjoint halves (see Figure 10.9). It can be proved that one of the two halves consists of all points for which

$$a_0 + a_1 x_1 + a_2 x_2 > 0$$

and the other half consists of all points for which

$$a_0 + a_1 x_1 + a_2 x_2 < 0.$$

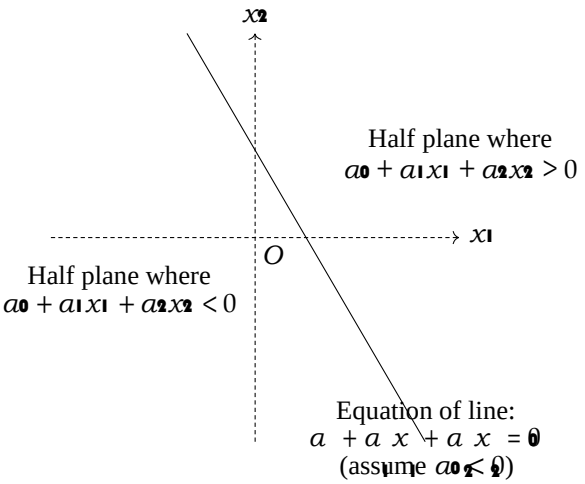


Figure 10.9: Half planes defined by a line

Hyperplanes in 3-dimensional vector spaces: Planes

Consider the 3-dimensional vector space R^3. Vectors in this space are ordered triples of the form (x_1, x_2, x_3). Choosing appropriate coordinate axes, such a vector can be represented by a point with coordinates $x = (x_1, x_2, x_3)$ in the ordinary three-dimensional space. So, the vector space R^3 can be identified with the set of points in the three-dimensional space. As in the case of R^2, the norm $\|x\|$ is the distance of the point (x_1, x_2, x_3) from the origin. The angle between the vectors $\dot{x} = (x_1, x_2, x_3)$ and $\dot{y} = (y_1, y_2, y_3)$ is the angle between the lines joining the origin to the points (x_1, x_2, x_3) and (y_1, y_2, y_3).

Consider the set of all vectors $\dot{x} = (x_1, x_2, x_3)$ in R^3 which satisfy the following equation:

$$a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 = 0$$

where a_0, a_1, a_2, a_3 are scalars. From elementary analytical geometry we can see that the corresponding set of points in space form a plane. This plane divides the space into two disjoint halves. It can be proved that one of the two halves consists of all points for which

$$a_0 + a_1x_1 + a_2x_2 + a_3x_3 > 0$$

and the other half consists of all points for which

$$a_0 + a_1x_1 + a_2x_2 + a_3x_3 < 0.$$

Geometry of hyperplanes in n-dimensional vector spaces

By analogy with a plane (which is a geometrical object having two dimensions) and the space of our experience (which is a geometrical world having three dimensions) we imagine that there is a geometrical world or object having n dimensions for any value of n. We also imagine that the points in this world can be represented by ordered n tuples of the form $x = (x_1, x_2, \ldots, x_n)$. We now identify the set of n-dimensional vectors with the points in this geometrical world of n-dimensions. Because of this identification, vectors in the n-dimensional vector space $\mathbb{R}^n$ are also referred as *points in a n-dimensional space*. The hyperplanes in $\mathbb{R}^n$ are defined by analogy with the geometrical straight lines and planes.

Distance of a hyperplane from a point

In two-dimensional space, that is, in a plane, using elementary analytical geometry, it can be shown that the perpendicular distance PN of a point $P(x'_1, y'_1)$ from a line

$$a_0 + a_1x_1 + a_2x_2 = 0$$

is given by

$$PN = \frac{|a_0 + a_1x'_1 + a_2x'_2|}{\sqrt{a_1^2 + a_2^2}}.$$

Similarly, in three-dimensional space, using elementary analytical geometry, it can be shown that the perpendicular distance PN of a point $P(x'_1, x'_2, x'_3)$ from a plane

$$a_0 + a_1x_1 + a_2x_2 + a_3x_3 = 0$$

is given by (see Figure 10.10)

$$PN = \frac{|a_0 + a_1x'_1 + a_2x'_2 + a_3x'_3|}{\sqrt{a_1^2 + a_2^2 + a_3^2}}.$$

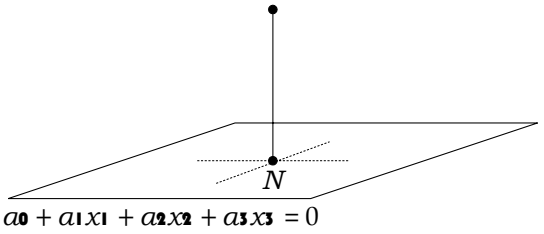


Figure 10.10: Perpendicular distance of a point from a plane

Motivated by these special cases, we introduce the following definition.

Definition

In $\mathbb{R}^n$, the *perpendicular distance PN* of a point $P(x'_1, x'_2, \ldots, x'_n)$ from a hyperplane

$$a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0$$

is given by

$$PN = \frac{|a_0 + a_1x'_1 + a_2x'_2 + \ldots + a_nx'_n|}{\sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}}. \tag{10.6}$$

Remarks

Let $\dot{x}' = (x'_1, x'_2, \ldots, x'_n)$ and $\dot{a} = (a_1, a_2, \ldots, a_n)$, then using the notations of inner product and norm, Eq.(10.6) can be written in the following form:

$$PN = \frac{|a_0 + \dot{a} \cdot \dot{x}'|}{\|\dot{a}\|}.$$

Two-class data sets

In a machine learning problem, the variable being predicted is called the *output variable*, the *target variable*, the *dependent variable* or the *response*. A *two-class data* set is a data set in which the target variable takes only one of two possible values only. If the target variable takes more than two possible values, the data set is called a *multi-class dataset*.

In a two-class data set, the set of values of the target variable may be {"yes", "no"}, or {"TRUE", "FALSE"}, or $\{0, 1\}$, or $\{-1, +1\}$ or any such similar set.

The methods of support vector machines were originally developed for classification problems involving two-class data sets. So in this chapter we consider mainly two-class data sets.

Linearly separable data

Definitions

Consider a two -class data set having n numeric features and two possible class labels -1 and $+1$. Let the vector $\dot{x} = (x_1, \ldots, x_n)$ represent the values of the features in one instance of the data set. We say that the data set is *linearly separable* if we can find a hyperplane in the n-dimensional vector space $\mathbb{R}^n$, say

$$a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0 \tag{10.7}$$

having the following two properties:

1. For each instance $\dot{x}$ with class label -1 we have

$$a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n < 0.$$

2. For each instance $\dot{x}$ with class label $+1$ we have

$$a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n > 0.$$

A hyperplane given by Eq.(10.7) having the two properties given above is called a *separating hyperplane* for the data set.

Remarks 1

If a data set with two class labels is linearly separable, then, in general, there will be several separating hyperplanes for the data set. This is illustrated in the example below.

Remarks 2

Given a two-class data set, there is no simple method for determining whether the data set is linearly separable. One of the efficient ways for doing this is to apply the methods of linear programming. We omit the details.

Example

Example 1

We have seen in Section 10.1 that the data in Table 10.1 is linearly separable.

Example 2

Show that the data set given in Table 10.2 is not separable.

x	y	Class label
0	0	0
0	1	1
1	0	1
1	1	0

Table 10.2: Example of a two-class data that is not linearly separable

Solution

The scatterplot of data in TableTableVXOR shown in Figure 10.11 shows that the data is not linearly separable.

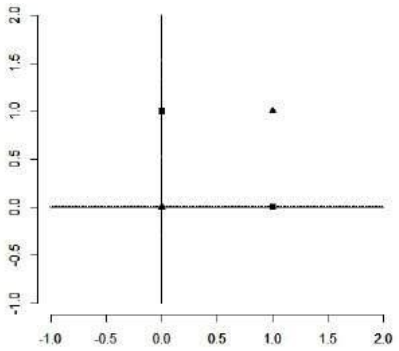


Figure 10.11: Scatterplot of data in Table 10.2

Maximal margin hyperplanes

Definitions

Consider a linearly separable data set having two class labels "−1" and "1". Consider a separating hyperplane H for the data set.

1. Consider the perpendicular distances from the training instances to the separating hyperplane H and consider the smallest such perpendicular distance. The double of this smallest distance is called the *margin* of the separating hyperplane H.

2. The hyperplane for which the margin is the largest is called the *maximal margin hyperplane* (also called *maximum margin hyperplane*) or the *optimal separating hyperplane*.

3. The maximal margin hyperplane is also called the *support vector machine* for the data set.

4. The data points that lie closest to the maximal margin hyperplane are called the *support vectors*.

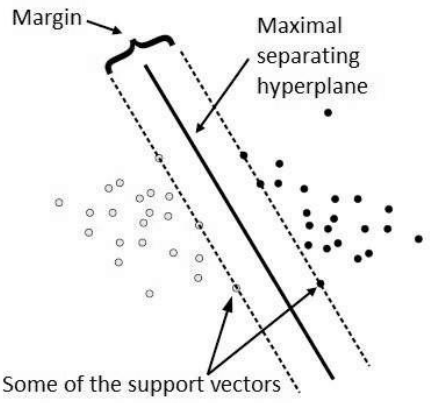


Figure 10.12: Maximal separating hyperplane, margin and support vectors

Special cases

To fix ideas, let us consider two special datasets in 2-dimensional space, namely, datasets having 2 and 3 examples.

Dataset with two examples

Consider the dataset in Table 10.3.

Example no.	x_1	x_2	Class
1	2	1	1
2	4	3	−1

Table 10.3: 2-dimensional dataset with 2 examples

Geometrically it can be easily seen that the maximum margin hyperplane for this data is the perpendicular bisector of the line segment joining the points $(2, 1)$ and $(4, 3)$ (see Figure 10.13). This is true for any two-sample dataset in two-dimensional space.

Dataset with three examples

Consider a dataset with three examples from a two-dimensional space. Let these examples correspond to the points A, B, C in the coordinate plane. Two of these examples, say B and C, must have the same class label say 1 and the other point A must have a different class label, say -1.

The maximal margin hyperplane of the dataset can be obtained as follows. Draw the line joining B and C and draw the line through A parallel to BC. The line midway between these two lines in the maximal margin hyperplane of the three-sample dataset

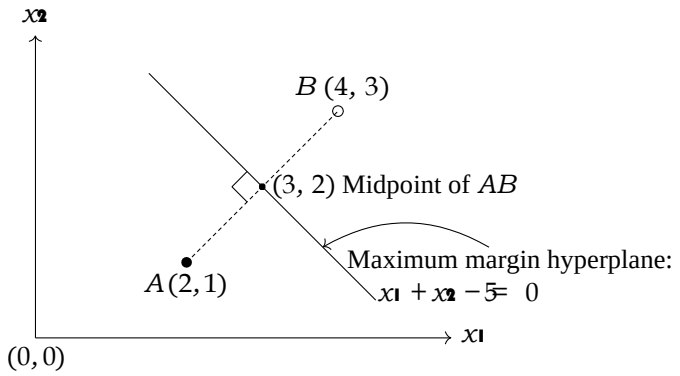


Figure 10.13: Maximal margin hyperplane of a 2-sample set in 2-dimensional space

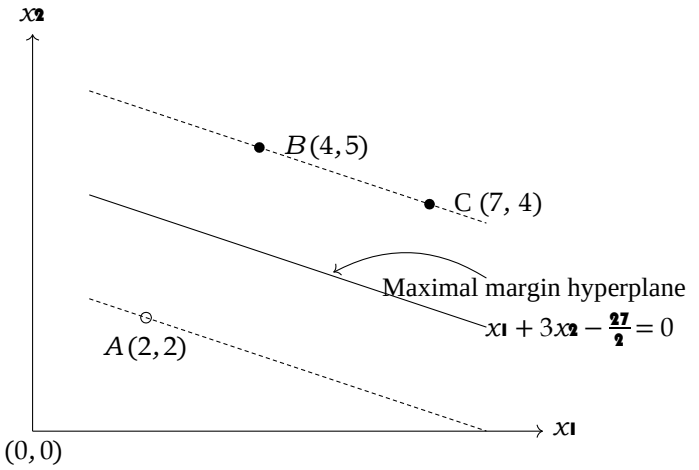


Figure 10.14: Maximal margin hyperplane of a 3-sample set in 2-dimensional space

Mathematical formulation of the SVM problem

The SVM problem is the problem of finding the equation of the SVM, that is, the maximal margin hyperplane, given a linearly separable two-class data set. By the very definition of SVM, this is an optimisation problem. The give below the mathematical formulation of this optimisation problem.

Notations and preliminaries

- Assume that we are given a two-class training dataset of N points of the form

$$(\vec{x}_1,\, y_1\,,\, \vec{x}_2,\, y_2\,,\, \ldots,\, \vec{x}_N,\, y_N\,).$$

  where the y_i's are either -1 or 1 (the class labels). Each $\vec{x}_i$ is a n-dimensional real vector.

- We assume that the dataset is linearly separable.

- Any hyperplane can be written as the set of points $\vec{x} = (x_1, \ldots, x_n)$ satisfying an equation of the form

$$\vec{w} \cdot \vec{x} - b = 0.$$

- Since the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The maximum margin hyperplane is the hyperplane that lies halfway between them. It can be shown that these hyperplanes can be described by equations of the following forms:

$$\vec{w} \cdot \vec{x} - b = +1 \tag{10.8}$$

$$\vec{w} \cdot \vec{x} - b = -1 \tag{10.9}$$

- For any point on or "above" the hyperplane Rq.(10.8), the class label is +1. This implies that

$$\vec{w} \cdot \vec{x}_i - b \geq +1, \text{ if } y_i = +1 \tag{10.10}$$

Similarly, for any point on or "below" the hyperplane Eq.(10.9), the class label is 1. This implies that

$$\vec{w} \cdot \vec{x}_i - b \leq -1, \text{ if } y_i = -1. \tag{10.11}$$

- The two conditions in Eq.10.10 and Eq.10.11 can be written as a single condition as follows:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq N.$$

- Now, the distance between the two hyperplanes in Eq.(10.8) and Eq.(10.9) is

$$\frac{2}{\|\vec{w}\|}.$$

So, to maximize the distance between the planes we have to minimize $\|\vec{w}\|$. Further we also note that $\|\vec{w}\|$ is minimum when $\frac{1}{2}\|\vec{w}\|^2$ is minimum. (The square of the norm is used to avoid square-roots and the factor "$\frac{1}{2}$" is introduced to simplify certain expressions.)

Formulation of the problem

Based on the above discussion, we now formulate the SVM problem as the following optimization problem.

Problem

Given a two-class linearly separable dataset of N points of the form

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_N, y_N).$$

where the y_i's are either -1 or 1, find a vector $\vec{w}$ and a number b which

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\vec{w}\|^2 \\ &\text{subject to} && y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \text{ for } i = 1, \ldots N \end{aligned}$$

The SVM classifier

The solution of the SYM problem gives us a claasifier for classifying unclassified data instances. This is known as the *SVM classifier* for a given dataset.

The classifier

Let $\vec{w} = \vec{w}^*$ and $b = b^*$ be a solution of the SVM problem. Let $\vec{x}$ be an unclassified data instance.

- Assign the class label $+1$ to $\vec{x}$ if $\vec{w}^* \cdot \vec{x} - b^* > 0$.
- Assign the class label -1 to $\vec{x}$ if $\vec{w}^* \cdot \vec{x} - b^* < 0$.

Solution of the SVM problem

The SVM optimization problem as formulated above is an example of a constrained optimization problem. The general method for solving it is to convert it into a quadratic programming problem and then apply the algorithms for solving quadratic programming problems. These methods yield the following solution to the SVM problem. The details of these processes are beyond the scope of these notes.

Solution

The vector $\dot{w}$ and the scalar b are given by

$$\dot{w} = \sum_{i=1}^{N} a_i y_i \dot{x}_i \tag{10.12}$$

$$b = \frac{1}{2}\left(\min_{\dot{x}_i y_i = +1} (\dot{w} \cdot \dot{x}_i) + \max_{\dot{x}_i y_i = -1} (\dot{w} \cdot \dot{x}_i)\right) \tag{10.13}$$

where $\dot{a} = (a_1, a_2, \ldots, a_N)$ is a vector which maximizes

$$\sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1, j=1}^{N} a_i a_j y_i y_j (\dot{x}_i \cdot \dot{x}_j)$$

subject to

$$\sum_{i=1}^{N} a_i y_i = 0$$

$$a_i > 0 \text{ for } i = 1, 2, \ldots, N.$$

Remarks

It can be proved that an a_i is nonzero only if $\dot{x}_i$ lies on the two margin boundaries, that is, only if $\dot{x}_i$ is a support vector. So, to specify a solution to the SVM problem, we need only specify the support vectors $\dot{x}_i$ and the corresponding coefficients $a_i y_i$.

An algorithm to find the SVM classifier

The solution of the SVM problem given in Section **??** can be used to develop an algorithm to find a SVM classifier for linearly separable two-class dataset. Here is an outline of such an algorithm.

Algorithm to find SVM classifier

Given a two-class linearly separable dataset of N points of the form

$$(\dot{x}_1, y_1), (\dot{x}_2, y_2), \ldots, (\dot{x}_N, y_N),$$

where the y_i's are either $+1$ or 1:

Step 1.Find $\dot{a} = (a_1, a_2, \ldots, a_N)$ which maximizes

$$\varphi(\dot{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1, j=1}^{N} a_i a_j y_i y_j (\dot{x}_i \cdot \dot{x}_j)$$

subject to

$$\sum_{i=1}^{N} a_i y_i = 0$$

$$a_i > 0 \text{ for } i = 1, 2, \ldots, N.$$

Step 2.Compute $\vec{w} = \sum_{i=1}^{N} a_i y_i \vec{x}_i$.

Step 3.Compute $b = \frac{1}{2}(\min_{i y_i=+1}(\vec{w} \cdot \vec{x}_i) + \max_{i y_i=-1}(\vec{w} \cdot \vec{x}_i))$.

Step 4.The SVM classifier function is given by

$$f(\vec{x}) = \vec{w} \cdot \vec{x} - b \tag{10.14}$$

where a_i is nonzero only if $\vec{x}_i$ is a support vector.

Remarks

There are specialised software packages for solving the SVM optimization problem. For example, there is a special package called `svm`in the R programming language to solve such problems.

Illustrative example

Problem 1

Using the SVM algorithm, find the SVM classifier for the follwoing data.

Example no.	x_1	x_2	Class
1	2	1	1
2	4	3	−1

Solution

For this data we have:

$$N = 2$$
$$\vec{x}_1 = (2,1), \quad y_1 = +1$$
$$\vec{x}_2 = (4,3), \quad y_2 = -1$$
$$\vec{a} = (a_1, a_2)$$

Step 1.We have:

$$\varphi(\vec{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1,j=1}^{N} a_i a_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$$
$$= (a_1 + a_2) - \frac{1}{2}[a_1 a_1 y_1 y_1 (\vec{x}_1 \cdot \vec{x}_1) + a_1 a_2 y_1 y_2 (\vec{x}_1 \cdot \vec{x}_2) +$$
$$a_2 a_1 y_2 y_1 (\vec{x}_2 \cdot \vec{x}_1) + a_2 a_2 y_2 y_2 (\vec{x}_2 \cdot \vec{x}_2)]$$
$$= (a_1 + a_2) -$$
$$\frac{1}{2}[a_1 a_1 (+1)(+1)(2 \times 2 + 1 \times 1) + a_1 a_2 (+1)(-1)(2 \times 4 + 1 \times 3) +$$
$$a_2 a_1 (-1)(+1)(4 \times 2 + 3 \times 1) + a_2 a_2 (-1)(-1)(4 \times 4 + 3 \times 3)]$$
$$= (a_1 + a_2) - \frac{1}{2}[5a_1^2 - 22 a_1 a_2 + 25 a_2^2]$$

$$\sum_{i=1}^{N} a_i y_i = a_1 y_1 + a_2 y_2$$
$$= a_1 - a_2$$

We have to solve the following problem.

Problem

Find values of a_1 and a_2 which maximizes

$$\varphi(\dot{a}) = (a_1 + a_2) - \frac{1}{2}[5a_1^2 - 22a_1a_2 + 25a_2^2]$$

subject to the conditions

$$a_1 - a_2 = 0, \quad a_1 > 0, \; a_2 > 0.$$

Solution

To find the required values of a_1 and a_2, we note that from the constraints we have $a_2 = a_1$. Using this in the expression for φ we get

$$\varphi(\dot{a}) = 2a_1 - 4a_1^2.$$

For φ to be maximum we must have

$$\frac{d\varphi}{da_1} = 2 - 8a_1 = 0$$

that is

$$a_1 = \frac{1}{4}$$

and so we also have

$$a_2 = \frac{1}{4}.$$

(For this value of a_1, clearly $\frac{d^2f}{da_1^2} < 0$ and f is indeed maximum. Also we have $a_1 > 0$ and $a_2 > 0$.)

Step 2.Now we have

$$\begin{aligned}\dot{w} &= \sum_{i=1}^{2} a_i y_i \dot{x}_i \\ &= a_1 y_1 \dot{x}_1 + a_2 y_2 \dot{x}_2 \\ &= \frac{1}{4}(+1)(2,1) + \frac{1}{4}(-1)(4,3) \\ &= \frac{1}{4}(-2,-2) \\ &= (-\tfrac{1}{2}, -\tfrac{1}{2})\end{aligned}$$

Step 3.Next we find

$$\begin{aligned}b &= \frac{1}{2}\left(\min_{i\,y_i=+1}(\dot{w}\cdot\dot{x}_i) + \max_{i\,y_i=-1}(\dot{w}\cdot\dot{x}_i)\right) \\ &= \frac{1}{2}\left((\dot{w}\cdot\dot{x}_1) + (\dot{w}\cdot\dot{x}_2)\right) \\ &= \frac{1}{2}\left((-\tfrac{1}{4}\times 2 - \tfrac{1}{4}\times 1) + (-\tfrac{1}{2}\times 4 - \tfrac{1}{2}\times 3)\right) \\ &= \frac{1}{2}\left(-\frac{10}{2}\right) \\ &= -\frac{5}{2}\end{aligned}$$

Step 4.Let $\dot{x} = (x_1, x_2)$. The SVM classifier function is given by

$$f(\dot{x}) = \dot{w} \cdot \dot{x} + b$$
$$= \left(-\frac{1}{2}, -\frac{1}{2}\right) \cdot (x_1, x_2) - \left(-\frac{5}{2}\right)$$
$$= -\frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{5}{2}$$
$$= -\frac{1}{2}(x_1 + x_2 - 5)$$

Step 5.The equation of the maximal margin hyperplane is

$$f(\dot{x}) = 0$$

that is

$$-\frac{1}{2}(x_1 + x_2 - 5) = 0$$

that is

$$x_1 + x_2 - 5 = 0.$$

Note that this the equation of the perpendicular bisector of the line segment joining the points $(2, 1)$ and $(4, 3)$ (see Figure 10.13).

Problem 2

Using the SVM algorithm, find the SVM classifier for the follwoing data.

Example no.	x_1	x_2	Class
1	2	2	1
2	4	5	1
3	7	4	+1

Solution

For this data we have:

$$N = 3$$
$$\dot{x}_1 = (2, 2), \quad y_1 = -1$$
$$\dot{x}_2 = (4, 5), \quad y_2 = +1$$
$$\dot{x}_3 = (7, 4), \quad y_3 = +1$$
$$\dot{a} = (a_1, a_2, a_3)$$
$$\dot{x} = (x_1, x_2)$$

Srep 1.We have

$$\varphi(\dot{a}) = \sum_{i=1}^{3} a_i - \frac{1}{2} \sum_{i=1, j=1}^{3} a_i a_j y_i y_j (\dot{x}_i \cdot \dot{x}_j)$$
$$= \sum_{i=1}^{3} a_i - \frac{1}{2} \sum_{i=1, j=1}^{3} a_i a_j y_i y_j (\dot{x}_i \cdot \dot{x}_j)$$

We have

$$(\dot{x}_1 \cdot \dot{x}_1) = 08, \quad (\dot{x}_1 \cdot \dot{x}_2) = 18, \quad (\dot{x}_1 \cdot \dot{x}_3) = 22$$
$$(\dot{x}_2 \cdot \dot{x}_1) = 18, \quad (\dot{x}_2 \cdot \dot{x}_2) = 41, \quad (\dot{x}_2 \cdot \dot{x}_3) = 48,$$
$$(\dot{x}_3 \cdot \dot{x}_1) = 22, \quad (\dot{x}_3 \cdot \dot{x}_2) = 48, \quad (\dot{x}_3 \cdot \dot{x}_3) = 65$$

Substituting these and simplifying we get

$$\varphi(\dot{a}) = (a_1 + a_2 + a_3) - \frac{1}{2}[8a_1^2 + 41a_2^2 + 65a_3^2 - 36a_1a_2 - 44a_1a_3 + 96a_2a_3]$$

We also have

$$\sum_{i=1}^{N} a_i y_i = -a_1 + a_2 + a_3$$

Now we have to solve the following problem.

Problem

Find $\dot{a} = (a_1, a_2, a_3)$ which maximizes

$$\varphi(\dot{a}) = (a_1 + a_2 + a_3) - \frac{1}{2}[8a_1^2 + 41a_2^2 + 65a_3^2 - 36a_1a_2 - 44a_1a_3 + 96a_2a_3]$$

subject to the conditions

$$-a_1 + a_2 + a_3 = 0, \quad a_1 > 0,\ a_2 > 0,\ a_3 > 0.$$

Solution

From the constraints we have

$$a_1 = a_2 + a_3.$$

Using this in the expression for $\varphi(\dot{a})$ and simplifying we get

$$\varphi(\dot{a}) = 2(a_2 + a_3) - \frac{1}{2}(13a_2^2 + 32a_2a_3 + 29a_3^2)$$

When $\varphi(\dot{a})$ is maximum we have

$$\frac{\partial \varphi}{\partial a_2} = 0, \quad \frac{\partial \varphi}{\partial a_3} = 0 \tag{10.15}$$

that is

$$2 - 13a_2 - 16a_3 = 0, \quad 2 - 16a_2 - 29a_3 = 0.$$

Solving these equations we get

$$a_2 = \frac{26}{121}, \quad a_3 = -\frac{6}{121}$$

Hence

$$a_1 = \frac{26}{121} - \frac{6}{121} = \frac{20}{121}.$$

(The conditions given in Eq.(**??**) are only necessary conditions for getting a maximum value for $\varphi(\dot{a})$. It can be shown that the values for a_2 and a_3 obtained above do indeed satisfy the sufficient conditions for yielding a maximum value of $\varphi(a)$.)

Srep 2.Now we have

$$\dot{w} = \sum_{i=1}^{N} a_i y_i \dot{x}_i$$

$$= \frac{20}{121}(-1)(2, 2) + \frac{26}{121}(+1)(4, 5) - \frac{6}{121}(+1)(7, 4)$$

$$= (\frac{11}{2}, \frac{11}{6})$$

Srep 3.We have

$$\begin{aligned} b &= \frac{1}{2}\left(\min_{\vec{x}_i y_i=+1}(\vec{w}\cdot\vec{x}_i) + \max_{\vec{x}_i y_i=-1}(\vec{w}\cdot\vec{x}_i)\right) \\ &= \frac{1}{2}\left(\min\{(\vec{w}\cdot\vec{x}_2),(\vec{w}\cdot\vec{x}_3)\} + \max\{(\vec{w}\cdot\vec{x}_1)\}\right) \\ &= \frac{1}{2}\left(\min\{\tfrac{38}{11},\tfrac{38}{11}\} + \max\{\tfrac{16}{11}\}\right) \\ &= \frac{1}{2}\left(\frac{38}{11}+\frac{16}{11}\right) \\ &= \frac{27}{11} \end{aligned}$$

Srep 4.The SVM classifier function is

$$\begin{aligned} f(\vec{x}) &= \vec{w}\cdot\vec{x} - b \\ &= \frac{2}{11}x_1 + \frac{6}{11}x_2 - \frac{27}{11}. \end{aligned}$$

Srep 5.The equation of the maximal hyperplane is

$$f(\vec{x}) = 0$$

that is

$$\frac{2}{11}x_1 + \frac{6}{11}x_2 - \frac{27}{11} = 0$$

that is

$$x_1 + 3x_2 - \frac{27}{2} = 0.$$

(See Figure 10.14.)

Soft margin hyperlanes

The algorithm for finding the SVM classifier will give give a solution only if the the given two-class dataset is linearly separable. But, in real life problems, two-class datasets are only rarely linearly separable. In such a case we introduce additional variables, ξ_i, called *slack variables* which store deviations from the margin. There are two types of deviation: An instance may lie on the wrong side of the hyperplane and be misclassified. Or, it may be on the right side but may lie in the margin, namely, not sufficiently away from the hyperplane (see Figure 10.15).

If $\xi_i = 0$, then $\vec{x}_i$ is correctly classified and there is no problem with $\vec{x}_i$. If $0 < \xi_i < 1$ then $\vec{x}_i$ is correctly classified but it is in the margin. If $\xi_i > 1$, $\vec{x}_i$ is misclassified. Th sum $\sum_{i=1}^{N} \xi_i$ is defined as the *soft error* and this is added as a penalty to the function to be minimized. We also introduce a factor C to the soft error.

With these modifications, we now reformulate the SVM problem as follows (see Section 10.7.2 for the original formulation of the problem):

Reformulated problem

Given a two-class linearly separable dataset of N points of the form

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_N, y_N).$$

where the y_i's are either +1 or 1, find vectors $\vec{w}$ and $\vec{\xi}$ and a number b which

minimize

$$\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

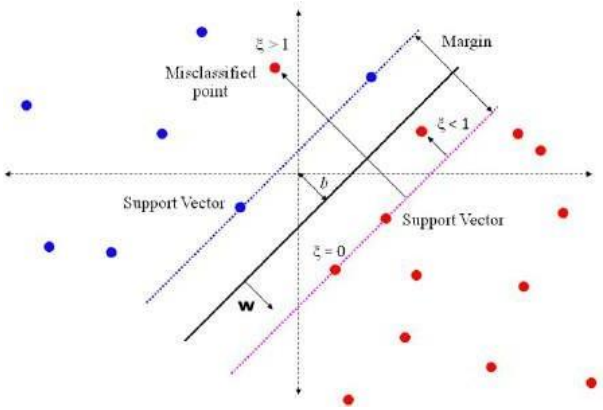


Figure 10.15: Soft margin hyperplanes

$$\text{subject to} \qquad y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1 - \xi_i, \text{ for } i = 1, \ldots N$$
$$\xi_i \geq 0, \text{ for } i = 1, \ldots, N$$

Remarks

1. There are algorithms for solving the reformulated SVM problem given above. The details of these algorithms are beyond the scope of these notes.

2. The hyperplanes given by the equations

$$\vec{w}\cdot\vec{x}_i - b = +1 \quad \text{and} \quad \vec{w}\cdot\vec{x}_i - b = -1$$

with the values of $\vec{w}$ and b obtained as solutions of the reformulated problem, are called the *soft margin hyperplanes* for the SVM problem.

Kernel functions

In the context of SVM's, a kernel function is a function of the form $K(\vec{x}, \vec{y})$, where $\vec{x}$ and $\vec{y}$ are n-dimensional vectors, having a special property. These functions are used to obtain SVM-like classifiers for two-class datasets which are not linearly separable.

Definition

Let $\vec{x}$ and $\vec{y}$ be arbitrary vectors in the n-dimensional vector space $\mathbb{R}^n$. Let φ be a mapping from $\mathbb{R}^n$ to some vector space. A function $K(\vec{x}, \vec{y})$ is called a kernel function if there is a function φ such that $K(\vec{x}, \vec{y}) = \varphi(\vec{x}) \cdot \varphi(\vec{y})$.

Examples

Example 1

Let

$$\vec{x} = (x_1, x_2) \in \mathbb{R}^2$$
$$\vec{y} = (y_1, y_2) \in \mathbb{R}^2$$

We define

$$K(\vec{x}, \vec{y}) = (\vec{x}\cdot\vec{y})^2.$$

We show that this is a kernel function. To do this, we note that

$$\begin{aligned} K(\dot{x}, \dot{y}) &= (\dot{x} \cdot \dot{y})^2 \\ &= (\\ &= x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 \\ & x\ y\ ) \end{aligned}$$

Now we define

$$\varphi(\dot{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \in R^3$$
$$\varphi(\dot{y}) = (y_1^2, \sqrt{2}y_1 y_2, y_2^2) \in R^3$$

Then we have

$$\begin{aligned} \varphi(\dot{x}) \cdot \varphi(\dot{y}) &= x_1^2 y_1^2 + (\sqrt{2}x_1 x_2)(\sqrt{2}y_1 y_2) + x_2^2 y_2^2 \\ &= x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2 \\ &= K(\dot{x}, \dot{y}) \end{aligned}$$

This shows that $K(\dot{x}, \dot{y})$ is indeed a kernel function.

Example 2

Let

$$\dot{x} = (x_1, x_2) \in R^2$$
$$\dot{y} = (y_1, y_2) \in R^2$$

We define $$K(\dot{x}, \dot{y}) = (\dot{x} \cdot \dot{y} + \theta)^2.$$

We show that this is a kernel function. To do this, we note that

$$\begin{aligned} K(\dot{x}, \dot{y}) &= (\dot{x} \cdot \dot{y} + \theta)^2 \\ &= (\\ &= \varphi(\dot{x}) \cdot \varphi(\dot{y}) \\ &+ \theta) \end{aligned}$$

where $$\varphi(x) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2\theta}x_1, \sqrt{2\theta}x_2, \sqrt{\theta}) \in R^6.$$

This shows that $K(\dot{x}, \dot{y})$ is indeed a kernel function.

Some important kernel functions

In the following we assume that $\dot{x} = (x_1, x_2, \ldots, x_n)$ and $\dot{y} = (y_1, y_2, \ldots, y_n)$.

1. **Homogeneous polynomial kernel**

   $$K(\dot{x}, \dot{y}) = (\dot{x} \cdot \dot{y})^d$$

   where d is some positive integer.

2. **Non-homogeneous polynomial kernel**

   $$K(\dot{x}, \dot{y}) = (\dot{x} \cdot \dot{y} + \theta)^d$$

   where d is some positive integer and θ is a real constant.

3. **Radial basis function (RBF) kernel**

$$K(\dot{x}, \dot{y}) = e^{-\|\dot{x}-\dot{y}\|^2/2\sigma^2}$$

This is also called the Gaussian radial function kernel.[1]

4. **Laplacian kernel function**

$$K(\dot{x}, \dot{y}) = e^{-\|\dot{x}-\dot{y}\|/\sigma}$$

5. **Hyperbolic tangent kernel function (Sigmoid kernel function)**

$$K(\dot{x}, \dot{y}) = \tanh(a(\dot{x} \cdot \dot{y}) + c)$$

The kernel method (kernel trick)

Outline

1. Choose an appropriate kernel function $K(\dot{x}, \dot{y})$.
2. Formulate and solve the optimization problem obtained by replacing each inner product $\dot{x} \cdot \dot{y}$ by $K(\dot{x}, \dot{y})$ in the SVM optimization problem.
3. In the formulation of the classifier function for the SVM problem using the inner products of unclassified data $\dot{z}$ and input vectors $\dot{x}_i$, replace each inner product $\dot{z} \cdot \dot{x}_i$ with $K(\dot{z}, \dot{x}_i)$ to obtain the new classifier function.

Algorithm

Algorithm of the kernel method

Given a two-class linearly separable dataset of N points of the form

$$(\dot{x}_1, y_1), (\dot{x}_2, y_2), \ldots, (\dot{x}_N, y_N),$$

where the y_i's are either +1 or 1 and appropriate kernel function $K(\dot{x}, \dot{y})$:

Step 1. Find $\dot{a} = (a_1, a_2, \ldots, a_N)$ which maximizes

$$\sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1, j=1}^{N} a_i a_j y_i y_j K(\dot{x}_i, \dot{x}_j)$$

subject to

$$\sum_{i=1}^{N} a_i y_i = 0$$

$$a_i > 0 \text{ for } i = 1, 2, \ldots, N.$$

Step 2. Compute $\dot{w} = \sum_{i=1}^{N} a_i y_i \dot{x}_i$.

Step 3. Compute $b = \frac{1}{2}(\min_{i\, y_i = +1} K(\dot{w}, \dot{x}_i) + \max_{i\, y_i = -1} K(\dot{w}, \dot{x}_i))$.

Step 4. The SVM classifier function is given by $f(\dot{z}) = \sum_{i=1}^{N} a_i y_i K(\dot{x}_i, \dot{z}) + b$.

---

[1] To represent this kernel as an inner product, we need map φ from $\mathbb{R}^n$ into an infinite-dimensional vector space. A discussion of these ideas is beyond the scope of these notes.

Multiclass SVM's

In machine learning, the *multiclass classification* is the problem of classifying instances into one of three or more classes. Classifying instances into one of the two classes is called binary classification.

Support vector machines can be constructed only when the dataset has only two class-labels and is linearly separable. We have already discussed a method to extend the concept of SVM's to the case where the dataset is not linearly separable. In this section we consider how the SVM's can be used to obtain classifiers when there are more than two class labels. Two methods are generally used to handle such cases known by the names "One-against-all" and "one-against-one".

"One-against-all" method

The *One-Against-All* (OAA) SVMs were first introduced by Vladimir Vapnik in 1995.

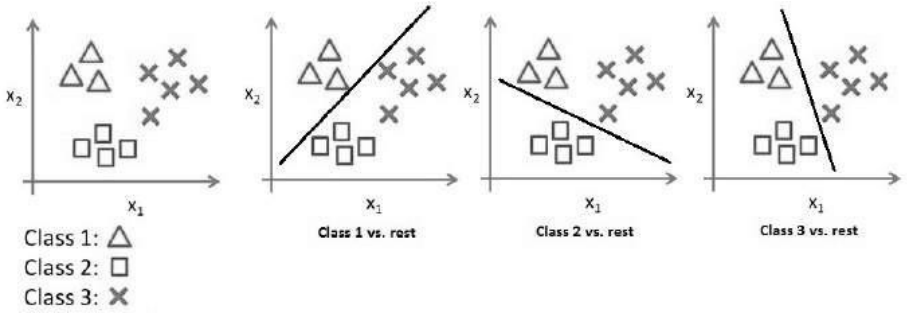


Figure 10.16: One-against all

Let there be p class labels, say, $c_1, c_2, \ldots, c_p$. We construct the following p two-class datasets and obtain the corresponding SVM classifiers. First, we assign the class labels $+1$ to all instances having class label c_1 and the class label -1 to all the remaining instances in the data set. Let $f_1(x)$ be the SVM classifier function for the resulting two-class dataset. Next, we assign the class labels $+1$ to all instances having class label c_2 and the class label -1 to all the remaining instances in the data set. Let $f_2(x)$ be the SVM classifier function for the resulting two-class dataset. We continue like this and generate SVM classifier functions $f_3(x), \ldots, f_p(x)$

Two criteria have been developed to assign a class label to a test instance z.

1. A data point z would be classified under a certain class if and only if that class's SVM accepted it and all other classes' SVMs rejected it. Thus z will be assigned c_i if $f_i(z) > 0$ and $f_j(z) < 0$ for all $j \neq i$.

2. z is the assigned the class label c_i if $f_i(z)$ has the highest value among $f_1(z), \ldots, f_p(z)$ regardless of sign.

Figure 10.16 illustrates the one-against-all method with three classes.

"One-against-one" method

In the *one-against-one* (OAO) (also called *one-vs-one* (OVO)) strategy, a SVM classifier is constructed for each pair of classes. If there are p different class labels, a total of $p(p-1)/2$ classifiers are constructed. An unknown instance is classified with the class getting the most votes. Ties are broken arbitrarily.

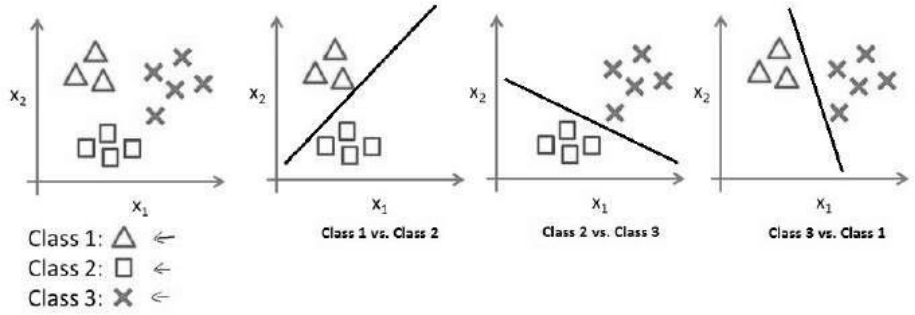


Figure 10.17: One-against-one

For example, let there be three classes, A, B and C. In the OVO method we construct $3 3 (-1)/2 = 3$ SVM binary classifiers. Now, if $\dot{z}$ is to be classified, we apply each of the three classifers to $\dot{z}$. Let the three classifiers assign the classes A, B, B respectively to z. Since a label to z is One-assigned by the majority voting, in this example, we assign the class label of B to z.

vs-one (OVO) strategy is not a particular feature of SVM. Indeed, OVO can be applied to any binary classifier to solve multi-class classification problem.

Sample questions

(a) Short answer questions

1. Define an hyperplane in an n-dimensional space. What are the hyperplanes in 2-dimensional and 3-dimensional spaces?
2. Find the distance of the point $(1, -2, 3)$ from the hyperplane
$$3x_1 - 4x_2 + 12x_3 - 1 = 0.$$
3. What is a linearly separable dataset? Give an example. Give an example for a dataset which is not linearly separable.
4. What is meant by maximum margin hyperplane?
5. Define the support vector machine of a two-class dataset.
6. Define the support vectors of a two-class dataset.
7. What is a kernel function? Give an example.

(b) Long answer questions

1. State the mathematical formulation of the SVM problem. Give an outline of the method for solving the problem.
2. Explain the significance of soft margin hyperplanes and explain how they are computed.
3. Show that the function
$$K(\dot{x}, \dot{y}) = (\dot{x} \cdot \dot{y})^3$$
is a kernel function.
4. What is meant by kernel trick in context of support vector machines? How is it used to find a SVM classifier.

5. Given the following dataset, using elementary geometry find the maximum margin hyperplane for the data. Verify the result by finding the same using the SVM algorithm.

Example	x_1	x_2	Class label
1	2	1	1
2	4	5	1
3	3	6	+1

Hidden Markov models

This chapter contains a brief introduction to hidden Markov models (HMM's). The HMM is one of the most important machine learning models in speech and language processing. To define it properly, we need to first understand the concept of discrete Markov processes. So, we begin the chapter with a description of Markov processes and then discuss HMM's. The three basic problems associated with a HMM are stated, but algorithms for their solutions are not given as they are beyond the scope of these notes.

Discrete Markov processes: Examples

Example 1

Through this example we introduce the various elements that constitute a discrete homogeneous Markov process.

1. **System and states**

   Let us consider a highly simplified model of the different states a stock-market is in, in a given week. We assume that there are only three possible states:

   S_1 : Bull market trend
   S_2 : Bear market trend
   S_3 : Stagnant market trend

2. **Transition probabilities**

   Week after week, the stock-market moves from one state to another state. From previous data, it has been estimated that there are certain probabilities associated with these movements. These probabilities are called transition probabilities.

3. **Markov assumption**

   We assume that the following statement (called Markov assumption or Markov property) regarding transition probabilities is true:

   - Let the weeks be counted as $1, 2, \ldots$ and let an arbitrary week be the t-th week. Then, the state in week $t+1$ depends only on the state in week t, regardless of the states in the previous weeks. This corresponds to saying that, given the present state, the future is independent of the past.

4. **Homogeneity assumption**

   To simplify the computations, we assume that the following property, called the homogeneity assumption, is also true.

- The probability that the stock market is in a particular state in a particular week $t+1$ given that it is in a particular state in week t, is independent of t.

5. **Representation of transition probabilities** Let the probability that a bull week is followed by another bull week be 90%, a bear week be 7.5%, and a stagnant week be 2.5%. Similarly, let the probability that a bear week is followed by another bull week be 15%, bear week be 80% and a stagnant week be 5%. Finally, let the probability that a stagnant week be followed by a bull week is 25%, a bear week be 25% and a stagnant week be 50%. The transition probabilities can be represented in two ways:

   (a) The states and the state transition probabilities can be represented diagrammatically as in Figure 11.1.

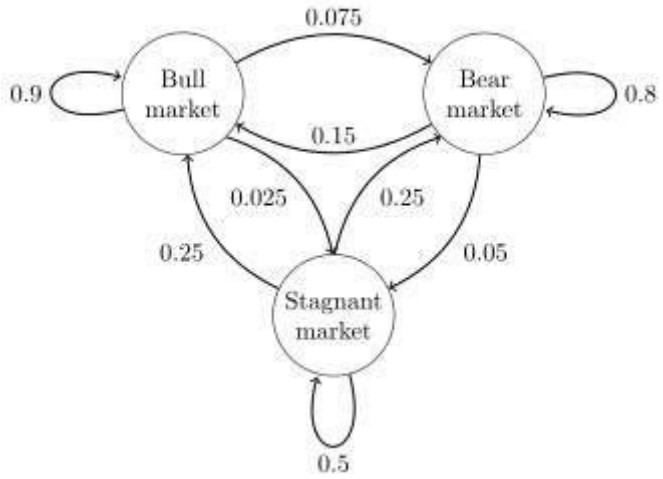


Figure 11.1: A state diagram showing state transition probabilities

   (b) The state transition probabilities can also be represented by a matrix called the *state transition matrix*. Let us label the states as "1 = bull", "2 = bear" and "3 = stagnant" and consider the matrix

$$P = \begin{pmatrix} 0.90 & 0.075 & 0.025 \\ 0.15 & 0.80 & 0.05 \\ 0.25 & 0.50 & 0.25 \end{pmatrix}$$

   In this matrix, the element in the i-th row, j-th column represents the probability that the market in state i is followed by market in state j.

   Note that in the state transition matrix P, the sum of the elements in every row is 1.

6. **Initial probabilities**

   The initial probabilities are the probabilities that the stock-market is in a particular state initially. These are denoted by π_1, π_2, π_3: π_1 is the probability that the stock-market is in bull state initially; similarly, π_2 and π_3. the values of these probabilities can be presented as a vector:

$$\Pi = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.3 \\ 0.2 \end{pmatrix}$$

7. **The discrete Markov process**

   The functioning of the stock-markets with the three states S_1, S_2, S_3 with the assumption that the Markov property is true, the transition probabilities given by the matrix P and the initial

probabilities given by the vector Π constitute a discrete Markov process. Since we also assume the homogeneity property for the transition probabilities is true, it is a homogeneous discrete Markov process.

Probabilities for future states

Consider the matrix:

$$\Pi^T P = [0.5 \quad 0.3 \quad 0.2] \begin{bmatrix} 0.90 & 0.075 & 0.025 \\ 0.15 & 0.80 & 0.05 \\ 0.25 & 0.50 & 0.25 \end{bmatrix}$$
$$= [0.5450 \; 0.3775 \; 0.0775]$$

The elements in this row vector represent the probabilities that the stock-market is in the bull state, the bear state and the stagnant state respectively in the second week.

In general, the elements of the row vector $\Pi^T P^n$ represent the probabilities that the stock-market is in the bull state, the bear state and the stagnant state respectively in the $(n+1)$-th week.

Example 2

Consider a simplified model of weather. We assume that the weather conditions are observed once a day at noon and it is recorded as in one of the following states:

S_1 : Rainy
S_2 : Cloudy
S_3 : Sunny

Assuming that the Markov property and the homogeneity property are true, we can write the state transition probability matrix P. Let the matrix be

$$P = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Let the initial probabilities be

$$\Pi = [0.25 \; 0.25 \; 0.50]$$

The changes in weather with the three sates S_1, S_2, S_3 satisfying the Markov property and the homogeneity property, the transition probability matrix P and the initial probabilities given by Π constitute a discrete homogeneous Markov process.

Discrete Markov processes: General case

A Markov process is a random process indexed by time, and with the property that the future is independent of the past, given the present. The time space may be discrete taking the values $1, 2, \ldots$ or continuous taking any nonnegative real number as a value. In these notes, we consider only discrete time Markov processes.

1. **System and states**

   Consider a system that at any time is in one of N distinct states:

   $$S_1, S_2, \ldots, S_N$$

   We denote the state at time t by q_t for $t = 1, 2, \ldots$. So, $q_t = S_i$ means that the system is in state S_i at time t.

2. Transition probabilities

At regularly spaced discrete times, the system moves to a new state with a given probability, depending on the values of the previous states. These probabilities are called the transition probabilities.

3. Markov assumptions (Markov property)

We assume the following called the Markov assumption or the Markov property:

- The state at time $t+1$ depends only on state at time t, regardless of the states in the previous times. This corresponds to saying that, given the present state, the future is independent of the past.

4. Homogeneity property

We assume that the following property, called the homogeneity property, is true.

- We also assume that these transition probabilities are independent of time, that is, the probabilities $P(q_{t+1} = S_j \mid q_t = S_i)$ are constants and do not depend on t. We denote this probablity by a_{ij}:

$$a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i).$$

We immediately note that

$$a_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} a_{ij} = 1 \text{ for all } i.$$

5. Representation of transition probabilities

The transition probabilities can be represented in two ways:

(a) If the number of states is small, the state transition probabilities can be represented diagrammatically as in Figure 11.1.

(b) The state transition probabilities can also be represented by a matrix called the *state transition matrix*.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & & & \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix}$$

In this matrix, the element in the i-th row, j-th column represents the probability that the system in state S_i moves to state S_j. Note that in the state transition matrix A, the sum of the elements in every row is 1.

6. Initial probabilities

We define the initial probabilities π_i which is the probability that the first state in the sequence is S_i:

$$\pi = P(q_1 = S_i).$$

We also write

$$\Pi = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \dots \\ \pi_N \end{pmatrix}$$

We must have

$$\sum_{i=1}^{N} \pi_i = 1.$$

7. Discrete Markov process

A system with the states $S_1, S_2, \ldots, S_N$ satisfying the Markov property is called a discrete Markov process. If it satisfies the homogeneity property, then it is called a homogeneous discrete Markov process.

Probability for an observation sequence

Observable Markov model

The discrete Markov process described in Section 11.2 is also called an *observable Markov model* or *observable discrete Markov process*. It is so called because the state of the system at any time t can be directly observed. This is in contrast to models where the state of the system cannot be directly observed. If the state of the system cannot be directly observed the system is called a *hidden Markov model*. Such systems are considered in Section **??**.

Probability for an observation sequence

In an observable Markov model, the states are observable. At any time t we know q_t, and as the system moves from one state to another, we get an observation sequence that is a sequence of states. The output of the process is the set of states at each instant of time where each state corresponds to a physical observable event.

Let O be an arbitrary observation sequence of length T. Let us consider a particular observation sequence

$$Q = (q_1, q_2, \ldots, q_T) .$$

Now, given the transition matrix A and the initial probabilities Π we can calculate the probability $P(O = Q)$ as follows.

$$\begin{aligned} P(O = Q) &= P(q_1) P(q_2 | q_1) P(q_3 | q_2) \ldots P(q_T | q_{T-1}) \\ &= \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \end{aligned}$$

Here, π_{q_1} is the probability that the first state is q_1, $a_{q_1 q_2}$ is the probability of going from q_1 to q_2, and so on. We multiply these probabilities to get the probability of the whole sequence.

Example

Consider the discrete Markov process described in Section 11.1.1. Let us compute the probability of having a bull week followed by a stagnant week followed by two bear weeks. In this case the observation sequence is

$$\begin{aligned} Q &= (\text{bull}, \text{stagnant}, \text{bear}, \text{bear}) \\ &= (S_1, S_2, S_3, S_3) \end{aligned}$$

The required probability is

$$\begin{aligned} P(O = Q) &= P(S_1) P(S_2 | S_1) P(S_3 | S_2) P(S_3 | S_3) \\ &= \pi_1 a_{12} a_{23} a_{33} \\ &= 0.5 \times 0.075 \times 0.05 \times 0.25 \\ &= 0.00046875 \end{aligned}$$

Learning the parameters

Consider a homogeneous discrete Markov process with transition matrix A and initial probability vector Π. A and Π are the parameters of the process. The following procedure may be applied to learn these parameters.

Step 1.Obtain K observation sequences each of length T. Let q_{tk} be the observed state at time t in the k-th observation sequence.

Step 2.Let $\hat{\pi}_i$ be the estimate of the initial probability π_i. Then

$$\hat{\pi}_i = \frac{\text{number of sequences starting with } S_i}{\text{total number of sequences}}.$$

Step 3.Let $\hat{a}_{ij}$ be the estimate of a_{ij}. Then

$$\hat{a}_{ij} = \frac{\text{number of transitions from } S_i \text{ to } S_j}{\text{number of transitions from } S_i}$$

Example

Let there be a discrete Markov process with three states S_1, S_2 and S_3. Suppose we have the following 10 observation sequences each of length 5:

$$\begin{aligned}
O_1 &: S_1\ S_2\ S_1\ S_1\ S_1\\
O_2 &: S_2\ S_3\ S_3\ S_2\ S_2\\
O_3 &: S_3\ S_1\ S_3\ S_2\ S_2\\
O_4 &: S_2\ S_3\ S_3\ S_2\ S_2\\
O_5 &: S_3\ S_2\ S_1\ S_1\ S_3\\
O_6 &: S_2\ S_3\ S_3\ S_2\ S_2\\
O_7 &: S_1\ S_1\ S_2\ S_3\ S_2\\
O_8 &: S_2\ S_3\ S_3\ S_2\ S_2\\
O_9 &: S_2\ S_3\ S_1\ S_2\ S_3\\
O_{10} &: S_1\ S_2\ S_2\ S_1\ S_1
\end{aligned}$$

We have:

$$\hat{\pi}_1 = \frac{\text{number of sequences starting with } S_1}{\text{total number of sequences}} = \frac{4}{10}$$

$$\hat{\pi}_2 = \frac{\text{number of sequences starting with } S_2}{\text{total number of sequences}} = \frac{2}{10}$$

$$\hat{\pi}_3 = \frac{\text{number of sequences starting with } S_3}{\text{total number of sequences}} = \frac{4}{10}$$

Therefor

$$\Pi = \begin{bmatrix} 4/10 \\ 2/10 \\ 4/10 \end{bmatrix}$$

We illustrate the computation of a_{ij}'s with an example.

$$\hat{a}_{21} = \frac{\text{number of transitions from } S_2 \text{ to } S_1}{\text{number of transitions from } S_2} = \frac{6}{11}$$

$$\hat{a}_{22} = \frac{\text{number of transitions from } S_2 \text{ to } S_2}{\text{number of transitions from } S_2} = \frac{3}{11}$$

$$\hat{a}_{23} = \frac{\text{number of transitions from } S_2 \text{ to } S_3}{\text{number of transitions from } S_2} = \frac{2}{11}$$

The remaining transition probabilities can be estimated in a similar way.

$$\hat{A} = \begin{bmatrix} 9/19 & 6/19 & 4/19 \\ 6/11 & 3/11 & 2/11 \\ 5/10 & 4/10 & 1/10 \end{bmatrix}.$$

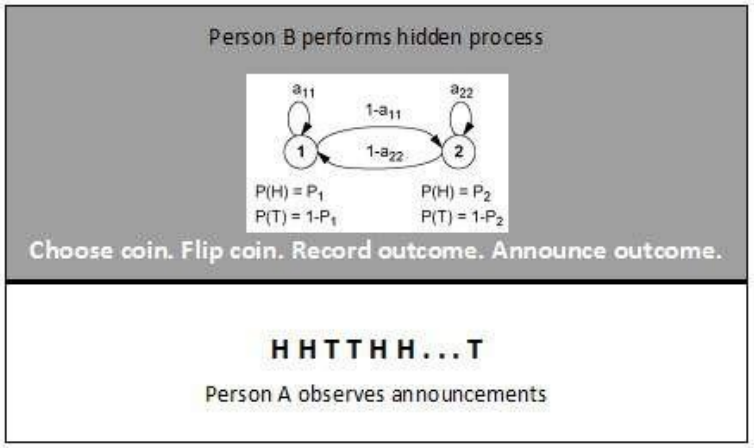


Figure 11.2: A two-coin model of an HMM

Hidden Markov models

Coin tossing example

Let us consider the following scenario:

Consider a room which is divided into two parts by a curtain through which we cannot see what is happening on the other half of the room. Person A is sitting in one half and person B is sitting in the other half. Person B is doing some coin tossing experiment, but she will not tell person A anything about what she is doing. Person B will only announce the result of each coin flip. Let a typical sequence of announcements be

$$O = O_1 O_2 \ldots O_T = H H T H H T T T \ldots H \quad \text{(say)}$$

where as usual H stands for heads and T stands for tails. Person A wants to create a mathematical model which explains this sequence of observation. Person A suspects that person B is announcing the results based on the outcomes of some discrete Markov process. If that is true, then the Markov process that is happening behind the curtain is hidden from the rest of the world and we are left with a *hidden Markov process*. To verify whether actually a Markov process is happening is a daunting task. Based on the observations like O alone, we have to decide on the following:

- A Markov process has different states. What should the states in the process correspond to what is happening behind the curtain?
- How many states should be there?
- What should be the initial probabilities?
- What should be the transition probabilities?

Let us assume that person B is doing something like the following before announcing the outcomes.

1. Let person B be in possession of two biased coins (or, three coins, or any number of coins) and she is flipping these coins in some order. When flipping a particular coin, the system is in the state of that coin. So, each of these coins may be identified as a state and there are two states, say S_1 and S_2.
2. The outcomes of the flips of the coins are the observations. These observations are represented by the observation symbols “H” (for “head”) and “T” (for “tail”).

3. After flipping coin, one of the two coins should be flipped next. There must be some definite procedure for doing this. The procedure is some random process with definite probabilities for selecting the coins. These are the transition probabilities and they define the transition probability matrix A.

4. Since the coins are biased, there would be definite probabilities for getting "H" or "T" each time the coin is flipped. These probabilities are called the observation probabilities.

5. There must be some procedure for selecting the first coin. This is specified by the initial probabilities vector Π.

The urn and ball model

Again, consider a room which is divided into two parts by a curtain through which we cannot see what is happening on the other half of the room. Person A is sitting in one half and person B is sitting in the other half. Person B is doing some experiment, but she will not tell person A anything about what she is doing. Person B will only announce the result of each experiment. Let a typical sequence of announcements be

$$O = O_1 O_2 \ldots O_T$$
$$= \text{"red", "green", "red", . . . , "blue"}$$

Person A wants to create a mathematical model which explains this sequence of observations.

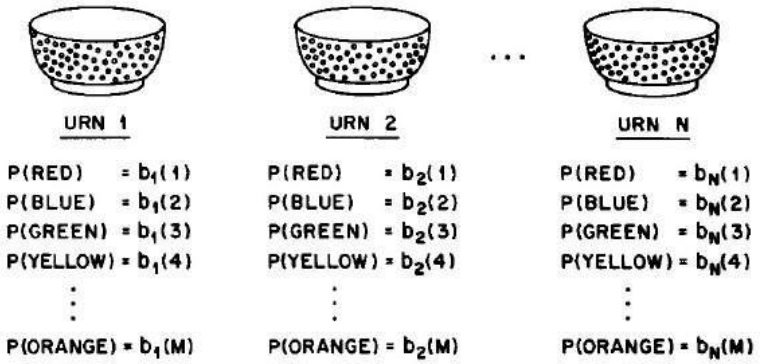


Figure 11.3: An N-state urn and ball model which illustrates the general case of a discrete symbol HMM

Person A suspects that person B is announcing the results based on the outcomes of some discrete Markov process. If that is true, then the Markov process that is happening behind the curtain is hidden from the rest of the world and we are left with a *hidden Markov process*.

In this example, let us assume that person A suspects that something like the following is happening behind the curtain.

There are N large urns behind the curtain. Within each urn there are large number of coloured balls. There are M distinct colours of balls. Person B, according to some random process, chooses an initial urn. From this urn a ball is chosen at random and the colour of the ball is announced. The ball is then replaced in the urn. A new urn is then selected according to some random selection process associated with the current urn and the ball selection process is repeated.

This process is a typical example of a hidden Markov process. Note the following:

1. Selection of an urn may be made to correspond to a state of the process. Then, there are N states in the process.

2. The colours of the balls selected are the observations. The name of the colour may be referred to as the "observation symbol". Hence, there are M observation symbols in the process.

3. The random selection process associated with the current urn specifies the transition probabilities.

4. Each urn contains a mixture of balls of different colours. So, corresponding to each urn, there are definite probabilities for getting balls of different colours. These probabilities are called the observation probabilities.

5. The procedure for selecting the first urn provides the initial probabilities.

Hidden Markov model (HMM): The general case

A hidden Markov model (HMM) is characterized by the following:

1. The number of states in the model, say N. Let the states be $S_1, S_2, \ldots, S_N$.

2. The number of distinct observation symbols, say M. Let the observation symbols be $v_1, v_2, \ldots, v_M$. (The observation symbols correspond to the physical outputs of the system.)

3. The state transition probabilities specified by an $N \times N$ matrix $A = [a_{ij}]$:

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \text{ for } i, j = 1, 2, \ldots, N.$$

where q_t is the state at time t.

4. The observation symbol probability distributions $b_j(k)$ for $j = 1, \ldots, N$ and $k = 1, \ldots, M$. $b_j(k)$ is the probability that, at time t, the outcome is the symbol v_k given that the system is in state S_j:

$$b_j(k) = P(v_k \text{ at } t | q_t = S_j).$$

We denote by B the $N \times M$ matrix whose element in the j-th row k-column is $b_j(k)$.

5. The initial probabilities $\Pi = [\pi_i]$:

$$\pi = P(q_1 = S_i), \text{ for } i = 1, 2, \ldots, N.$$

The values of N and M are implicitly defined in A, B and Π. So, a HMM is completely defined by the parameter set

$$\lambda = (A, B, \Pi).$$

Three basic problems of HMMs

Given the general model of HMM, there are three basic problems that must be solved for the model to be useful for real-world applications. These problems are the following:

Problem 1. Evaluation problem

Given the observation sequence

$$O = O_1 O_2 \ldots O_T,$$

and a HMM model

$$\lambda = (A, B, \Pi)$$

how do we efficiently compute

$$P(O|\lambda),$$

the probability of the observation sequence O given the model λ?

Problem 2. Finding state sequence problem

Given the observation sequence

$$O = O_1 O_2 \ldots O_T,$$

and a HMM model

$$\lambda = (A, B, \Pi)$$

how do we find the the state sequence

$$Q = q_1 q_2 \ldots, q_T$$

which has the highest probability of generating O; that is, how do we find Q^* that maximizes the probability $P(Q|O, \lambda)$?

Problem 3. Learning model parameters problem

Given a training set X observation sequences, how do we learn the model

$$\lambda = (A, B, \Pi)$$

that maximizes the probability of generating X; that is, how do we find λ^* that maximizes the probability

$$P(X|\lambda).$$

Solutions of the basic problems

The details of the algorithms for solving these problems are beyond the scope of these notes. Problem 1 is solved using the Forwards-Backwards algorithms. Problem 2 is solved by the Viterbi algorithm and posterior decoding. Finally, Problem 3 is solved by the Baum-Welch algorithm.[1]

HMM application: Isolated word recognition

Most speech-recognition systems are classified as isolated or continuous. Isolated word recognition requires a brief pause between each spoken word, whereas continuous speech recognition does not. Speech-recognition systems can be further classified as speaker-dependent or speaker-independent. A speaker-dependent system only recognizes speech from one particular speaker's voice, whereas a speaker-independent system can recognize speech from anybody.

In this section, we consider in an outline form how HMMs are used in building an *isolated word recogniser*.

1. Assume that we have a vocabulary V of words to be recognised.
2. For each word in the vocabulary, there is a training set of K occurrences of each spoken word (spoken by 1 or more talkers) where each occurrence of the word constitute an observation sequence.
3. The observations are some appropriate representation of the characteristics of the word. These representations are obtained via some preprocessing of the speech signal like linear predictive coding (LPC).
4. For each word $v \in V$, we build an HMM, say

   $$\lambda^v = (A^v, B^v, \Pi^v).$$

   For this, we have to apply the algorithms for learning an HMM to estimate the parameters (A^v, B^v, Π^v) that maximise the probability of generating the observations in the training set of K occurrences of the word v.

[1] For a concise presentation of the algorithms, visit `http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html`.

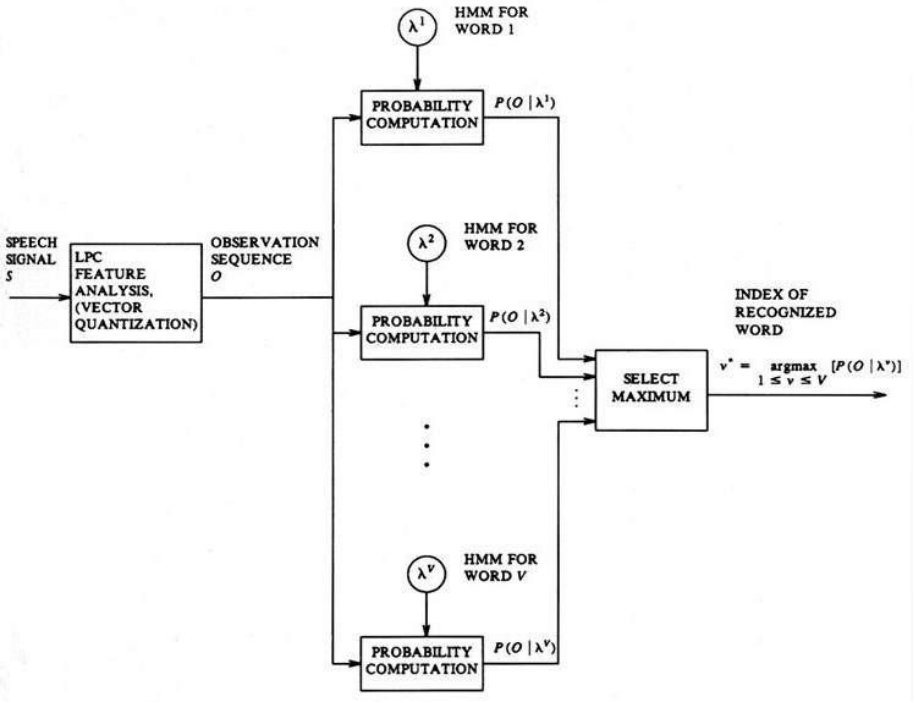


Figure 11.4: Block diagram of an isolated word HMM recogniser

5. Now consider an unknown word v which needs to be recognised. The following procedure is used to recognise the word.

   (a) The speech signal corresponding to the word w is subjected to preprocessing like LPC and converted to the representation used in building the HMMs and the measurement of the observation sequence $O = O_1 O_2 \ldots O_T$ is obtained.

   (b) The probabilities $P(O|\lambda^v)$, for each word $v \in V$ are calculated.

   (c) Choose the word v for which $P(O|\lambda^v)$ is highest:

$$v^* = \arg\max_{v \in V} P(O|\lambda^v).$$

   (d) The word w is recognised as the word v^*.

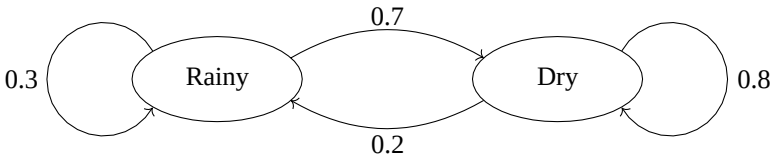
Sample questions

(a) Short answer questions

1. What is the state transition matrix of a discrete Markov process?

2. What is the Markov property of a discrete Markov process?

3. Consider a Markov process with two states "Rainy" and "Dry" and the transition probabilities as shown in the following diagram.



If $P(\text{Rain}) = 0.4$ and $P(\text{Dry}) = 0.6$ compute the probability for the sequence "Rain, Rain, Dry, Dry".

(b) Long answer questions

1. Describe a discrete Markov process with an example.
2. Describe a hidden Markov model.
3. Explain how hidden Markov models are used in speech recognition.
4. What are the basic problems associated with a hidden Markov model.
5. Describe the urn and ball model of a hidden Markov model.
6. Describe the coin tossing model of a hidden Markov model.
7. Let there be a discrete Markov process with two states S_1 and S_2. Given the following sequences of observations of these states, estimate the initial probabilities and the transition probabilities of the process.

$$S_1S_2, \quad S_2S_2, \quad S_1S_2, \quad S_2S_1, \quad S_1S_1, \quad S_2S_1, \quad S_1S_2, \quad S_1S_1.$$

Combining multiple learners

In general there are several algorithms for learning the same task. Though these are generally successful, no one single algorithm is always the most accurate. Now, we shall discuss models composed of multiple learners that complement each other so that by combining them, we attain higher accuracy.

Why combine many learners

There are several reasons why a single learner may not produce accurate results.

- Each learning algorithm carries with it a set of assumptions. This leads to error if the assumptions do not hold. We cannot be fully sure whether the assumptions are true in a particular situation.
- Learning is an ill-posed problem. With finite data, each algorithm may converge to a different solution and may fail in certain circumstances.
- The performance of a learner may be fine-tuned to get the highest possible accuracy on a validation set. But this fine-tuning is a complex task and still there are instances on which even the best learner is not accurate enough.
- It has been proved that there is no single learning algorithm that always produces the most accurate output.

Ways to achieve diversity

When many learning algorithms are combined, the individual algorithms in the collection are called the *base learners* of the collection.

When we generate multiple base-learners, we want them to be reasonably accurate but do not require them to be very accurate individually. The base-learners are not chosen for their accuracy, but for their simplicity. What we care for is the final accuracy when the base- learners are combined, rather than the accuracies of the bas-learners we started from.

There are several different ways for selecting the base learners.

1. **Use different learning algorithms**

   There may be several learning algorithms for performing a given task. For example, for classification, one may choose the naive Bayes' algorithm, or the decision tree algorithm or even the SVM algorithm.

   Different algorithms make different assumptions about the data and lead to different results. When we decide on a single algorithm, we give emphasis to a single method and ignore all others. Combining multiple learners based on multiple algorithms, we get better results.

2. **Use the same algorithm with different hyperparameters**

   In machine learning, a *hyperparameter* is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.

   The number of layers, the number of nodes in each layer and the initial weights are all hyperparameters in an artificial neural network. When we train multiple base-learners with different hyperparameter values, we average over it and reduce variance, and therefore error.

3. **Use different representations of the input object**

   For example, in speech recognition, to recognize the uttered words, words may be represented by the acoustic input. Words can also be represented by video images of the speaker's lips as the words are spoken.

   Different representations make different characteristics explicit allowing better identification. In many applications, there are multiple sources of information, and it is desirable to use all of these data to extract more information and achieve higher accuracy in prediction. We make separate predictions based on different sources using separate base-learners, then combine their predictions.

4. **Use different training sets to train different base-learners**

   - This can be done by drawing random training sets from the given sample; this is called *bagging*.
   - The learners can be trained serially so that instances on which the preceding base-learners are not accurate are given more emphasis in training later base-learners; examples are *boosting* and *cascading*.
   - The partitioning of the training sample can also be done based on locality in the input space so that each base-learner is trained on instances in a certain local part of the input space.

5. **Multiexpert combination methods**

   These base learners work in parallel. All of them are trained and then given an instance, they all give their decisions, and a separate combiner computes the final decision using their predictions. Examples include voting and its variants.

6. **Multistage combination methods**

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

Model combination schemes

Voting

This is the simplest procedure for combining the outcomes of several learning algorithms. Let us examine some special cases of this scheme

1. **Binary classification problem**

   Consider a binary classification problem with class labels -1 and $+1$. Let there be L base learners and let x be a test instance. Each of the base learners will assign a class label to x. If the class label assigned is $+1$, we say that the learner votes for $+1$ and that the label $+1$ gets a vote. The number of votes obtained by the class labels when the different base learners are applied is counted. In the voting scheme for combining the learners, the label which gets the majority votes is assigned to x.

2. **Multi-class classification problem**

   Let there be n class labels $C_1, C_2, , C_n$. Let x be a test instance and let there be L base learners. Here also, each of the base learners will assign a class label to x and when a class label is assigned a label, the label gets a vote. In the voting scheme, the class label which gets the maximum number of votes is assigned to x.

3. **Regression**

   Consider L base learners for predicting the value of a variable y. Let $\hat{y}_i$ be the output predicted by the i-th base learner. The final output is computed as

$$y = w_i\hat{y}_1 + w_2\hat{y}_2 + ...+ w_L\hat{y}_L$$

   where $w_1, w_2, , w_L$ are called the weights attached to the outputs of the various base learners and they must satisfy the following conditions:

$$w_j \geq 0 \text{ for } j = 1, 2, ..., L$$
$$w + w + ... + w = 1$$
$$2 \qquad L$$

   This is the *weighted voting scheme*. In *simple voting*, we take

$$w_i = \frac{1}{L} \text{ for } j = 1, 2, \ldots, L.$$

Bagging

Bagging is a voting method whereby base-learners are made different by training them over slightly different training sets.

Generating L slightly different samples from a given sample is done by bootstrap, where given a training set X of size N, we draw N instances randomly from X with replacement (see Section **??**). Because sampling is done with replacement, it is possible that some instances are drawn more than once and that certain instances are not drawn at all. When this is done to generate L samples X_j, j 1, . . . , L, these samples are similar because they are all drawn from the same original sample, but they are also slightly different due to chance.

The base-learners are trained with these L samples X_j. A learning algorithm is an *unstable algorithm* if small changes in the training set causes a large difference in the generated learner. Bagging, short for bootstrap aggregating, uses bootstrap to generate L training sets, trains L base-learners using an unstable learning procedure and then during testing, takes an average. Bagging can be used both for classification and regression. In the case of regression, to be more robust, one can take the median instead of the average when combining predictions.

Algorithms such as decision trees and multilayer perceptrons are unstable.

Boosting

In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method. In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the previous learners. The original boosting algorithm combines three weak learners to generate a strong learner. A weak learner has error probability less than 1/2, which makes it better than random guessing on a two-class problem, and a strong learner has arbitrarily small error probability.

The boosting method

1. Let d_1, d_2, d_3 be three learning algorithms for a particular task. Let a large training set X be given.

2. We randomly divide X into three sets, say X_1, X_2, X_3.

3. We use X_1 and train d_1.

4. We then take X_2 and feed it to d_1.

5. We take all instances misclassified by d_1 and also as many instances on which d_1 is correct from X_2, and these together form the training set of d_2.

6. We then take X_3 and feed it to d_1 and d_2.

7. The instances on which d_1 and d_2 disagree form the training set of d_3.

8. During testing, given an instance, we give it to d_1 and d_2 if they agree, that is the response; otherwise the response of d_3 is taken as the output.

It has been shown that this overall system has reduced error rate, and the error rate can arbitrarily be reduced by using such systems recursively. One disadvantage of the system is thaaaaaat it requires a very large training sample. An improved algorithm known as AdaBoost (short for "adaptive boosting"), uses the same training set over and over and thus need not be large. AdaBoost can also combine an arbitrary number of base-learners, not three.

Ensemble learning Y

The word "ensemble" literally means "a group of things or people acting or taken together as a whole, especially a group of musicians who regularly play together."

In machine learning, an ensemble learning method consists of the following two steps:

1. Create different models for solving a particular problem using a given data.

2. Combine the models created to produce improved results.

The different models may be chosen in many different ways:

- The models may be created using appropriate different algorithms like k-NN algorithm, Naive-Bayes algorithm, decision tree algorithm, etc.
- The models may be created by using the same algorithm but using different splits of the same dataset into training data and test data.
- The models may be created by assigning different initial values to the parameters in the algorithm as in ANN algorithms.

The models created in the ensemble learning methods are combined in several ways.

- Simple majority voting in classification problems: Every model makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes.
- Weighted majority voting in classification problem: In weighted voting we count the prediction of the better models multiple times. Finding a reasonable set of weights is up to us.
- Simple averaging in prediction problems: In simple averaging method, for every instance of test dataset, the average predictions are calculated.
- Weighted averaging in prediction problems: In this method, the prediction of each model is multiplied by the weight and then their average is calculated.

Random forest Y

A *random forest* is an ensemble learning method where multiple decision trees are constructed and then they are merged to get a more accurate prediction.

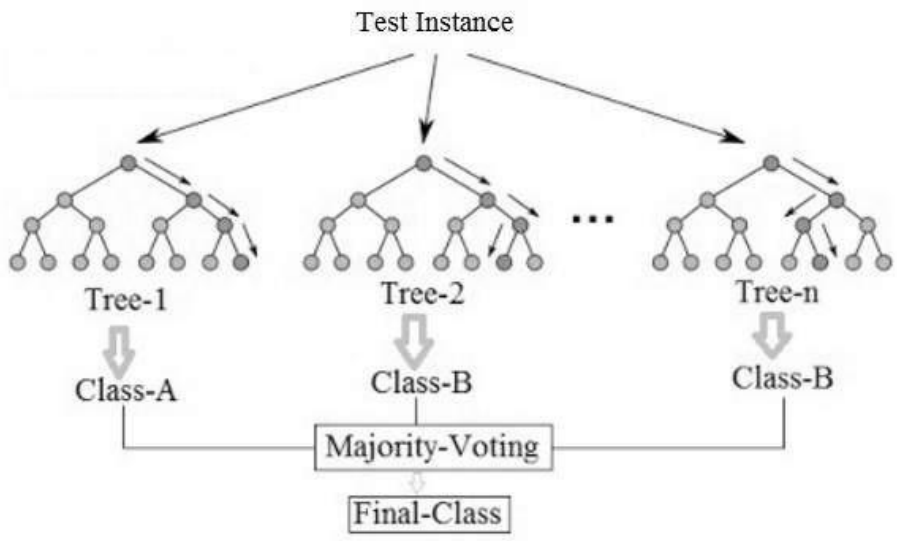


Figure 12.1: Example of random forest with majority voting

Algorithm

Here is an outline of the random forest algorithm.

1. The random forests algorithm generates many classification trees. Each tree is generated as follows:
   (a) If the number of examples in the training set is N , take a sample of N examples at random - but with replacement, from the original data. This sample will be the training set for generating the tree.
   (b) If there are M input variables, a number m is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the generation of the various trees in the forest.
   (c) Each tree is grown to the largest extent possible.
2. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification

Strengths and weaknesses

Strengths

The following are some of the important strengths of random forests.

- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- Generated forests can be saved for future use on other data.

- Prototypes are computed that give information about the relation between the variables and the classification.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.
- Random forest run times are quite fast, and they are able to deal with unbalanced and missing data.
- They can handle binary features, categorical features, numerical features without any need for scaling.
- There are lots of excellent, free, and open-source implementations of the random forest algorithm. We can find a good implementation in almost all major ML libraries and toolkits.

Weaknesses

- A weakness of random forest algorithms is that when used for regression they cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.
- The sizes of the models created by random forests may be very large. It may take hundreds of megabytes of memory and may be slow to evaluate.
- Random forest models are black boxes that are very hard to interpret.

Sample questions

(a) Short answer questions

1. Explain the necessity of combining several algorithms for accomplishing a particular task.
2. What is a base learner? How do we select base learners?

(b) Long answer questions

1. Explain the following: (i) voting (ii) bagging (iii) boosting.
2. Explain what is meant by random forests.

Module 6

Clustering methods

Clustering

Clustering or *cluster analysis* is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Clustering is a main task of exploratory data mining and used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances between cluster members, dense areas of the data space, etc.

Examples of data with natural clusters

In many applications, there will naturally be several groups or clusters in samples.

1. Consider the case of optical character recognition: There are two ways of writing the digit 7; the American writing is '7', whereas the European writing style has a horizontal bar in the middle (something like 7). In such a case, when the sample contains examples from both continents, the sample will contain two clusters or groups one corresponding to the American 7 and the other corresponding to the European 7̶.
2. In speech recognition, where the same word can be uttered in different ways, due to different pronunciation, accent, gender, age, and so forth, there is not a single, universal prototype. In a large sample of utterances of a specific word, All the different ways should be represented in the sample.

k-means clustering

Outline

The k-means clustering algorithm is one of the simplest unsupervised learning algorithms for solving the clustering problem.

Let it be required to classify a given data set into a certain number of clusters, say, k clusters. We start by choosing k points arbitrarily as the "centres" of the clusters, one for each cluster. We then associate each of the given data points with the nearest centre. We now take the averages of the data points associated with a centre and replace the centre with the average, and this is done for each of the centres. We repeat the process until the centres converge to some fixed points. The data points nearest to the centres form the various clusters in the dataset. Each cluster is represented by the associated centre.

Example

We illustrate the algorithm in the case where there are only two variables so that the data points and cluster centres can be geometrically represented by points in a coordinate plane. The distance between the points (x_1, x_2) and (y_1, y_2) will be calculated using the familiar distance formula of elementary analytical geometry:

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

Problem

Use k-means clustering algorithm to divide the following data into two clusters and also compute the the representative data points for the clusters.

x_1	1	2	2	3	4	5
x_2	1	1	3	2	3	5

Table 13.1: Data for k-means algorithm example

Solution

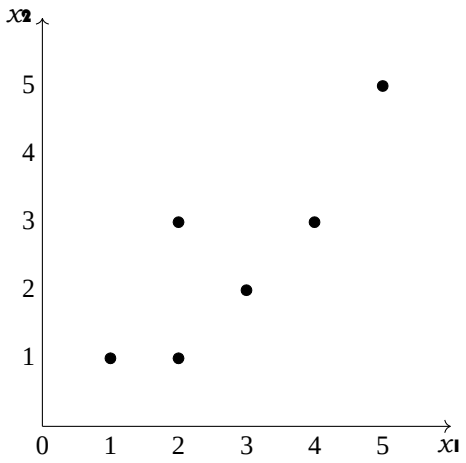


Figure 13.1: Scatter diagram of data in Table 13.1

1. In the problem, the required number of clusters is 2 and we take $k = 2$.

2. We choose two points arbitrarily as the initial cluster centres. Let us choose arbitrarily (see Figure 13.2)

$$v_1 = (2, 1), \quad v_2 = (2, 3).$$

3. We compute the distances of the given data points from the cluster centers.

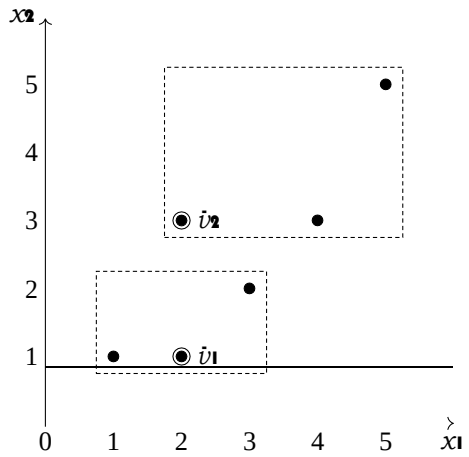


Figure 13.2: Initial choice of cluster centres and the resulting clusters

x_i	Data point	Distance from $v_1 = (2,1)$	Distance from $v_2 = (2,3)$	Minimum distance	Assigned center
x_1	$1,1)$	1	2.24	1	v_1
x_2	$(2,1)$	0	2	0	v_1
x_3	$(2,3)$	2	0	0	v_2
x_4	$(3,2)$	1.41	1.41	0	v_1
x_5	$(4,3)$	2.82	2	2	v_2
x_6	$(5,5)$	5	3.61	3.61	v_2

(The distances of x_4 from v_1 and v_2 are equal. We have assigned v_1 to x_4 arbitrarily.)

This divides the data into two clusters as follows (see Figure 13.2):

Cluster 1: $\{x_1, x_2, x_4$ represented by $v_1\}$

Number of data points in Cluster 1: $c_1 = 3$.

Cluster 2 : $\{x_3, x_5, x_6$ represented by $v_2\}$

Number of data points in Cluster 2: $c_2 = 3$.

4. The cluster centres are recalculated as follows:

$$v_1 = \frac{1}{c}(x_1 + x_2 + x_4)$$
$$= \frac{1}{3}(x_1 + x_2 + x_4)$$
$$= (2.00, 1.33)$$

$$v_2 = \frac{1}{c}(x_3 + x_5 + x_6)$$
$$= \frac{1}{3}(x_3 + x_5 + x_6)$$
$$= (3.67, 3.67)$$

5. We compute the distances of the given data points from the new cluster centers.

x_i	Data point	Distance from $v_1 = (2,1)$	Distance from $v_2 = (2,3)$	Minimum distance	Assigned center
x_1	$(1,1)$	1.05	3.77	1.05	v_1
x_2	$(2,1)$	0.33	3.14	0.33	v_1
x_3	$(2,3)$	1.67	1.80	1.67	v_1
x_4	$(3,2)$	1.20	1.80	1.20	v_1
x_5	$(4,3)$	2.60	0.75	0.75	v_2
x_6	$(5,5)$	4.74	1.89	1.89	v_2

This divides the data into two clusters as follows (see Figure 13.4):

Cluster 1 : $\{x_1, x_2, x_3, x_4$ represented by $v_1\}$

Number of data points in Cluster 1: $c_1 = 4$.

Cluster 2 : $\{x_5, x_6$ represented by $v_2\}$

Number of data points in Cluster 1: $c_2 = 2$.

6. The cluster centres are recalculated as follows:

$$v_1 = \frac{1}{c_1}(x_1 + x_2 + x_3 + x_4)$$
$$= \frac{1}{4}(x_1 + x_2 + x_3 + x_4)$$
$$= (2.00, 1.33)$$

$$v_2 = \frac{1}{2}(x_5 + x_6) = (3.67, 3.67)$$

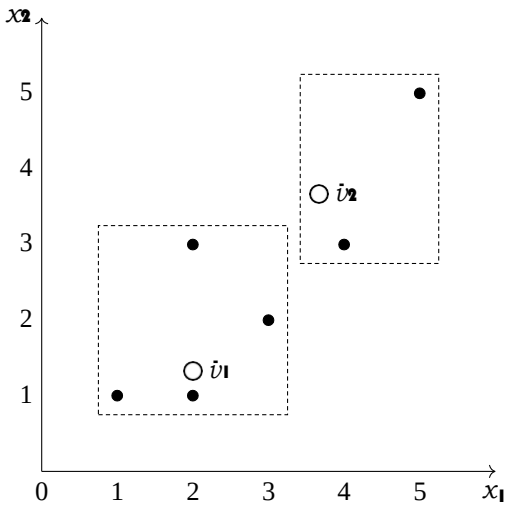


Figure 13.3: Cluster centres after first iteration and the corresponding clusters

7. We compute the distances of the given data points from the new cluster centers.

4.609772 3.905125 2.692582 2.500000 1.118034 1.118034

x_i	Data point	Distance from $v_1 = (2,1)$	Distance from $v_2 = (2,3)$	Minimum distance	Assigned center
x_1	$(1,1)$	1.25	4.61	1.25	v_1
x_2	$(2,1)$	0.75	3.91	0.75	v_1
x_3	$(2,3)$	1.25	2.69	1.25	v_1
x_4	$(3,2)$	1.03	2.50	1.03	v_1
x_5	$(4,3)$	2.36	1.12	1.12	v_2
x_6	$(5,5)$	4.42	1.12	1.12	v_2

This divides the data into two clusters as follows (see Figure **??**):

Cluster 1 : $\{x_1, x_2, x_3, x_4\}$ represented by v_1

Number of data points in Cluster 1: $c_1 = 4$.

Cluster 2 : $\{x_5, x_6\}$ represented by v_2

Number of data points in Cluster 1: $c_1 = 2$.

8. The cluster centres are recalculated as follows:

$$v_1 = \frac{1}{c_1}(x_1 + x_2 + x_3 + x_4)$$
$$= \frac{1}{4}(x_1 + x_2 + x_3 + x_4)$$
$$= (2.00, 1.75)$$

$$v_2 = \frac{1}{c_2}(x_5 + x_6)$$
$$= \frac{1}{2}(x_5 + x_6)$$
$$= (4.00, 4.50)$$

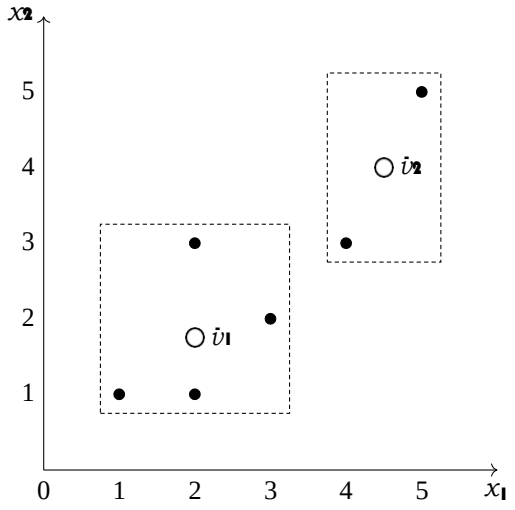


Figure 13.4: New cluster centres and the corresponding clusters

9. This divides the data into two clusters as follows (see Figure **??**):

Cluster 1 : $\{x_1, x_2, x_3, x_4\}$ represented by v_1

Cluster 2 : $\{x_5, x_6\}$ represented by v_2

10. The cluster centres are recalculated as follows:

$$\vec{v}_1 = \frac{1}{4}(\vec{x}_1 + \vec{x}_2 + \vec{x}_3 + \vec{x}_4) = (2.00, 1.75)$$
$$\vec{v}_2 = \frac{1}{2}(\vec{x}_5 + \vec{x}_6) = (4.00, 4.50)$$

We note that these are identical to the cluster centres calculated in Step 8. So there will be no reassignment of data points to different clusters and hence the computations are stopped here.

11. Conclusion: The k means clustering algorithm with $k = 2$ applied to the dataset in Table 13.1 yields the following clusters and the associated cluster centres:

Cluster 1 : $\{\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4$ represented by $\vec{y}_1 = (2.00, 1.75)$

Cluster 2 : $\{\vec{x}_5, \vec{x}_6$ represented by $\vec{y}_2 = (4.00, 4.75)$

The algorithm

Notations

We assume that each data point is a n-dimensional vector:

$$\vec{x} = (x_1, x_2, \ldots, x_n).$$

The distance between two data points

$$\vec{x} = (x_1, x_2, \ldots, x_n)$$

and

$$\vec{y} = (y_1, y_2, \ldots, x_n)$$

is defined as

$$\|\vec{x} - \vec{y}\| = \sqrt{(x_1 - y_1)^2 + \ldots (x_n - y_n)^2}$$

Let $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$ be the set of data points, $V = \{\vec{v}_1, \ldots, \vec{v}_k\}$ be the set of centres and c_i for $i = 1, \ldots, k$ be the number of data points in the i-th cluster

Basic idea

What the algorithm aims to achieve is to find a partition the set X into k mutually disjoint subsets $S = \{S_1, S_2, \ldots, S_k\}$ and a set of data points V which minimizes the following within-cluster sum of errors:

$$\sum_{i=1}^{k} \sum_{\vec{x} \in S_i} \|\vec{x} - \vec{v}_i\|^2$$

Algorithm

Step 1.Randomly select k cluster centers $\vec{v}_1, \ldots, \vec{v}_k$.

Step 2.Calculate the distance between each data point $\vec{x}_i$ and each cluster center $\vec{v}_j$.

Step 3.For each $j = 1, 2, \ldots, N$, assign the data point $\vec{x}_j$ to the cluster center $\vec{v}_i$ for which the distance $\|\vec{x} - \vec{v}\|$ is minimum. Let $\vec{x}_{i1}, \vec{x}_{i2}, \ldots, \vec{x}_{ic_i}$ be the data points assigned to $\vec{v}_i$

Step 4.Recalculate the cluster centres using

$$\vec{v}_i = \frac{1}{c_i}(\vec{x}_{i1} + \ldots + \vec{x}_{ic_i}), \quad i = 1, 2, \ldots, k.$$

Step 5.Recalculate the distance between each data point and newly obtained cluster centers.

Step 6.If no data point was reassigned then stop. Otherwise repeat from Step 3.

Some methods for initialisation

The following are some of the methods for choosing the initial v_i's.

- Randomly take some k data points as the initial v_i's.
- Calculate the mean of all data and add small random vectors to the mean to get the k initial v_i's.
- Calculate the principal component, divide its range into k equal intervals, partition the data into k groups, and then take the means of these groups as the initial centres.

Disadvantages

Even though the k-means algorithm is fast, robust and easy to understand, there are several disadvantages to the algorithm.

- The learning algorithm requires apriori specification of the number of cluster centers.
- The final cluster centres depend on the initial v_i's.
- With different representation of data we get different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).
- Euclidean distance measures can unequally weight underlying factors.
- The learning algorithm provides the local optima of the squared error function.
- Randomly choosing of the initial cluster centres may not lead to a fruitful result.
- The algorithm cannot be applied to categorical data.

Application: Image segmentation and compression

Image segmentation

The goal of segmentation is to partition an image into regions each of which has a reasonably homogeneous visual appearance or which corresponds to objects or parts of objects. Each pixel in an image is a point in a 3-dimensional space comprising the intensities of the red, blue, and green channels. A segmentation algorithm simply treats each pixel in the image as a separate data point. For any value of k, each pixel is replaced by the pixel vector with the (R, G, B) intensity triplet given by the centre μ_k to which that pixel has been assigned. For a given value of k, the algorithm is representing the image using a palette of only k colours. It should be emphasized that this use of k-means is a very crude approach to image segmentation. The image segmentation problem is in general extremely difficult.

Data compression

We can also the clustering algorithm to perform data compression. There are two types of data compression: *lossless data compression*, in which the goal is to be able to reconstruct the original data exactly from the compressed representation, and *lossy data compression*, in which we accept some errors in the reconstruction in return for higher levels of compression than can be achieved in the lossless case.

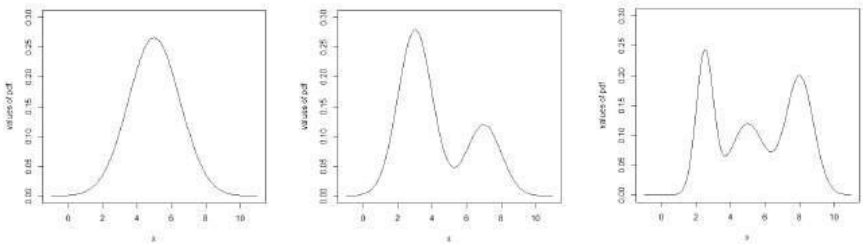
We can apply the k-means algorithm to the problem of lossy data compression as follows. For each of the N data points, we store only the identity of the cluster to which it is assigned. We also store the values of the k cluster centres μ_k, which requires much less data, provided we choose k much smaller than N. Each data point is then approximated by its nearest centre μ_k. New data points can similarly be compressed by first finding the nearest μ_k and then storing the label k instead of the original data vector. This framework is often called *vector quantization*, and the vectors Îijμ_k are called *code-book vectors*.

Multi-modal distributions

Definitions

1. In statistics, a *unimodal distribution* is a continuous probability distribution with only one mode (or "peak").

   A random variable having the normal distribution is a unimodal distribution. Similarly, the *t*-distribution and the chi-squared distribution are also unimodal distributions.



Unimodal Bimodal Multimodal

Figure 13.5: Probability distributions

2. A *bimodal distribution* is a continuous probability distribution with two different modes. The modes appear as distinct peaks in the graph of the probability density function.

3. A *multimodal distribution* is a continuous probability distribution with two or more modes.

Mixture of normal distributions

Bimodal mixture

Consider the following functions which are probability density functions of normally distributed random variables.

$$f_1(x) = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} \tag{13.1}$$

$$f_2(x) = \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \tag{13.2}$$

Now consider the following function:

$$f(x) = \pi_1 f_1(x) + \pi_2 f_2(x) \tag{13.3}$$

where π_1 and π_2 are some constants satisfying the relation

$$\pi_1 + \pi_2 = 1. \tag{13.4}$$

It can be shown that the function given in Eq.(13.3) together with Eq.(13.4) defines a probability density function. It can also be shown that the graph of this function has two peaks. Hence this function defines a bimodal distribution. This distribution is called a mixture of the normal distributions defined by Eqs.(13.1) and (13.2). We may mix more than two normal distributions.

Definition

Consider the following k probability density functions:

$$f_i(x) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}, \quad i = 1, 2, \ldots, k. \tag{13.5}$$

Let $\pi_1, \pi_2, \ldots, \pi_k$ be constants such that

$$\pi_i \geq 0, \quad i = 1, 2, \ldots, k \tag{13.6}$$

$$\pi_1 + \pi_2 + \ldots + \pi_k = 1. \tag{13.7}$$

Then the random variable X whose probability density function is

$$f(x) = f_1(x) + f_2(x) + \ldots + f_k(x), \tag{13.8}$$

is said to be a *mixture of the k normal distributions* having the probability density functions defined in Eq.(13.5).

A natural example

As a natural example for such mixtures of normal populations, we consider the probability distribution of heights of people in a region. This is a mixture of two normal distributions: the distribution of heights of males and the distribution of heights of females. Given only the height data and not the gender assignments for each data point, the distribution of all heights would follow the weighted sum of two normal distributions.

Example for mixture of two normal distributions

Data and histogram

Consider the 100 observations of some attribute X given in Table 13.2.

```
 [1] 5.39 1.30 2.95 2.16 2.37 2.33 4.76 2.99 1.71 2.41
[11] 2.71 2.79 0.54 1.37 5.16 1.22 1.58 4.34 3.83 3.44
[21] 3.68 5.03 0.92 2.57 1.97 2.17 5.02 2.73 1.63 3.09
[31] 4.05 3.76 3.13 6.50 5.10 3.62 3.14 2.36 2.73 4.08
[41] 3.28 2.28 1.52 3.86 2.10 0.86 2.94 2.18 3.39 2.55
[51] 3.23 3.30 2.16 3.86 1.92 2.55 4.33 0.86 2.68 2.24
[61] 2.82 3.63 2.84 3.82 2.49 3.25 2.39 3.18 6.35 4.16
[71] 6.68 5.26 8.00 6.27 7.98 6.50 6.56 8.50 7.48 6.42
[81] 5.99 7.44 6.96 7.10 8.48 6.99 7.29 6.87 6.71 7.99
[91] 8.19 8.28 6.98 7.43 8.33 5.65 8.96 7.36 5.24 7.30
```

Table 13.2: A set of 100 observations of a numeric attribute X

To make some sense of this set of observations, let us construct the frequency table for the data as in Table 13.3.

Range	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7 -8	8-9	9-10
Frequency	4	9	26	18	6	9	12	9	7	0
Relative frequency	0.04	0.09	0.26	0.18	0.06	0.09	0.12	0.09	0.07	0.00

Table 13.3: Frequency table of data in Table 13.2

Figure 13.6 shows the histogram of the relative frequencies. Notice that the histogram has two "peaks", one near $x = 2.5$ and one near $x = 6.5$. So, the graph of the probability density function of the attribute X must have two peaks. Recall that the graph of the probability density function of a random variable having the normal distribution has only one peak.

Probability distribution

The data in Table 13.2 was generated using the R programming language. It is a true "mixture" of the values two normally distributed random variables. 70% of the observations are random values of a normally distributed random variable with $\mu_1 = 3$ and $\sigma_1 = 1.20$ and 30% of the observations are values of a normally distributed random variable with $\mu_2 = 7$ and $\sigma_2 = 0.87$. The weight for the first normal distribution is $\pi_1 = 70\% = 0.7$ and that for the second distribution is $\pi_2 = 30\% = 0.3$. The probability density function for the mixed distribution is

$$f(x) = 0.7 \times \frac{1}{1.20\sqrt{2\pi}} e^{-(x-3)^2/(2\times 1.20^2)} + 0.3 \times \frac{1}{0.87\sqrt{2\pi}} e^{-(x-7)^2/(2\times 0.87^2)}. \quad (13.9)$$

Figure 13.6 also shows the curve defined by Eq.(13.9) superimposed on the histogram of the relative frequency distribution.

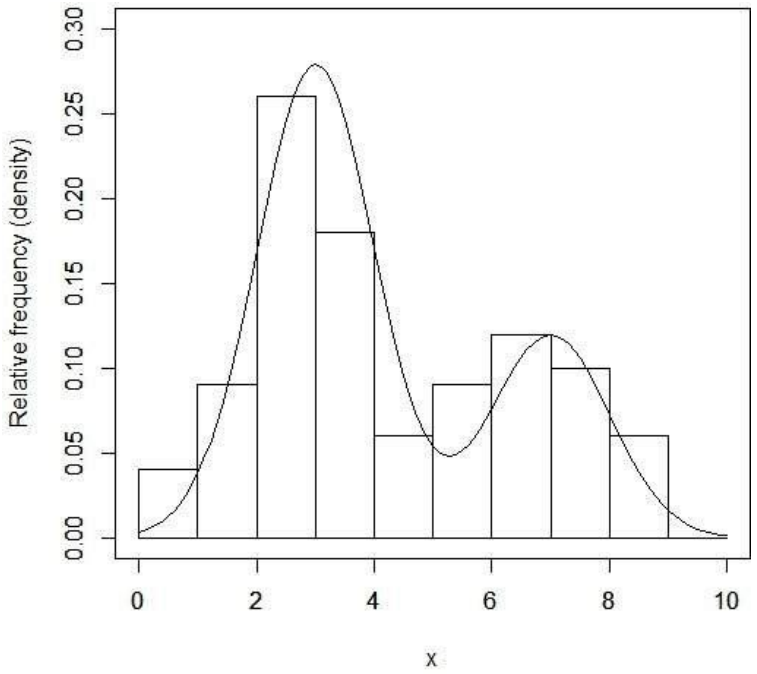


Figure 13.6: Graph of pdf defined by Eq.(13.9) superimposed on the histogram of the data in Table 13.3

Mixtures in terms of latent variables

Consider the mixture of k normal distributions defined by Eqs.(13.5) – (13.8).

Let us define a k-dimensional random variable

$$\dot{Z} = (z_1, z_2, \ldots, z_k)$$

where each z_1 is either 0 or 1 and a 1 appears only at one place; that is,

$$z_i \in \{0, 1\} \text{ and } z_1 + z_2 + ... + z_k = 0.$$

We also assume that

$$P(z_k = 1) = \pi_k.$$

The probability function of $\dot{Z}$ can be written in the form

$$P(\dot{Z}) = \pi_1^{z_1} \pi_2^{z_2} \ldots \pi_k^{z_k}.$$

Now, suppose we have a set of observations $\{x_1, x_2, \ldots, x_N\}$. Suppose that, in some way, we can associate a value of the random variable $\dot{Z}$, say $\dot{Z}_i$, with each value x_i and think of the given set of observations as a set of ordered pairs

$$\{(x_1, \dot{Z}_1), (x_2, \dot{Z}_2), \ldots, (x_N, \dot{Z}_N)\}.$$

Here, only the x_i-s are known; the $\dot{Z}_i$-s are unknown. Let us further assume that the conditional probability distribution $p(x|\dot{Z})$ be given by

$$p(x|\dot{Z}) = [f_1(x)]^{z_1} \times [f_2(x)]^{z_2} \times \ldots \times [f_k(x)]^{z_k}$$

Then the marginal distribution of x is given by

$$p(x) = \sum_{\dot{Z}} p(\dot{Z}) P(x|\dot{Z})$$

$$= \pi_1 f_1(x) + ... + \pi_k f_k(x). \tag{13.10}$$

The right hand side of Eq.(13.10) is the probability density function of a mixture of k normal distributions with weights $\pi_1, \ldots, \pi_k$.

Thus, a mixture of normal distributions is the marginal distribution of a bivariate distribution $(x, \dot{Z})$ where $\dot{Z}$ is an unobserved or latent variable.

Expectation-maximisation algorithm

The maximum likelihood estimation method (MLE) is a method for estimating the parameters of a statistical model, given observations (see Section 6.5 for details). The method attempts to find the parameter values that maximize the likelihood function, or equivalently the log-likelihood function, given the observations.

The *expectation-maximisation algorithm* (sometimes abbreviated as the *EM algorithm*) is used to find maximum likelihood estimates of the parameters of a statistical model in cases where the equations cannot be solved directly. These models generally involve latent or unobserved variables in addition to unknown parameters and known data observations. For example, a Gaussian mixture model can be described by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component to which each data point belongs.

The EM Algorithm is not really an algorithm. Rather it is a general procedure to create algorithms for specific MLE problems. The complete details of this general procedure are beyond the scope of this book. However, we present below a minimal outline of the algorithm

Outline of EM algorithm

Step 1.Initialise the parameters θ to be estimated.

Step 2. **Expectation step (E-step)**

Take the expected value of the complete data given the observation and the current parameter estimate, say, $\hat{\theta}_j$. This is a function of θ and $\hat{\theta}_j$, say, $Q(\theta, \hat{\theta}_j)$.

Step 3. **Maximization step (M-step)**

Find the values θ that maximizes the function $Q(\theta, \hat{\theta}_j)$.

Step 4.Repeat Steps 1 and 2 until the parameter values or the likelihood function converge.

The EM algorithm for Gaussian mixtures

In the case of Gaussian mixture problems, because of the nature of the function, finding a maximum likelihood estimate by taking the derivatives of the log-likelihood function with respect to all the parameters and simultaneously solving the resulting equations is nearly impossible. So we apply the EM algorithm to solve the problem.

As already indicated, the EM algorithm is a general procedure for estimating the parameters in a statistical model. This algorithm can be adapted to develop an algorithm for estimating the parameters in a Gaussian mixture model. The adapted EM algorithm has been explained below. (The details of how the EM algorithm can be adapted to estimate the parameters in a Gaussian mixture model are also beyond the scope of this book. For details on these matters, one may refer to [1]).

Problem

Suppose we are given a set of N observations

$$\{x_1, x_2, \ldots, x_N\}$$

of a numeric variable X. Let X be a mix of k normal distributions and let the probability density function of X be

$$f(x) = \pi_1 f_1(x) + \ldots + \pi_k f_k(x)$$

where

$$\pi_i \geq 0, \quad i = 1, 2, \ldots, k$$
$$\pi_1 + \ldots + \pi_k = 1$$
$$f_i(x) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}, \quad i = 1, 2, \ldots, k.$$

Estimate the parameters $\mu_1, \ldots, \mu_k$, $\sigma_1, \ldots, \sigma_k$ and $\pi_1 \ldots, \pi_k$.

Log-likelihood function

Let θ denote the set of parameters $\mu_i, \sigma_i, \pi_i (i = 1, \ldots, k)$. The log-likelihood function for the above problem is given below:

$$\log L(\theta) = \log f(x_1) + \ldots + \log f(x_N)$$
$$= \sum_{i=1}^{N} \log \left(\frac{\pi_1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}} + \ldots + \frac{\pi_k}{\sigma_k \sqrt{2\pi}} e^{-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}} \right) \tag{13.11}$$

The algorithm

Step 1.Initialise the means μ_i's, the variances σ_i^2's and the mixing coefficients π_i's.

Step 2.Calculate the following for $n = 1, \ldots, N$ and $i = 1, \ldots, k$:

$$\gamma_{in} = \frac{\pi_i f_i(x_n)}{\pi_1 f_1(x_n) + \ldots + \pi_k f_k(x_n)}$$
$$N_i = \gamma_{i1} + \ldots + \gamma_{iN}$$

Step 3.Recalculate the parameters using the following:

$$\mu_i^{(\text{new})} = \frac{1}{N_i} (\gamma_{i1} x_1 + \ldots \gamma_{iN} x_N)$$

$$\sigma_i^{2(\text{new})} = \frac{1}{N_i}(\gamma_{i1}(x_1 - \mu_i^{(\text{new})})^2 + \ldots + \gamma_{iN}(x_1 - \mu_i^{(\text{new})})^2)$$

$$\pi_i^{(\text{new})} = \frac{N_i}{N}$$

Step 4.Evaluate the log-likelihood function given in Eq.(13.11) and check for convergence of either the parameters or the log-likelihood function. If the convergence criterion is not satisfied, return to Step 2.

Hierarchical clustering

Hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters (or groups) in a given dataset. The hierarchical clustering produces clusters in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level. At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all of the data.

The decision regarding whether two clusters are to be merged or not is taken based on the *measure of dissimilarity* between the clusters. The distance between two clusters is usually taken as the measure of dissimilarity between the clusters.

In Section **??**, we shall see various methods for measuring the distance between two clusters.

Dendrograms

Hierarchical clustering can be represented by a rooted binary tree. The nodes of the trees represent groups or clusters. The root node represents the entire data set. The terminal nodes each represent one of the individual observations (singleton clusters). Each nonterminal node has two daughter nodes.

The distance between merged clusters is monotone increasing with the level of the merger. The height of each node above the level of the terminal nodes in the tree is proportional to the value of the distance between its two daughters (see Figure 13.9).

A *dendrogram* is a tree diagram used to illustrate the arrangement of the clusters produced by hierarchical clustering.

The dendrogram may be drawn with the root node at the top and the branches growing vertically downwards (see Figure 13.8(a)). It may also be drawn with the root node at the left and the branches growing horizontally rightwards (see Figure 13.8(b)). In some contexts, the opposite directions may also be more appropriate.

Dendrograms are commonly used in computational biology to illustrate the clustering of genes or samples.

Example

Figure 13.7 is a dendrogram of the dataset $\{a, b, c, d, e\}$. Note that the root node represents the entire dataset and the terminal nodes represent the individual observations. However, the dendrograms are presented in a simplified format in which only the terminal nodes (that is, the nodes representing the singleton clusters) are explicitly displayed. Figure 13.8 shows the simplified format of the dendrogram in Figure 13.7.

Figure 13.9 shows the distances of the clusters at the various levels. Note that the clusters are at 4 levels. The distance between the clusters $\{a\}$ and $\{b\}$ is 15, between $\{c\}$ and $\{d\}$ is 7.5, between $\{c, d\}$ and $\{e\}$ is 15 and between $\{a, b\}$ and $\{c, d, e\}$ is 25.

Methods for hierarchical clustering

There are two methods for the hierarchical clustering of a dataset. These are known as the *agglomerative method* (or the bottom-up method) and the *divisive method* (or, the top-down method).

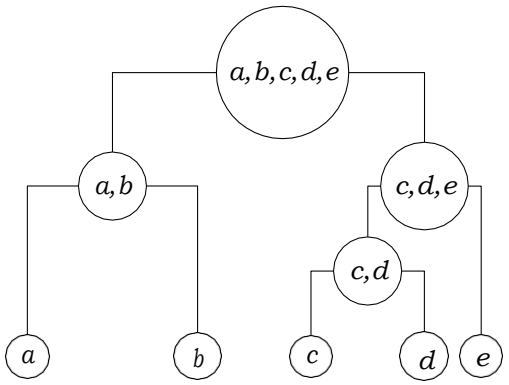


Figure 13.7: A dendrogram of the dataset {*a, b, c, d, e*}

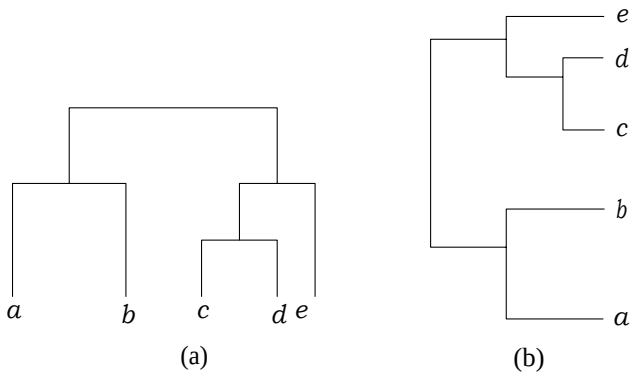


Figure 13.8: Different ways of drawing dendrogram

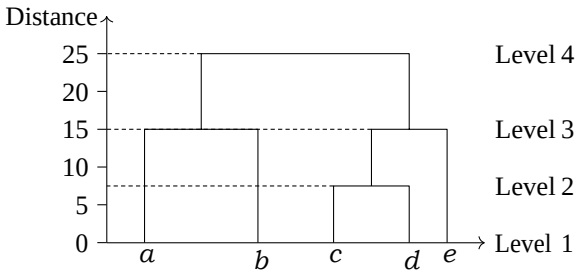


Figure 13.9: A dendrogram of the dataset {*a, b, c, d, e*} showing the distances (heights) of the clusters at different levels

Agglomerative method

In the agglomerative we start at the bottom and at each level recursively merge a selected pair of clusters into a single cluster. This produces a grouping at the next higher level with one less cluster. If there are N observations in the dataset, there will be N–1 levels in the hierarchy. The pair chosen for merging consist of the two groups with the smallest "intergroup dissimilarity".

For example, the hierarchical clustering shown in Figure 13.7 can be constructed by the agglomerative method as shown in Figure 13.10. Each nonterminal node has two daughter nodes. The daughters represent the two groups that were merged to form the parent.

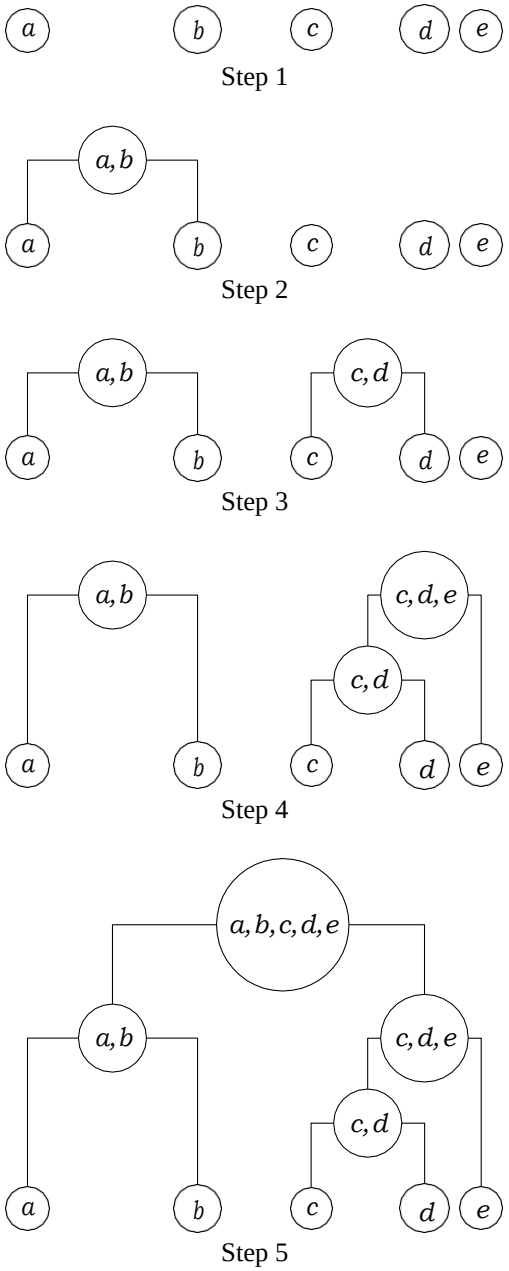


Figure 13.10: Hierarchical clustering using agglomerative method

Divisive method

The divisive method starts at the top and at each level recursively split one of the existing clusters at that level into two new clusters. If there are N observations in the dataset, there the divisive method also will produce $N-1$ levels in the hierarchy. The split is chosen to produce two new groups with the largest "between-group dissimilarity".

For example, the hierarchical clustering shown in Figure 13.7 can be constructed by the divisive method as shown in Figure 13.11. Each nonterminal node has two daughter nodes. The two daughters represent the two groups resulting from the split of the parent.

Measures of dissimilarity

In order to decide which clusters should be combined (for agglomerative), or where a cluster should be split (for divisive), a measure of dissimilarity between sets of observations is required. In most methods of hierarchical clustering, the dissimilarity between two groups of observations is measured by using an appropriate measure of distance between the groups of observations. The distance between two groups of observations is defined in terms of the distance between two observations. There are several ways in which the distance between two observations can be defined and also there are also several ways in which the distance between two groups of observations can be defined.

Measures of distance between data points

Numeric data

We assume that each observation or data point is a n-dimensional vector. Let $\dot{x} = (x_1, \ldots, x_n)$ and $\dot{y} = (y_1, \ldots, y_n)$ be two observations. Then the following are the commonly used measures of distances in the hierarchical clustering of numeric data.

Name	Formula
Euclidean distance	$\|\dot{x} - \dot{y}\|_2 = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$
Squared Euclidean distance	$\|\dot{x} - \dot{y}\|_2^2 = (x_1 - y_1)^2 + \ldots + (x_n - y_n)^2$
Manhattan distance	$\|\dot{x} - \dot{y}\|_1 = \|x_1 - y_1\| + \ldots + \|x_n - y_n\|$
Maximum distance	$\|\dot{x} - \dot{y}\|_\infty = \max\{\|x_1 - y_1\|, \ldots, \|x_n - y_n\|\}$

Non-numeric data

For text or other non-numeric data, metrics such as the Levenshtein distance are often used.

The *Levenshtein distance* is a measure of the "distance" between two words. The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

kitten → sitten (substitution of "s" for "k")
sitten → sittin (substitution of "i" for "e")
sittin → sitting (insertion of "g" at the end)

Measures of distance between groups of data points

Let A and B be two groups of observations and let x and y be arbitrary data points in A and B respectively. Suppose we have chosen some formula, say Euclidean distance formula, to measure the distance between data points. Let $d(x, y)$ denote the distance between x and y. We denote by

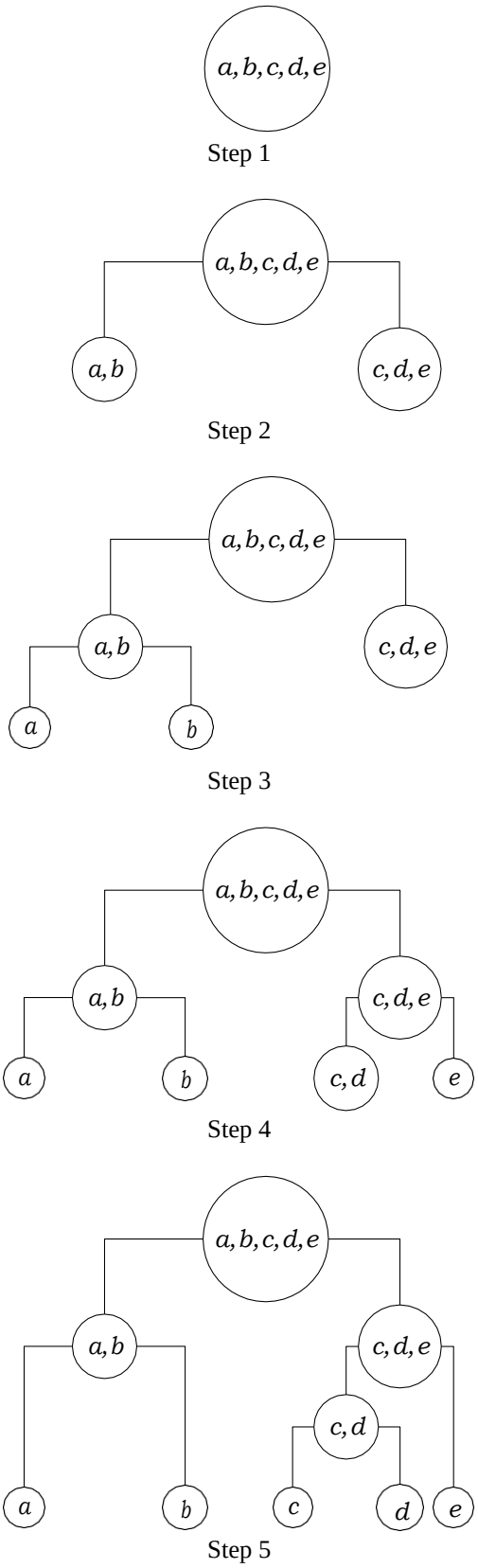


Figure 13.11: Hierarchical clustering using divisive method

$d(A, B)$ the distance between the groups A and B. The following are some of the different methods in which $d(A, B)$ is defined.

1. $d(A, B) = \max\{ d(x, y) : x \in A, y \in B \}$

   Agglomerative hierarchical clustering using this measure of dissimilarity is known as *complete-linkage clustering*. The method is also known as *farthest neighbour clustering*.

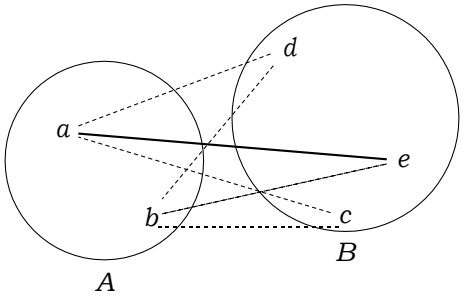


Figure 13.12: Length of the solid line "ae" is $\max\{d(x, y) : x \in A, y \in B\}$

2. $d(A, B) = \min\{ d(x, y) : x \in A, y \in B \}$

   Agglomerative hierarchical clustering using this measure of dissimilarity is known as *single-linkage clustering*. The method is also known as *nearest neighbour clustering*.

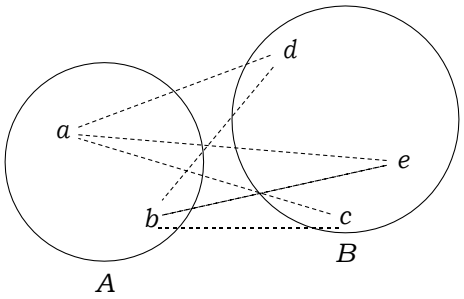


Figure 13.13: Length of the solid line "bc" is $\min\{d(x, y) : x \in A, y \in B\}$

3. $d(A, B) = \dfrac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$ where $|A|$, $|B|$ are respectively the number of elements in A and B.

   Agglomerative hierarchical clustering using this measure of dissimilarity is known as *mean or average linkage clustering*. It is also known as UPGMA (Unweighted Pair Group Method with Arithmetic Mean).

Algorithm for agglomerative hierarchical clustering

Given a set of N items to be clustered and an $N \times N$ distance matrix, required to construct a hierarchical clustering of the data using the agglomerative method.

Step 1. Start by assigning each item to its own cluster, so that we have N clusters, each containing just one item. Let the distances between the clusters equal the distances between the items they contain.

Step 2.Find the closest pair of clusters and merge them into a single cluster, so that now we have one less cluster.

Step 3.Compute distances between the new cluster and each of the old clusters.

Step 4.Repeat Steps 2 and 3 until all items are clustered into a single cluster of size N.

Example

Problem 1

Given the dataset $\{a, b, c, d, e\}$ and the following distance matrix, construct a dendrogram by complete-linkage hierarchical clustering using the agglomerative method.

	a	b	c	d	e
a	0	9	3	6	11
b	9	0	7	5	10
c	3	7	0	9	2
d	6	5	9	0	8
e	11	10	2	8	0

Table 13.4: Example for distance matrix

Solution

The complete-linkage clustering uses the "maximum formula", that is, the following formula to compute the distance between two clusters A and B:

$$d(A, B) = \max\{d(x, y) : x \in A, y \in B\}$$

1. Dataset : $\{a, b, c, d, e\}$.

   Initial clustering (singleton sets) C_1: $\{a\}, \{b\}, \{c\}, \{d\}, \{e\}$

2. The following table gives the distances between the various clusters in C_1:

	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$
a	0	9	3	6	11
$b\}$	9	0	7	5	10
$\{c\}$	3	7	0	9	**2**
$d\}$	6	5	9	0	8
$\{e\}$	11	10	**2**	8	0

   In the above table, the minimum distance is the distance between the clusters c and e . Also

$$d(\{c\}, \{e\}) = 2.$$

   We merge c and e to form the cluster c, e .

   The new set of clusters C_2: $\{a\}, \{b\}, \{d\}, \{c, e\}$

3. Let us compute the distance of $\{c, e\}$ from other clusters.

   $d(\{c, e\}, \{a\}) = \max\{d(c, a), d(e, a)\} = \max\{3, 11\} = 11.$

   $d(\{c, e\}, \{b\}) = \max\{d(c, b), d(e, b)\} = \max\{7, 10\} = 10.$

   $d(\{c, e\}, \{d\}) = \max\{d(c, d), d(e, d)\} = \max\{9, 8\} = 9.$

   The following table gives the distances between the various clusters in C_2.

{ }	{a}	{b}	{d}	{c, e}
a	0	9	6	11
b}	9	0	**5**	10
{d}	6	**5**	0	9
{c, e}	11	10	9	0

In the above table, the minimum distance is the distance between the clusters b and d. Also

$$d(\{b\}, \{d\}) = 5.$$

We merge $\{b\}$ and $\{d\}$ to form the cluster b, d.

The new set of clusters C_3: $\{a\}, \{b, d\}, \{c, e\}$

4. Let us compute the distance of $\{b, d\}$ from other clusters.

$$d(\{b, d\}, \{a\}) = \max\{d(b, a), d(d, a)\} = \max\{9, 6\} = 9.$$

$$d(\{b, d\}, \{c, e\}) = \max\{d(b, c), d(b, e), d(d, c), d(d, e)\} = \max\{7, 10, 9, 8\} = 10.$$

The following table gives the distances between the various clusters in C_3.

{ }	{a}	{b, d}	{c, e}
a	0	**9**	11
b, d}	**9**	0	10
{c, e}	11	10	0

In the above table, the minimum distance is the distance between the clusters $\{a\}$ and $\{b, d\}$. Also

$$d(\{a\}, \{b, d\}) = 9.$$

We merge $\{a\}$ and $\{b, d\}$ to form the cluster a, b, d.

The new set of clusters C_4: $\{a, b, d\}, \{c, e\}$

5. Only two clusters are left. We merge them form a single cluster containing all data points. We have

$$\begin{aligned} d(\{a, b, d\}, \{c, e\}) &= \max\{d(a, c), d(a, e), d(b, c), d(b, e), d(d, c), d(d, e)\} \\ &= \max\{3, 11, 7, 10, 9, 8\} \\ &= 11 \end{aligned}$$

6. Figure 13.14 shows the dendrogram of the hierarchical clustering.

Problem 2

Given the dataset $\{a, b, c, d, e\}$ and the distance matrix given in Table 13.4, construct a dendrogram by single-linkage hierarchical clustering using the agglomerative method.

Solution

The complete-linkage clustering uses the "maximum formula", that is, the following formula to compute the distance between two clusters A and B:

$$d(A, B) = \min\{d(x, y) : x \in A, y \in B\}$$

1. Dataset : $\{a, b, c, d, e\}$.

Initial clustering (singleton sets) C_1: $\{a\}, \{b\}, \{c\}, \{d\}, \{e\}$.

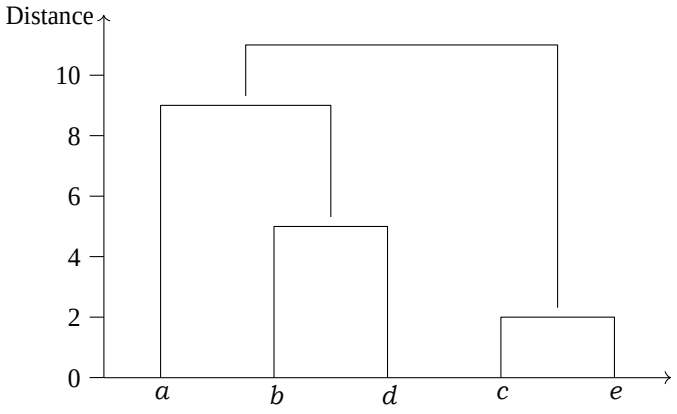


Figure 13.14: Dendrogram for the data given in Table 13.4 (complete linkage clustering)

2. The following table gives the distances between the various clusters in C_1:

	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$
a	0	9	3	6	11
$b\}$	9	0	7	5	10
$\{c\}$	3	7	0	9	**2**
$d\}$	6	5	9	0	8
$\{e\}$	11	10	**2**	8	0

In the above table, the minimum distance is the distance between the clusters c and e. Also

$$d(\{c\}, \{e\}) = 2.$$

We merge c and e to form the cluster c, e.

The new set of clusters C_2: a, b, d, c, e

3. Let us compute the distance of $\{c, e\}$ from other clusters.

$d(\{c, e\}, \{a\}) = \min(d(c, a), d(e, a)) = \max\{3, 11\} = 3.$

$d(\{c, e\}, \{b\}) = \min(d(c, b), d(e, b)) = \max\{7, 10\} = 7.$

$d(\{c, e\}, \{d\}) = \min(d(c, d), d(e, d)) = \max\{9, 8\} = 8.$

The following table gives the distances between the various clusters in C_2.

	$\{a\}$	$\{b\}$	$\{d\}$	$\{c, e\}$
a	0	9	6	**3**
$b\}$	9	0	5	7
$\{d\}$	6	5	0	8
$\{c, e\}$	**3**	7	8	0

In the above table, the minimum distance is the distance between the clusters a and c, e. Also

$$d(\{a\}, \{c, e\}) = 3.$$

We merge a and c, e to form the cluster a, c, e.

The new set of clusters C_3: $\{a, c, e\}, \{b\}, \{d\}$.

4. Let us compute the distance of $\{a, c, e\}$ from other clusters.

$d(\{a, c, e\}, \{b\}) = \min\{d(a, b), d(c, b), d(e, b)\} = \{9, 7, 10\} = 7$

$d(\{a, c, e\}, \{d\}) = \min\{d(a, d), d(c, d), d(e, d)\} = \{6, 9, 8\} = 6$

The following table gives the distances between the various clusters in C_3.

	$\{a, c, e\}$	$\{b\}$	$\{d\}$
a, c, e	0	7	6
$b\}$	7	0	**5**
$\{d\}$	6	**5**	0

In the above table, the minimum distance is between $\{b\}$ and $d\{.$ Also

$$d(\{b\}, \{d\}) = 5.$$

We merge $\{b\}$ and $d\{$ to form the cluster $b, d\{. \}$

The new set of clusters C_4: $\{a, c, e\}$, $\{b, d\}$

5. Only two clusters are left. We merge them form a single cluster containing all data points. We have

$$\begin{aligned} d(\{a, c, e\}, \{b, d\}) &= \min\{d(a, b), d(a, d), d(c, b), d(c, d), d(e, b), d(e, d)\} \\ &= \min\{9, 6, 7, 9, 10, 8\} \\ &= 6 \end{aligned}$$

6. Figure 13.15 shows the dendrogram of the hierarchical clustering.

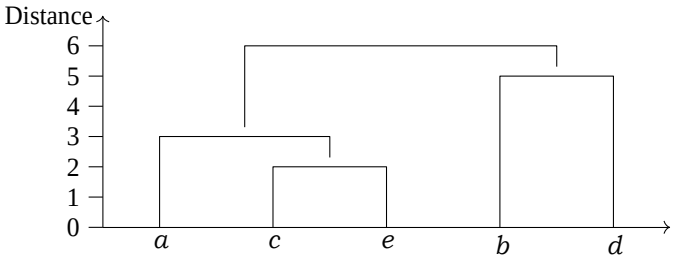


Figure 13.15: Dendrogram for the data given in Table 13.4 (single linkage clustering)

Algorithm for divisive hierarchical clustering

Divisive clustering algorithms begin with the entire data set as a single cluster, and recursively divide one of the existing clusters into two daughter clusters at each iteration in a top-down fashion. To apply this procedure, we need a separate algorithm to divide a given dataset into two clusters.

- The divisive algorithm may be implemented by using the k-means algorithm with $k = 2$ to perform the splits at each iteration. However, it would not necessarily produce a splitting sequence that possesses the monotonicity property required for dendrogram representation.

DIANA (DIvisive ANAlysis)

DIANA is a divisive hierarchical clustering technique. Here is an outline of the algorithm.

Step 1.Suppose that cluster C_l is going to be split into clusters C_i and C_j.

Step 2.Let $C_i = C_l$ and $C_j = \emptyset$.

Step 3.For each object $x \in C_i$:

(a)For the first iteration, compute the average distance of x to all other objects.

(b)For the remaining iterations, compute

$$D_x = \text{average } \{d(x, y) : y \in C_i\} - \text{average}\{d(x, y) : y \in C_j\}.$$

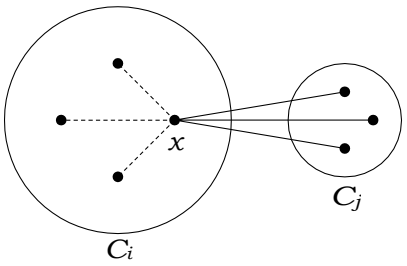


Figure 13.16: D_x= (average of dashed lines) − (average of solid lines)

Step 4.(a)For the first iteration, move the object with the maximum average distance to C_j.

(b)For the remaining iterations, find an object x in C_i for which D_x is the largest. If $D_x > 0$ then move x to C_j.

Step 5.Repeat Steps 3(b) and 4(b) until all differences D_x are negative. Then C_l is split into C_i and C_j.

Step 6.Select the smaller cluster with the largest diameter. (The diameter of a cluster is the largest dissimilarity between any two of its objects.) Then divide this cluster, following Steps 1-5.

Step 7.Repeat Step 6 until all clusters contain only a single object.

Example

Problem

Given the dataset $\{a, b, c, d, e\}$ and the distance matrix in Table 13.4, construct a dendrogram by the divisive analysis algorithm.

Solution

1. We have, initially

$$C_l = \{a, b, c, d, e\}$$

2.We write

$$C_i = C_l, \quad C_j = \emptyset.$$

3. Division into clusters

(a) Initial iteration

Let us calculate the average dissimilarities of the objects in C_i with the other objects in C_i.

Average dissimilarity of a

$$= \frac{1}{4}(d(a,b)+d(a,c)+d(a,e)) = \frac{1}{4}(9+3+6+11) = 7.25$$

Similarly we have :

Average dissimilarity of b 7.75

Average dissimilarity of c 5.25

Average dissimilarity of d 7.00

Average dissimilarity of $e = 7.75$

The highest average distance is 7.75 and there are two corresponding objects. We choose one of them, b, arbitrarily. We move b to C_j.

We now have

$$C_i = \{a, c, d, e\}, \quad C_j = \emptyset \cup \{b\} = \{b\}.$$

(b) Remaining iterations

(i)2-nd iteration.

$$D = \frac{1}{a}(d(a, c) + d(a, d) + d(a, e)) - \frac{1}{3}(d(a, b)) = \frac{20}{3} - 9 = -2.33$$

$$D = \frac{1}{c}(d(c, a) + d(c, d) + d(c, e)) - \frac{1}{1}(d(c, b)) = \frac{14}{3} - 7 = -2.33$$

$$D = \frac{1}{d}(d(d, a) + d(d, c) + d(d, e)) - \frac{1}{3}(d(c, b)) = \frac{23}{3} - 7 = 0.67$$

$$D = \frac{1}{e}(d(e, a) + d(e, c) + d(e, d)) - \frac{1}{3}(d(e, b)) = \frac{21}{3} - 7 = 0$$

D_d is the largest and $D_d > 0$. So we move, d to C_j.

We now have

$$C_i = \{a, c, e\}, \quad C_j = \{b\} \cup \{d\} = \{b, d\}.$$

(ii)3-rd iteration

$$D = \frac{1}{a}(d(a, c) + d(a, e)) - \frac{1}{2}(d(a, b) + d(a, d)) = \frac{14}{2} - \frac{15}{2} = -0.5$$

$$D_c = \frac{1}{2}(d(c,a)+d(c,e)) - \frac{1}{2}(d(c,b)+d(c,d)) = \frac{5}{2} - \frac{16}{2} = -13.5$$

$$D_e = \frac{1}{2}(d(e,a)+d(e,c)) - \frac{1}{2}(d(e,b)+d(e,d)) = \frac{13}{2} - \frac{18}{2} = -2.5$$

All are negative. So we stop and form the clusters C_i and C_j.

4. To divide, C_i and C_j, we compute their diameters.

$$\begin{aligned} \text{diameter}(C_i) &= \max\{d(a, c), d(a, e), d(c, e)\} \\ &= \max\{3, 11, 2\} \\ &= 11 \\ \text{diameter}(C_j) &= \max\{d(b, d)\} \\ &= 5 \end{aligned}$$

The cluster with the largest diameter is C_i. So we now split C_i.

We repeat the process by taking $C_l = \{a, c, e\}$. The remaining computations are left as an exercise to the reader.

Density-based clustering

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

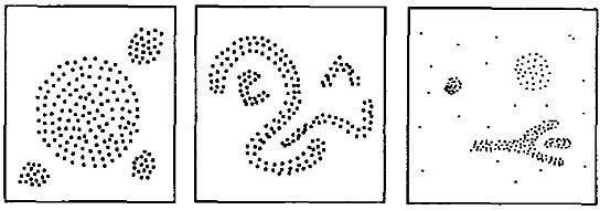


Figure 13.17: Clusters of points and noise points not belonging to any of those clusters

Density

We introduce some terminology and notations.

- Let s (epsilon) be some constant distance. Let p be an arbitrary data point. The *s-neighbourhood of p* is the set
$$N_s(p) = \{q : d(p,q) < s\}$$
- We choose some number m_0 to define points of "high density": We say that a point p is point of *high density* if $N_s(p)$ contains at least m_0 points.
- We define a point p as a *core point* if $N_s(p)$ has more than m_0 points.
- We define a point p as a *border point* if $N_s(p)$ has fewer than m_0 points, but is in the s-neighbourhood of a core point.
- A point which is neither a core point nor a border point is called a *noise point*.

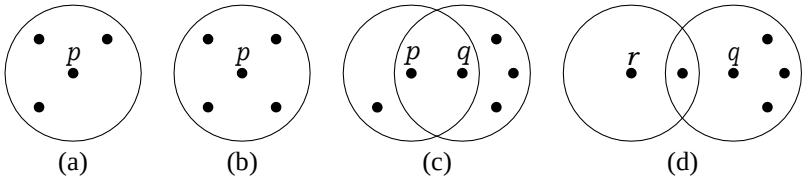


Figure 13.18: With $m_0 = 4$: (a) p a point of high density (b) p a core point (c) p a border point (d) r a noise point

- An object q is *directly density-reachable* from object p if p is a core object and q is in $N_s(p)$.
- An object q is *indirectly density-reachable* from an object p if there is a finite set of objects $p_1, \ldots, p_r$ such that p_1 is directly density-reachable form p, p_2 is directly density reachable from p_1, etc., q is directly density-reachable form p_r.

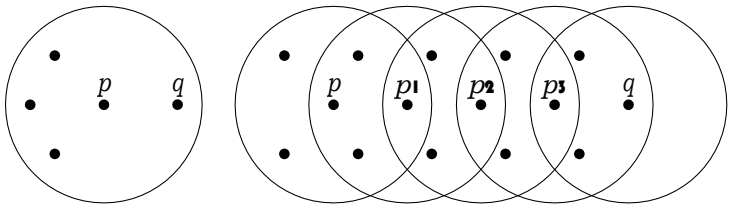


Figure 13.19: With $m_0 = 4$: (a) q is directly density-reachable from p (b) q is indirectly density-reachable from p

DBSCAN algorithm

Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of data points. DBSCAN requires two parameters: s (eps) and the minimum number of points required to form a cluster (m_0).

Step 1. Start with an arbitrary starting point p that has not been visited.

Step 2. Extract the s-neighborhood $N_s(p)$ of p.

Step 3. If the number of points in $N_s(p)$ is not greater than m_0 then the point p is labeled as noise (later this point can become the part of the cluster).

Step 4. If the number of points in $N_s(p)$ is greater than m_0 then the point p is a core point and is marked as visited. Select a new *cluster-id* and mark all objects in $N_s(p)$ with this cluster-id.

Step 5. If a point is found to be a part of the cluster then its s-neighborhood is also the part of the cluster and the above procedure from step 2 is repeated for all s-neighborhood points. This is repeated until all points in the cluster are determined.

Step 6. A new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

Step 7. This process continues until all points are marked as visited.

Sample questions

(a) Short answer questions

1. What is clustering?
2. Is clustering supervised learning? Why?
3. Explain some applications of the k-means algorithm.
4. Explain how clustering technique is used in image segmentation problem.
5. Explain how clustering technique used in data compression.
6. What is meant by the mixture of two normal distributions?
7. Explain hierarchical clustering.
8. What is a dendrogram? Give an example.
9. Is hierarchical clustering unsupervised learning? Why?
10. Describe the two methods for hierarchical clustering.

11. In a clustering problem, what does the measure of dissimilarity measure? Give some examples of measures of dissimilarity.

12. Explain the different types of linkages in clustering.

13. In the context of density-based clustering, define high density point, core point, border point and noise point.

14. What is agglomerative hierarchical clustering?

(b) Long answer questions

1. Apply k-means algorithm for given data with $k = 3$. Use $C_1(2)$, $C_2(16)$ and $C_3(38)$ as initial centers. Data:

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

2. Explain K-means algorithm and group the points (1, 0, 1), (1, 1, 0), (0, 0, 1) and (1, 1, 1) using K-means algorithm.

3. Applying the k-means algorithm, find two clusters in the following data.

x	185	170	168	179	182	188	180	180	183	180	180	177
y	72	56	60	68	72	77	71	70	84	88	67	76

4. Use k-means algorithm to find 2 clusters in the following data:

No.	1	2	3	4	5	6	7
x_1	1.0	1.5	3.0	5.0	3.5	4.5	3.5
x_2	1.0	2.0	4.0	7.0	5.0	5.0	4.5

5. Give a general outline of the expectation-maximization algorithm.

6. Describe EM algorithm for Gaussian mixtures.

7. Describe an algorithm for agglomerative hierarchical clustering.

8. Given the following distance matrix, construct the dendrogram using agglomerative clustering with single linkage, complete linkage and average linkage.

	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

9. Describe an algorithm for divisive hierarchical clustering.

10. For the data in Question 8, construct a dendrogram using DIANA algorithm.

11. Describe the DBSCAN algorithm for clustering.