PATTERN RECOGNITION

Basics of Pattern Recognition	2
Bayesian Decision Theory	8
Parameters Estimation Methods	26
Hidden Markov Models for Sequential Pattern Classification	42
Dimension Reduction Methods	52
Non-Parametric Techniques for Density Estimation	67
Linear Discriminant Function Based Classifier	75
Non Metric Method for Pattern Classification	97
Unsupervised Learning and Clustering	105

NOTE:

MAKAUT course structure and syllabus of 6th semester has been changed from 2021. **PATTERN RECOGNITION** has been introduced as a new subject in present curriculum. Taking special care of this matter we are providing chapterwise model questions & answer, so that students can get an idea about university questions patterns.

BASICS OF PATTERN RECOGNITION

Multiple Choice Type Questions

1. Which of the following is an example of Pattern Recognition? [MODEL QUESTION]

a) Speech recognition
b) Speaker identification
c) MDR
d) All of the above

Answer: (d)

2. Pattern recognition solves the problem of fake bio metric detection. [MODEL QUESTION]

a) TRUE
b) FALSE
c) Can be true or false
d) cannot say

Answer: (a)

3. Which of the following is disadvantages pattern recognition? [MODEL QUESTION]

a) Syntactic Pattern recognition approach is complex to implement
b) It is very slow process
c) Sometime to get better accuracy, larger dataset is required
d) All of these

Answer: (d)

4. In a typical pattern recognition application, the raw data is processed and converted into a form that is amenable for a machine to use. [MODEL QUESTION]

a) TRUE
b) FALSE
c) Can be true or false
d) cannot say

Answer: (a)

5. _____ is the process of recognizing patterns by using machine learning algorithm. [MODEL QUESTION]

a) Processed Data
b) Literate Statistical Programming
c) Pattern Recognition
d) Likelihood

Answer: (c)

Short Answer Type Questions

1. What is pattern recognition? [MODEL QUESTION]

Answer:

Pattern recognition is the process of recognizing patterns by using a machine learning algorithm. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation. One of the important aspects of pattern recognition is its application potential.

Examples: Speech recognition, speaker identification, multimedia document recognition (MDR), automatic medical diagnosis.
In a typical pattern recognition application, the raw data is processed and converted into a form that is amenable for a machine to use. Pattern recognition involves the classification and cluster of patterns.

- In classification, an appropriate class label is assigned to a pattern based on an abstraction that is generated using a set of training patterns or domain knowledge. Classification is used in supervised learning.
- Clustering generated a partition of the data which helps decision making, the specific decision-making activity of interest to us. Clustering is used in unsupervised learning.

Features may be represented as continuous, discrete, or discrete binary variables. A feature is a function of one or more measurements, computed so that it quantifies some significant characteristics of the object.

Example: consider our face then eyes, ears, nose, etc are features of the face. A set of features that are taken together, forms the **features vector**.
Example: In the above example of a face, if all the features (eyes, ears, nose, etc) are taken together then the sequence is a feature vector([eyes, ears, nose]). The feature vector is the sequence of a feature represented as a d-dimensional column vector. In the case of speech, MFCC (Mel-frequency Cepstral Coefficient) is the spectral feature of the speech. The sequence of the first 13 features forms a feature vector.

Pattern recognition possesses the following features:

- Pattern recognition system should recognize familiar patterns quickly and accurate
- Recognize and classify unfamiliar objects
- Accurately recognize shapes and objects from different angles
- Identify patterns and objects even when partly hidden
- Recognize patterns quickly with ease, and with automaticity.

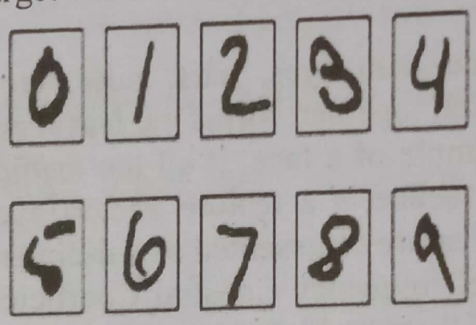
Long Answer Type Questions

1. Explain the idea of pattern recognition. **[MODEL QUESTION]**

Answer:
The problem of searching for patterns in data is a fundamental one and has a long and successful history. For instance, the extensive astronomical observations of Tycho Brahe in the 16th century allowed Johannes Kepler to discover the empirical laws of planetary motion, which in turn provided a springboard for the development of classical mechanics. Similarly, the discovery of regularities in atomic spectra played a key role in the development and verification of quantum physics in the early twentieth century. The field of pattern recognition is concerned with the automatic discovery of regularities in data

through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.

Consider the example of recognizing handwritten digits, illustrated in below figure each digit corresponds to a 28×28 pixel image and so can be represented by a vector x comprising 784 real numbers. The goal is to build a machine that will take such a vector x as input and that will produce the identity of the digit 0, ..., 9 as the output. This is a nontrivial problem due to the wide variability of handwriting. It could be far better results can be obtained by adopting a machine learning approach in which a large set of N digits $\{x_1, \ldots, x_N\}$ called a training set is used to tune the parameters of an adaptive model. The categories of the digits in the training set are known in advance, typically by inspecting them individually and hand-levelling them. We can express the category of a digit using target vector t, which represents the identity of the corresponding digit. Suitable techniques for representing categories in terms of vectors will be discussed later. Note that there is one such target vector t for each digit image x.



Examples of hand-written
digits taken from US zip codes

Pre-processing might also be performed in order to speed up computation. For example, if the goal is real-time face detection in a high-resolution video stream, the computer must handle huge numbers of pixels per second and presenting these directly to a complex pattern recognition algorithm may be computationally infeasible. Instead, the aid is to find useful features that are fast to compute and yet that also preserve useful discriminatory information enabling faces to be distinguished from non-faces. These features are then used as the inputs to the pattern recognition algorithm. For instance, the average value of the image intensity over a rectangular sub-region can be evaluated extremely efficiently (Viola and Jones, 2004) and a set of such features can prove very effective in fast face detection. Because the number of such features is smaller than the number of pixels, this kind of pre-processing represents a form of dimensionality reduction. Care must be taken during pre-processing because often information is discarded and if this information is important to the solution of the problem, then the overall accuracy of the system can suffer.

In other pattern recognition problems, the training data consists of a set of input vectors x without any corresponding target values. The goal in such unsupervised learning problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

2. What are the scopes of pattern recognition? **[MODEL QUESTION]**

Answer:

Pattern Recognition is a mature but exciting and fast developing field, which underpins developments in cognate fields such as computer vision, image processing, text and document analysis and neural networks. It is closely akin to machine learning and also finds applications in fast emerging areas such as biometrics, bio-informatics, multimedia data analysis and most recently data science. The journal Pattern Recognition was established some 50 years ago, as the field emerged in the early years of computer science. Over the intervening years it has expanded considerably.

Scope of PR in Machine Learning:

- **Data Mining:** It refers to the **extraction** of useful information from large amounts of data from heterogeneous sources. The meaningful data obtained from data **mining techniques** are used for prediction making and data analysis.
- **Recommender Systems:** Most of the websites dedicated to online shopping make use of recommender systems. These systems collect data related to each customer purchase and make suggestions using machine learning algorithms by identifying the trends in the pattern of customer purchase.
- **Image Processing:** Image process is basically of two types – Digital Image Processing and Analog Image Processing. Digital image processing uses intelligent machine learning algorithms for enhancing the quality of the image obtained from distant sources such as satellites.
- **Bio Informatics:** It is a field of science that uses compotation tools and software to make predictions relating to biological data. for example, suppose someone discovered a new protein in the lab but the sequence of the protein is not known. Using bio-informatics tools, the unknown protein is compared with a huge number of proteins stored in the database to predict a sequence based on similar patterns.
- **Analysis:** Pattern recognition is used for identifying important data trends. These trends can be used for future predictions. An analysis is required in almost every domain be it technical or non-technical. For example, the tweets made by a person on twitter helps in the sentiment analysis by identifying the patterns in the posts using natural language processing.

3. What are challenges in pattern recognition? **[MODEL QUESTION]**

Answer:

- **Data Collection:** The input data, whatever its form is, is sampled at fixed intervals in time or image metric domain and digitised to be presented with a preset number of bits per measurement. Any additional noise will be disadvantageous to successful operation of the system. So, to have a "clean" input, we have to use a lot of techniques such as filtering. Sometimes, for example, online processing, we also have to store/transfer a lot of data, the real problem may be come up is bottleneck.
- **Segmentation:** Depending on how the application has been realized, the segmentation block may either add the information regarding the segment boundaries

to the data flow, or alternatively, copy all the segments in separate buffers and hand them over to the following stage one by one. There are plenty of research in this field, but it is still an open-problem.

- **Feature Extraction:** It is still a big question for every researcher because of some obvious questions such as how objects are described, how extensive may this description be, what are the ways to incorporate knowledge from the application domain? At this moment, we don't have a completely answer for all kind of objects. So, the only thing can help is doing research and study existing research on the application domain.
- **Classification:** This is the most crucial step in the process of pattern recognition. The primary division of the various classification algorithms used is that between syntactic and statistical methods. Numerous taxonomies for classification methods in pattern recognition have been presented. None has been so clearly more advantageous than the others that it would have gained uncontested status.

4. What are applications of pattern recognition? **[MODEL QUESTION]**

Answer:

1. **Natural Language Processing (NLP):** The Pattern Recognition algorithms ae used in NLPs for building strong software systems that have further applications in the computer and communications industry.
2. **Network Intrusion Detection:** Network intrusion detection is one of the sectors of security. The intrusion is one of the serious threats posed to any data firm. Thus, the PR system applications help in intrusion detection by recognizing patterns of intrusion over time. This ensures security systems to be at alarm if the slightest of patterns of intrusion show their traces over the network.
3. **Disease Categorization:** The PR systems have been employed in disease recognition and imaging.
4. **Image Sensing and Recognition:** Pattern recognition well suits the image processing and its segmentation. The analysis is then performed. This is forwarded to expert reviews. PR algorithms have gradually incorporated intelligence, similar to humans. Machine learning has boosted their recognition powers in medical image sensing and recognizing.
5. **Data Mining and Warehousing Patterns or Knowledge Discovery:** The KDD and other algorithms are used for finding patterns when performing data mining activities.
6. **Acting as Eyes in Computer Vision:** Pattern recognition algorithms are widely used in computer vision. They help in extracting meaningful features from excerpts of images, videos, etc. There are applications in biomedical and medical imaging of diseases.
7. **Prediction of Survival Rates for Patients with Specific Disease:** The probability rates of patients can also be predicted with pattern recognition algorithms.
8. **Seismic Analysis:** PR approaches are used for findings, imaging and elucidation of sequential patterns of seismic display of recorded data. in this application, we

implement Statistical pattern recognition techniques. They are used in variants of seismic analysis and data models.

9. **Radar Signal Recognition and Analysis:** The Pattern recognition schemes are used in radar signal and classification. Signal processing methods are used in various applications of radar signal classifications like AP mine detection and identification.
10. **Speech Recognition:** The huge success of pattern recognition is seen in the speech recognition domain. the linguistics and PR systems are going hand in hand with the research and developments. It uses algorithms that are competitive and is able to treat large data sets simultaneously.
11. **Agriculture:** In the Agriculture industry, we have a lot of applications which are reflected in the contribution of economic benefits. It works hand in hand with the breeding industry; the researchers are using multiple pattern recognition schemes for research for identification, improvement and breed key traits. This leads to dealing with the rising production demands, increase the resistance to various diseases, reducing the threats to the environment by using less water, fertilizers, etc.
12. **Financial Services:** In financial companies, the PR systems are helping data recognition related to trends in the financial markets. They are doing the job of identifying they are able to identify key insights. This might prevent financial crashes and save society from financial troubles. This technology is further used to make investments and expand businesses. Cyber surveillance is one fo the examples that help in timely recognizing risks and taking steps to prevent them.
13. **Finger Print Identification:** Fingerprint recognition technology is a dominant technology in the biometric market. A number of recognition methods have been used to perform fingerprint matching out of which pattern recognition approaches are widely used.
14. **Texture Discrimination:** The textile industry makes use of PR systems for texture determination for its clients based on their housing needs.
15. **Transportation:** The PR systems are expanding its applications to the transportation sector as well. On the basis of travel history, the mould of routes, packages, destinations and costs are made pre available to the customers using PR systems. The transportation companies thus are able to forecast potential risks that might occur on certain routes and suitably counsel their customers timely.

BAYESIAN DECISION THEORY

Multiple Choice Type Questions

1. Which of the following statement is TRUE about the Bayes classifier? **[MODEL QUESTION]**

a) Bayes classifier works on the Bayes theorem of probability.
b) Bayes classifier is an unsupervised learning algorithm.
c) Bayes classifier is also known as maximum apriori classifier.
d) It assumes the independence between the independent variables or features.

Answer: (a)

2. How do we perform Bayesian classification when some features are missing? **[MODEL QUESTION]**

a) We assuming the missing values as the mean of all values.
b) We ignore the missing features.
c) We integrate the posteriors probabilities over the missing features.
d) Drop the features completely.

Answer: (c)

Short Answer Type Questions

1. What is random variable? Narrate Bayes theorem. **[MODEL QUESTION]**

Answer:

Random Variable

A random variable is a function that maps a possible set of outcomes to some values like while tossing a coin and getting head H as 1 and Tail T as 0 where 0 and 1 are random variables.

Bayes Theorem

The conditional probability of A given B, represented by P(A|B) is the chance of occurrence of A given that B has occurred.

P(A|B) = P(A, B)/P(B) or

By Using the Chain rule, this can also be written as:

P(A, B) = P(A|B)P(B)=P(B|A)P(A)

P(A|B) = P(B|A)P(A)/P(B) .... (1)

Where, **P(B) = P(B, A) + P(B, A') = P(B|A)P(A) + P(B|A')P(A')**

Here, Eqn. (1) is known as the **Bayes Theorem of probability**

2. What is prior probability? **[MODEL QUESTION]**

Answer:

The probability is calculated according to the past occurrences of the outcomes (i.e. events). This is called the **prior probability** ("prior" meaning "before"). In other words, the prior probability refers to the probability in the past.

Assume that someone asks who will be the winner of a future match between two teams. Let AA and BB refer to the first or second team winning, respectively.
In the last 10 cup matches, AA occurred 4 times and BB occurred the remaining 6 times. So, what is the probability that AA occurs in the next match? Based on the experience (i.e., the events that occurred in the past), the prior probability that the first team (AA) wins in the next match is:

$$P(A) = \frac{4}{10} = 0.4$$

But the post events may not always hold, because the situation or context may change. For example, team A could have won only 4 matches because there were some injured players. When the next match comes, all of these injured players will have recovered. Based on the current situation, the first team may win the next match with a higher probability than the one calculated based on past events only.
The prior probability measures the probability of the next action without taking into consideration a current observation (i.e., the current situation). It's like predicting that a patient has a given disease based only on past doctors visits.
In other words, because the prior probability is solely calculated based on past events (without present information), this can degrade the prediction value quality. The past predictions of the two outcomes A and B may have occurred while some conditions were satisfied, but at the current moment, these conditions may not hold.

3. What is likelihood probability? **[MODEL QUESTION]**

Answer:

The likelihood helps to answer the question: given some conditions, what is the probability that an outcome occurs? It is denoted as follows:

$$P(X \mid C_i)$$

where, X refers to the conditions and C_i refers to the outcome. Because there may be multiple outcomes, the variable C is given the subscript i.

The likelihood is read as follows:

"Under a set of conditions X, what is the probability that the outcome is C_i?"

According to our example of predicting the winning team, the probability that the outcome A occurs does not only depend on past events, but also on current conditions. The likelihood relates the occurrence of an outcome to the current conditions at the time of making a prediction.
Assume the conditions change so that the first team has no injured players, while the second team has many injured players. As a result, it is more likely that A occurs than B. Without considering the current situation and using only the prior information, the outcome would be B, which is not accurate given the current situation.
For the example of diagnosiing a patient, this could be an understandably better prediction, as the diagnosis will take into account their current symptoms rather than their prior condition.

A drawback of using only the likelihood is that it neglects experience (prior probability), which is useful in many cases. So, a better way to do a predictio is to combine them both.

4. What is Bayesian decision theory? **[MODEL QUESTION]**

Answer:

Bayesian Decision Theory (i.e., the Bayesian Decision Rule) predicts the outcome not only based on previous observations, but also by taking into account the current situation. **The rule describes the most reasonable action to take based on an observation.**

The formula for Bayesian (Bayes) decision theory is given below:

$$P(C_i \mid X) = \frac{P(C_i)P(X \mid C_i)}{P(X)}$$

The elements of the theory are:

- $P(C_i)$: Prior probability. This accounts for how many times the calss C_i occurred independently from any conditions (i.e., regardless of the input X).
- $P(X \mid C_i)$: Likelihood. Under some conditions X, this is how many times the outcome C_i occurred.
- $P(X)$: Evidence. The number of times the conditions X occurred.
- $P(C_i \mid X)$: Posterior. The probability that the outcome C_i occurs given some conditions X.

Bayesian Decision Theory gives balsnced predictions, as it takes into consideration the following:

1. $P(X)$: How many times did the conditions X occur?
2. $P(C_i)$: How many times did the outcome C_i occur?
3. $P(X \mid C_i)$: How many times did both the conditions X and the outcome C_i occur together?

If any of the previous factors was not used, the prediction would be hindered. Let's explain the effect of excluding any of these factors and mention a case where using each factor might help.

- $P(C_i)$: Assume that the prior probability $P(C_i)$ is not used; then we cannot know whether the outcome C_i occurs frequently or not. If the prior probability is high, then the outcome C_i frequently occurs and it is an indication that it may occur again.
- $P(X \mid C_i)$: If the likelihood probability $P(X \mid C_i)$ is not used, then there is no information to associate the current input X with the outcome C_i for example, the outcome C_i may have occurred frequently, but it rarely occurs with the current input X.
- $P(X)$: If the evidence probability $P(X)$ is excluded, then there is no information to reflect the frequency of X occurring. Assuming that both the

outcome C_i and the input X occur frequently, then it is probable that the outcome is C_i when the input is X.

When there is information about the frequency of the occurrence of C_i alone, X alone and both C_i and X together, then a better prediction can be made.

There are some things to note about the theory / rule:

1. The sum of all prior probabilities must be 1.
2. The sum of all posterior probabilities must be 1.
3. The evidence is the sum of products of the prior and likelihood probabilities of all outcomes.

Long Answer Type Questions

1. Explain Bayesian decision theory. **[MODEL QUESTION]**

Answer:

Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification. This approach is based on quantifying the tradeoffs between various classification decisions using probability and the costs that accompany such decisions. It makes the assumption that the decision problem is posed in probabilistic terms and that all of the relevant probability values are known.

Let us reconsider the hypothetical problem of designing a classifier to separate two kinds of fish: sea bass and salmon. Suppose that an observer watching fish arrive along the conveyor belt finds it hard to predict what type will emerge next and that the sequence of types of fish appears t be random. In decision-theoretic terminology we would say that as each fish emerges natue is in one or the other of the two possible states: either the fish is a sea bass or the fish in a salmon. We let ω denote the *state of nature*, with $\omega = \omega_1$ for sea bass and $\omega = \omega_2$ for salmon. Because the state of nature is so unpredictable, we consider ω to be a variable that must be described probabilistically.

If the catch produced as much sea bass as salmon, we would say that the next fish is equally likely to be sea bass or salmon. More generally, we assume that there is some a *priori probability* (or simply *prior*) $P(\omega_1)$ that the next fish is sea bass and some prior probability $P(\omega_2)$ that it is salmon. If we assume there are no other types of fish relevant here, then $P(\omega_1)$ and $P(\omega_2)$ sum to one. These prior probabilities reflect our prior knowledge of how likely we are to get a sea bass or salmon before the fish actually appears. It might, for instance, depend upon the time of year or the choice of fishing area. Suppose for a moment that we were forced to make a decision about the type of fish that will appear next without being allowed to see it. For the moment, we shall assume that any incorrect classification entails the same cost or consequence and that the only information we are allowed to use is the value of the prior probabilities. If a decision must be made with so little information, it seems logical to use the following *decision rule*: Decide ω_1 if $P(\omega_1) > P(\omega_2)$; otherwise decide ω_2.

This rule makes sense if we are to judge just one fish, but if we are to judge many fish, using this rule repeatedly may seem a bit strange. After all, we would always make the same decision even though we know that *both* types of fish will appear. How well it works depends upon the values of the prior probabilities. If $P(\omega_1)$ is very much greater than $P(\omega_2)$, our decision in favour of ω_1 will be right most of the time. If $P(\omega_1)=P(\omega_2)$, we have only a fifty-fifty chance of being right. In general, the probability of eror is the smaller of $P(\omega_1)$ and $P(\omega_2)$ and we shall see later that under these conditions no other decision rule can yield a larger probability of being right.

In most circumstances we are not asked to make decisions with so little information. In our example, we might for instance use a lightness measurement x to improve our classifier. Different fish will yield different lightness readings and we express this variability in probabilistic terms; we consider x to be a continuous random variable whose distribution depends on the state of nature and is expressed as. This is the *class-conditional probability density* function. Strictly speaking, the probability density function $p(x|\omega_1)$ should be written as $p_X(x|\omega_1)$ to indicate that we are speaking about a particular density function for the random variable X. This more elaborate subscripted notation makes it clear that $p_X(\cdot)$ and $p_Y(\cdot)$. Since this potential confusion rarely arises in practice, we have elected to adopt the simpler notation.

This is the probability density function for x given that the state of nature is ω_1. (It is also sometimes called state-conditional probability density). Then the difference between $p(x|\omega_1)$ and $p(x|\omega_2)$ describes the difference in lightness between populations of sea bass and salmon (Fig. 1).

Suppose that we know both the prior probabilities $P(\omega_j)$ and the conditional densities $p(x|\omega_j)$. Suppose further that we measure the lightness of a fish and discover that its value is x. How does this measurement influence our attitude concerning the ture state of nature — that is, the category of the fish? We note first that the (joint) probability density of finding a pattern that is in category ω_j and has feature value x can be written two ways: $p(\omega_j, x)=P(\omega_j|x)p(x)=p(x|\omega_j)P(\omega_j)$. Rearranging these leads us to the answer to our question, which is called *Bayes' formula*:

$$\boxed{P(\omega_j|x)=\frac{p(x|\omega_j)P(\omega_j)}{p(x)}} \qquad \text{.... (1)}$$

where in this case of two categories

$$p(x)=\sum_{j=1}^{2} p(x|\omega_j)P(\omega_j) \qquad \text{.... (2)}$$

Bayes' formula can be expressed informally in English by saying that

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \qquad \text{.... (3)}$$

bayes' formula shows that by observing the value of x we can convert the prior probability $P(\omega_j)$ to the a *posteriori* probability (or *posterior*) probability $P(\omega_j | x)$ — the probability of the state of nature being ω_j given that feature value x has been measured. We call $p(x|\omega_j)$ the *likelihood* of ω_j with respect to x (a term chosen to indicate that, other things being equal, the category ω_j for which $p(x|\omega_j)$ is large is more "likely" to be the true category). Notice that it is the product of the likelihood and the prior propbability that is most important in determining the posterior probability; the evidence factor, $p(x)$, can be viewed as merely a scale factor that guarantees that the posterior probabilities sum to one, as all good probabilities must. The variation of $P(\omega_j | x)$ with x is illustrated in Fig. 2 for the case $P(\omega_1) = \frac{2}{3}$ and $P(\omega_2) = \frac{1}{3}$.

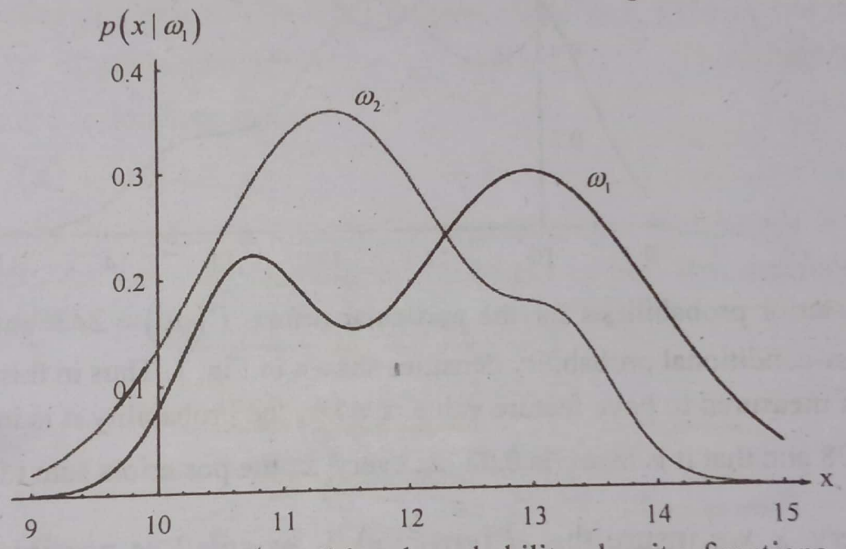


Fig: 1. Hypothetical class-conditional probability density functions show the probability density of measuring a particular feature value x given the pattern is in category ω_j. If x represents the length of a fish, the two curves might describe the difference in length of populations of two types of fish. Density functions are normalized and thus the area under each curve is 1.0

If we have an observation x for which $P(\omega_1 | x)$ is greater than $P(\omega_2 | x)$, we would naturally be inclined to decide that the true state of nature is ω_1. Similarly, if $P(\omega_2 | x)$ is greater than $P(\omega_1 | x)$, we would be inclined to choose ω_2. To justify this decision procedure, let us calculate the probability of error whenever we make a decision. Whenever we observe a particular x,

$$P(error|x)=\begin{cases}P(\omega_1|x) & \text{if we decide } \omega_2\\ P(\omega_2|x) & \text{if we decide } \omega_1\end{cases} \quad \text{.... (4)}$$

Clearly, for a given x we can minimize the probability of error by deciding ω_1 if $P(\omega_1|x)>P(\omega_2|x)$ and ω_2 otherwise. Of course, we may never observe exactly the same value of x twice. Will this rule minimize the average probability of error? Yes, because the average probability of error is given by

$$P(\text{error})=\int_{-\infty}^{\infty}P(\text{error},x)dx=\int_{-\infty}^{\infty}P(\text{error}|x)p(x)dx \quad \text{.... (5)}$$

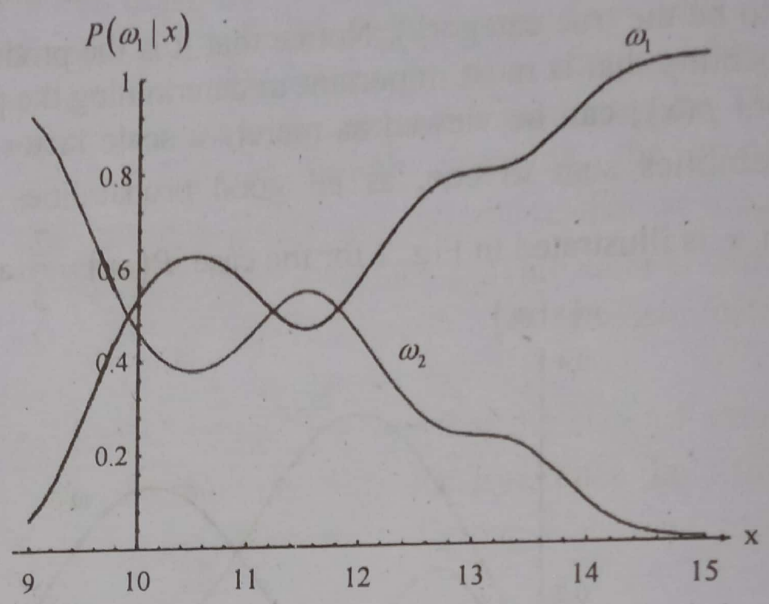


Fig: 2. Posterior probabilities for the particular priors $P(\omega_1)=2/3$ and $P(\omega_2)=1/3$ for the class-conditional probability densities shown in Fig. 1. Thus in this case, given that a pattern is measured to have feature value $x=14$, the probability it is in category ω_2 is roughly 0.08 and that it is in ω_1 is 0.92. At every x, the posteriors sum to 1.0

and if for every x we insure that $P(\text{error}|x)$ is as small as possible, then the integral must be as small as possible. Thus we have justified the following Bayes' *decision rule* for minimizing the probability of error:

Decide ω_1 if $P(\omega_1|x)>P(\omega_2|x)$; otherwise decide ω_2 .... (6)

and under this rule Eqn. (4) becomes

$$P(\text{error}|x)=\min\left[P(\omega_1|x),\ P(\omega_2|x)\right] \quad \text{.... (7)}$$

This form of the decision rule emphasizes the role of the posterior probabilities. By using Eqn. (1), we can instead express the rule in terms of the conditional and prior probabilities. First note that the *evidence*, $p(x)$, in Eqn. 1 is unimportant as far as making a decision is concerned. It is basically just a scale factor that states how frequently we will actually measure a pattern with feature value x; its presence in Eqn. (1) assures us

that $P(\omega_1|x)+P(\omega_2|x)=1$. By eliminating this scale factor, we obtain the following completely equivalent decision rule:

Decide ω_1 if $p(x|\omega_1)P(\omega_1)>p(x|\omega_2)P(\omega_2)$; otherwise decide ω_2 .... (8)

Some additional insight can be obtained by considering a few special cases. If for some x we have $p(x|\omega_1)=p(x|\omega_2)$, then that particular observation gives us no information about the state of nature; in this case, the decision hinges entirely on the prior probabilities. On the other hand, if $P(\omega_1)=P(\omega_2)$, then the states of nature are equally probable; in this case the decision is based entirely on the likelihoods $p(x|\omega_j)$. In general, both of these factors are important in making a decision and the Bayes decision rule combines them to achieve the minimum probability of error.

2. Explain classifier, discriminant functions and decision surface.

[MODEL QUESTION]

Answer:

The Multi-Category case

There are many different ways to represent pattern classifiers. One of the most useful is in terms of a set of *discriminant functions* $g_i(x), i=1,\ldots,c$. The classifier is said to assign a feature vector x to class ω_i if

$$g_i(x)>g_j(x) \quad \text{for all } j\neq i \qquad \ldots. (1)$$

Thus, the classifier is viewed as a network or machine that computers c discriminant functions and selects the category corresponding to the largest discriminant. A network representation of a classifier is illustrated in Fig. 1.

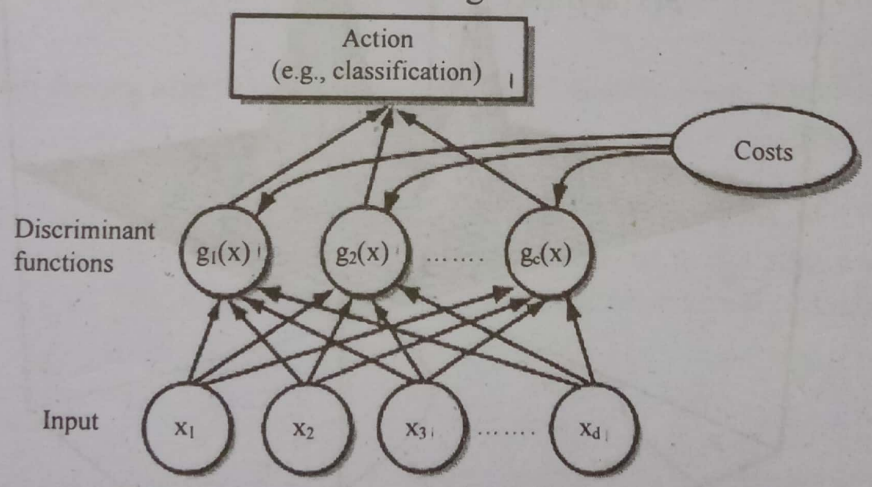


Fig: 1. The functional structure of a general statistical pattern classifier which includes d inputs and c discriminant functions $g_i(x)$. A subsequent step determines which of the discriminant values is the maximum and categorizes the input pattern accordingly. The arrows show the direction of the flow of information, though frequently the arrows are omitted when the direction of flow is self-evident

A Bayes classifier is easily and naturally represented in this way. For the general case with risks, we can let $g_i(x) = -R(\alpha_i \mid x)$, since the maximum discriminant function will then correspond to the minimum conditional risk. Fir the minimum-error-rate case, we can simplify things further by taking $g_i(x) = P(\omega_i \mid x)$, so that the maximum discriminant function corresponds to the maximum posterior probability.

Clearly, the choice of discriminant functions is not unique. We can always multiply all the discriminant functions by the same positive constant or shift them by the same additive constant without influencing the decision. More generally, if we replace every $g_i(x)$ by $f(g_i(x))$, where $f(\cdot)$ is a monotonically increasing function, the resulting classification is unchanged. This observation can lead to significant analytical and computational simplifications. In particular, for minimum-error-rate classification, any of the following choices gives identical classification results, but some can be much simpler to understand or to compute than others:

$$g_i(x) = P(\omega_i \mid x) = \frac{p(x \mid \omega_i) P(\omega_i)}{\sum_{j=1}^{c} p(x \mid \omega_j) P(\omega_j)} \quad \text{.... (2)}$$

$$g_i(x) = p(x \mid \omega_i) P(\omega_i) \quad \text{.... (3)}$$

$$g_i(x) = \ln p(x \mid \omega_i) + \ln P(\omega_i) \quad \text{.... (4)}$$

where ln denotes natural logarithm.

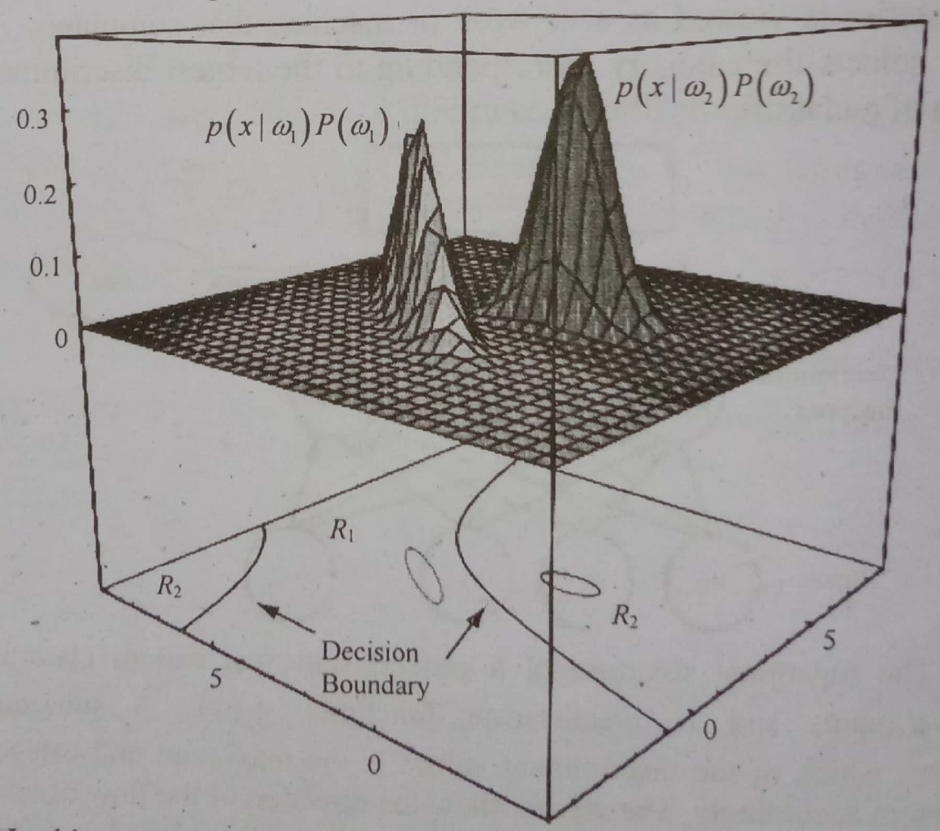


Fig: 2. In this two-dimensional two-category classifier, the probability densities are Gaussian (with $1/e$ ellipses shown), the decision boundary consists of two hyperbolas and thus the decision region R_2 is not simply connected

Even though the discriminant functions can be written in a variety of forms, the decision rules are equivalent. The effect of any decision rule is to divide the feature space into c *decision regions*, $R_1, \ldots, R_c$. If $g_i(x) > g_j(x)$ for all $j \neq i$, then x is in R_i and the decision rule calls for us to assign x to ω_i. The regions are separated by *decision boundaries*, surfaces in feature space where ties occur among the largest discriminant functions (Fig. 2).

The Two-Category Case:

While the two-category case is just a special instance of the multicategory case, it has traditionally received separate treatment. Indeed, a classifier that places a pattern in one of only two categories has a special name — a *dichotomizer*. Instead of using two discriminant functions g_1 and g_2 and assigning x to ω_1 if $g_1 > g_2$, it is more common to define a single discriminant function and to use the following decision rule: Decide ω_1 if $g(x) > 0$; otherwise decide ω_2.

$$g(x) \equiv g_1(x) - g_2(x) \qquad \ldots (5)$$

Thus, a dichotomizer can be viewed as a machine that computes a single discriminant function $g(x)$ and classifies x according to the algebraic sign of the result. Of the various forms in which the minimum-error-rate discriminant function can be written, the following two (derived from Eqns. (2) & (4) are particularly convenient:

$$g(x) = P(\omega_1 \mid x) - P(\omega_2 \mid x) \qquad \ldots (6)$$

$$g(x) = \ln \frac{p(x \mid \omega_1)}{p(x \mid \omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)} \qquad \ldots (7)$$

3. What do you know about normal density and discriminant function?
[MODEL QUESTION]

Answer:

Before talking about discriminant functions for the normal density, we first need to know that a normal distribution is and how it is represented for just a single variable and for a vector variable. Lets begin with the continuous univariate normal or Gaussian density.

$$f_x = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

for which the expected value of x is

$$\mu = \varepsilon[x] = \int_{-\infty}^{\infty} x p(x)\, dx$$

and where the expected squared deviation or variance is

$$\sigma^2 = \varepsilon\left[(x-\mu)^2\right] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)\, dx$$

The univariate normal density is completely specified by two parameters; its mean μ and variance σ^2. The function f_x can be writted as $N(\mu,\sigma)$ which says that x is distributed normally with mean μ and variance σ^2. Samples from normal distributions tend to cluster about the mean with a spread related to the standard deviation σ.
For the multivariate normal density in d dimensions, f_x is written as

$$f_x = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right]$$

where x is a d-component column vector, μ is the d-component mean vector, Σ is the d-by-d covariance matrix and $|\Sigma|$ and Σ^{-1} are its determinant and inverse respectively. Also, $(x-\mu)'$ denotes the transpose of $(x-\mu)$.

and $$\Sigma = \varepsilon\left[(x-\mu)(x-\mu)'\right] = \int(x-\mu)(x-\mu)'\,p(x)dx$$

where the expected value of a vector or a matrix is found by taking the expected value of the individual components, i.e., if x_i is the i th component of x, μ_i the i th component of μ and σ_{ij} the ij th component of Σ, then

$$\mu_i = \varepsilon[x_i]$$

and $$\sigma_{ij} = \varepsilon\left[(x_i-\mu_i)(x_j-\mu_j)\right]$$

The convariance matrix Σ is always symmetric and positive definite which means that the determinant of Σ is strictly positive. The diagonal elements σ_{ii} are the variances of the respective x_i (i.e., σ^2 and the off-diagonal elements σ_{ij} are the covariances of x_i and x_j. If x_i and x_j are statistically independent, then $\sigma_{ij}=0$. If all off-diagonal elements are zero, $p(x)$ reduces to the product of the univariate normal densities for the components of x.

Discriminant Functions:

Discriminant functions are used to find the minimum probability of error in decision making problems. In a problem with feature vector y and state of nature variable w, we can represent the discriminant function as:

$$g_i(Y) = \ln p(Y|w_i) + \ln P(w_i)$$ where, $p(Y|w_i)$

conditional probability density function for Y with w_i being the state of nature and $P(w_j)$ is the prior probability that nature is in state w_j. If we take $p(Y|w_i)$ as multivariate normal distributins. That is if $p(Y|w_i) = N(\mu,\sigma)$. Then the discriminant function changes to;

$$g_i(Y) = -\frac{\|x-\mu_i\|^2}{\sigma_i} + \ln P(w_i)$$

where, $\|.\|$ denotes the Euclidean norm, that is, in a problem with feature vector y and state of nature variable w, we can represent the discriminant function as:

$$g_i(x) = -\frac{1}{2}(x-\mu_i)' \Sigma_i^{-1}(x-\mu_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(w_i)$$

We will now look at the multiple cases for a multivariate normal distribution.

Case 1:

$$\Sigma_i = \sigma^2$$

This is the simplest case and it occurs when the features are statistically independent and each feature has the same variance, σ^2. Here, the covariance matrix is diagonal since its simply σ^2 times the identity matrix I. This means that each sample falls into equal sized clusters that are centered about their respective mean vactors. The computation of the determinant and the inverse $|\Sigma_i| = \sigma^{2d}$ and $\Sigma_i^{-1} = (1/\sigma^2)$. Because both $|\Sigma_i|$ and the $(d/2)\ln 2\pi$ term in the equation above are independent of i, we can ignore them and thus we obtain this simplified discriminant function:

$$g_i(x) = -\frac{\|x-\mu_i\|^2}{2\sigma^2} + \ln P(w_i)$$

where, $\|.\|$ denotes the Euclidean norm, that is, $\|x-\mu_i\|^2 = (x-\mu_i)'(x-\mu_i)$

If the prior probabilities are not equal, then the discriminant function shows that the squared distance $\|x-\mu\|^2$ must be normalized by the variance σ^2 and offset by adding $\ln P(w_i)$; therefore if x is equally near two different mean vectors, the optimal decision will favour the priori more likely. Expansion of the quadratic form $(x-\mu_i)'(x-\mu_i)$ yields:

$$g_i(x) = -\frac{1}{2\sigma^2}\left[x'x - 2\mu_i'x + \mu_i'\mu_i\right] + \ln P(w_i)$$

which looks like a quadratic function of x. However, the quadratic term $x'x$ is the same for all i, meaning it can be ignored since it just an additive constant, thereby we obtain the equivalent discriminant function:

$$g_i(x) = w_i'x + w_{i0}$$

where, $w_i = \frac{1}{\sigma^2}\mu_i$ and $w_{i0} = -\frac{1}{2\sigma^2}\mu_i'\mu_i + \ln P(w_i)$

w_{i0} is the threshold or bias for the i th category.

A classifier that uses linear discriminants is called a linear machine. For a linear machine, the decision surfaces for a linear machine are just pieces of hyperplanes defined by the linear equations $g_i(x) = g_j(x)$ for the two categories with the highest posterior probabilities. In this situation, the equation can be written as

$$w'(x - x_0) = 0$$

where, $w = \mu_i - \mu_j$ and $x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \ln \frac{P(w_i)}{P(w_j)} (\mu_i - \mu_j)$

The equations define a hyperplane through the point x_0 and orthogonal to the vector w. Because $w = \mu_i - \mu_j$, the hyperplane separating R_i and R_j is orthogonal to the line linking the means. If $P(w_i) = P(w_j)$, the point x_0 is halfway between the means and the hyperplane is the perpendicular bisector of the line between the means in Fig. 1 below. If $P(w_i) \neq P(w_j)$, the point x_0 shifts away from the more likely mean.

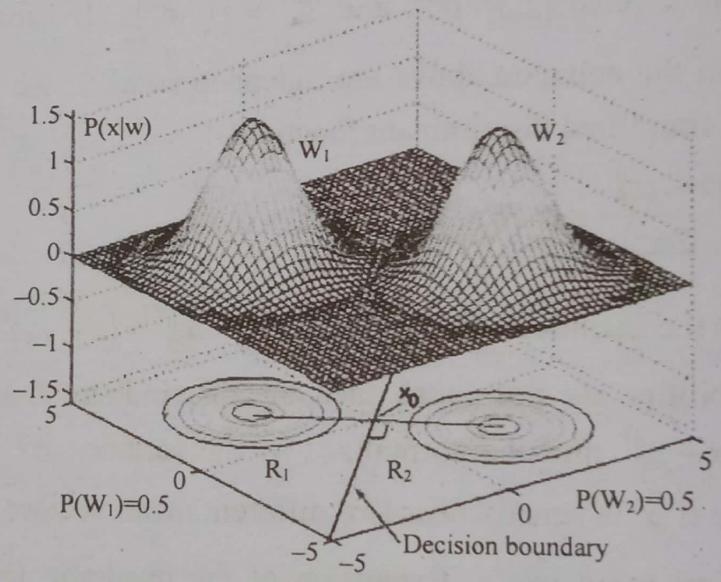


Fig: 1

Case 2:

$$\Sigma_i = \Sigma$$

Another case occurs when the covariance matrices for all the classes are identical. It corresponds to a situation where the samples fall into hyperellipsoidal clusters of equal size and shape, with the cluster of the i th class being centered around the mean vector μ_i. Both $|\Sigma_i|$ and the $(d/2)\ln 2\pi$ terms can also be ignored as done in the first step because they are independent of i. This leads to the simplified discriminant function:

$$g_i(x) = -\frac{1}{2}(x - \mu_i)' \Sigma_i^{-1} (x - \mu_i) + \ln P(w_i)$$

If the prior probabilities $P(w_i)$ are equal for all classes, then the $\ln P(w_i)$ term can be ignored, however, if they are unequal then the decision will be biased in favour of the

more likely priori. Expansion of the quadratic form $(x-\mu_i)'\Sigma^{-1}(x-\mu_i)$ results in a sum involving the term $x'\Sigma^{-1}x$ which is independent of i. After this term is dropped, we get the resulting linear discriminant function:

$$g_i(x) = w_i'x + w_{i0}$$

where $w_i = \Sigma^{-1}\mu_i$

and $$w_{i0} = -\frac{1}{2}\mu_i'\Sigma^{-1}\mu_i + \ln P(w_i)$$

because the discriminants are linear, the resulting decision boundaries are again hyperplanes. If R_i and R_j are very close, the boundary between them has the equation:

$$w'(x-x_0) = 0$$

where, $w = \Sigma^{-1}(\mu_i - \mu_j)$

and $$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln\left[P(w_i)/P(w_j)\right]}{(\mu_i-\mu_j)'\Sigma^{-1}(\mu_i-\mu_j)}(\mu_i - \mu_j)$$

Because $w = \sigma^{-1}(\mu_i - \mu_j)$ is generally not in the direction of $\mu_i - \mu_j$, the hyperplane separating R_i and R_j is generally not orthogonal to the line between the means. If the prior probabilities are equal, it intersects the line at point x_0 and then x_0 is halfway between the means. If the prior probabilities are not equal, the boundary hyperplane is shifted away from the more likely mean. Fig. 2 below shows what the boundary decision looks like for this case

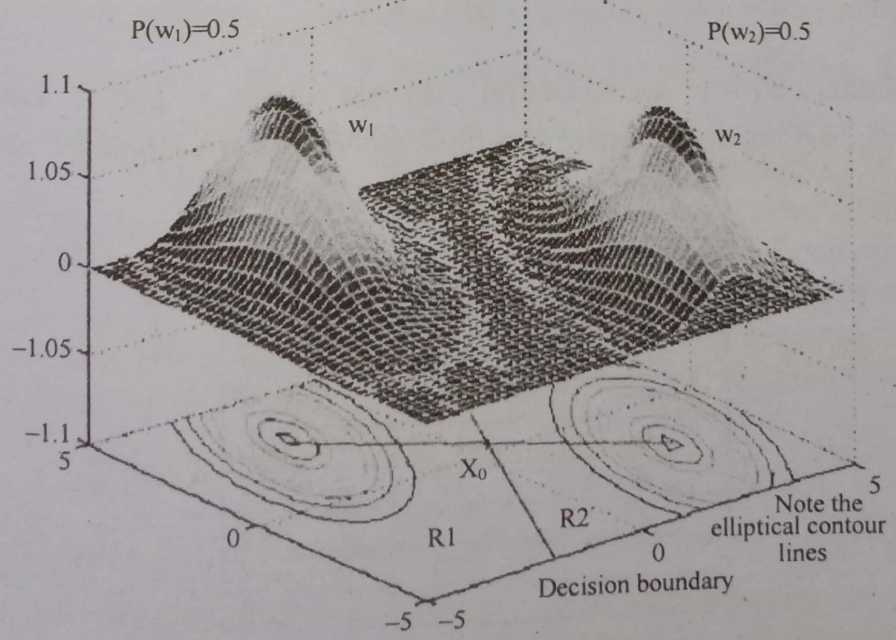


Fig: 2

Case 3:

$$\Sigma_i = \text{arbitrary}$$

In the general multivariate Gaussian case where the covariance matrices are different for each class, the only term that can be dropped from the initial discriminant function is the $(d/2)\ln 2\pi$ term. The resulting discriminant term is;

$$g_i(x) = x^t W_i x + w_i^t x + w_{i0}$$

where, $W_i = -\frac{1}{2}\Sigma_i^{-1}$

$$w_i = \Sigma_i^{-1}\mu_i \quad \text{and} \quad w_{i0} = -\frac{1}{2}\mu_i^t \Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(w_i)$$

This leads to hyperquadric decision boundaries as seen in the figure below:

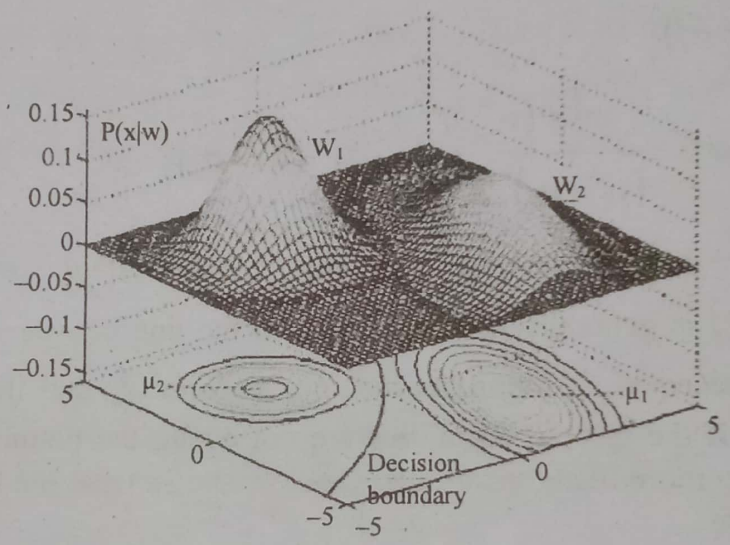


Fig: 3

Example:

Given the set of data below of a distribution with two classes w_1 and w_2 both with prior probability of 0.5, find the discriminant functions and decision boundary.

Sample	w_1	w_2
1	−5.01	−0.91
2	−5.43	1.30
3	1.08	−7.75
4	0.86	−5.47
5	−2.67	6.14
6	4.94	3.60
7	−2.51	5.37
8	−2.25	7.18
9	5.56	−7.39
10	1.03	−7.50

From the data given above we know to use the equations from case 1 since all points in each class have the same variance, therefore, the means are:

$$\mu_1 = \sum_{k=1}^{10} x_1 = -0.44$$

$$\mu_2 = \sum_{k=1}^{10} x_2 = -0.543$$

and the variances are

$$\sigma_1^2 = \sum_{k=1}^{10} x_1 - \mu_1 = 31.34$$

$$\sigma_2^2 = \sum_{k=1}^{10} x_2 - \mu_2 = 52.62$$

The discriminant functions are then

$$g_1 = \frac{-0.44}{31.34} - \frac{-0.44^2}{2*31.34} + \ln(0.5) = -0.710$$

$$g_2 = \frac{-0.543}{52.62} - \frac{-0.543^2}{2*52.62} + \ln(0.5) = -0.706$$

and the decision boundary x_0 is going to be halfway between the means at 0.492 because they have the same prior probability.

4. What do you understand by continuous and discrete features of Bayes decision theory? [MODEL QUESTION]

Answer:

Continuous feature

Allowing the use of more than one feature just means that we would replace the scaler y with the *feature vector* Y, where Y is in a d-dimensional Euclidean space R^d, called the **feature space**. Allowing more than two states of nature provides a useful generalization with small notational expense. Allowing more actions also opens up the possibility of rejection i.e., refusing to make a decision in too close cases. This is can be very useful if being indecisive is not too costly. The Loss function states exactly how costly each chosen action is and is used to convert a probability determination into a decision. Cost functions enables us to look at situations where certain errors are more costly than others, although we will often only be looking at cases where all errors are equally costly.

Putting this together, let $\{x_1, \ldots, x_c\}$ be the finite set of c states of nature and let $\{k_1, \ldots, k_a\}$ be the finite set of a possible actions. The loss function $\lambda(k_i \mid k_j)$ describes the loss incurred for taking action k_i when the state of nature is x_j. Let Y be a d-

component-vector-valued-RV and let $p(Y|x_j)$ be the conditional probability density function for Y with x_j being the true state of nature. As discussed before, $P(x_j)$ is the prior probability that nature is in state x_j, therefore by using Bayes formula we can find the posterior probability $P(x_j|Y)$:

$$P(x_j|Y)=\frac{p(Y|x_j)P(x_j)}{P(Y)} \qquad \text{.... (1)}$$

where, $$P(Y)=\sum_{j=1}^{c} p(Y|x_j)P(x_j) \qquad \text{.... (2)}$$

Now, suppose we observe a particular feature space Y and we decide to take an action k_i. If the state of nature is x_j, then from the definition of the loss function above we will incur the loss $\lambda(k_i|x_j)$. Because $P(x_j|Y)$ is the probability that the true state of nature is x_j, the loss associated with taking action k_i can be expressed as:

$$R(k_i|Y)=\sum_{j=1}^{c} \lambda(k_i|x_j)P(x_j|Y) \qquad \text{.... (3)}$$

In decision theory terminology, an expected loss is called a risk and $R(k_i|Y)$ is called the **conditional risk**. So whenever we have an observation Y, we can minimize the expected loss by choosing the action that minimized the conditional risk. To minimize the overall risk, compute the compute the conditional risk in Eqn. (3), for $i=1,\ldots,a$ and then select the action k_i for which $R(k_i|Y)$ is minimum. The resulting minimum risk is also called the Bayes risk and is denoted by R^*.

Discrete Features:

In many practical applications, the components of the feature vectors are binary, ternary or higher integer values so that Y can assume one of m discrete values $\{v_1,\ldots,v_m\}$. In these cases, the probability density functions become sums of the form

$$\sum_x P(Y|x_j) \qquad \text{.... (4)}$$

where we understand that the summation is overall values of x in the discrete distributions. Bayes formula then involves probabilities, rather than probability densities. So we have:

$$P(x_j \mid Y) = \frac{P(Y \mid x_j) P(x_j)}{P(Y)} \qquad \text{.... (5)}$$

where, $$P(Y) = \sum_{j=1}^{c} P(Y \mid x_j) P(x_j) \qquad \text{.... (6)}$$

However, the definition of the conditional risk $R(k \mid Y)$ is the same and the aim to minimize the overall risk for a selected action remains the same in the scenario.

PARAMETERS ESTIMATION METHODS

Multiple Choice Type Questions

[MODEL QUESTION]

1. The maximum likelihood estimate is
   a) minimum of α not necessarily in the parameter space
   b) maximum of α in the parameter space
   c) maximum of α not necessarily in the parameter space
   d) minimum of α in the parameter space

Answer: (b)

Short Answer Type Questions

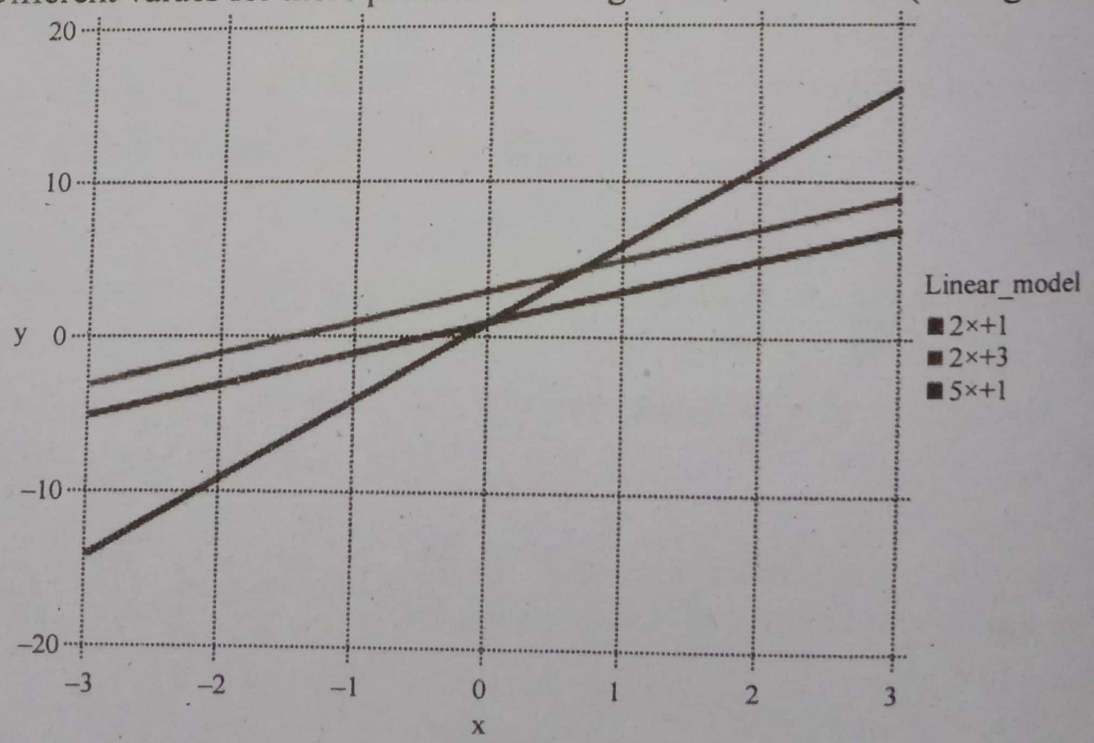
[MODEL QUESTION]

1. What are parameters?

Answer:

Often in machine learning we use a model to describe the process that results in the data that are observed. For example, we may use a random forest model to classify whether customers may cancel a subscription from a service (known as churn modelling) or we may use a linear model to predict the revenue that will be generated for a company depending on how much they may spend on advertising (this would be an example of linear regression). Each model contains its own set of parameters that ultimately defines what the model looks like.

For a linear model we can write this as $y = mx + c$. In this example x could represent the advertising spend and y might be the revenue generated. m and c are parameters for this model. Different values for these parameters will give different lines (see figure below).



Three linear models with different parameter values

So parameters define a blueprint for the model. It is only when specific values are chosen for the parameters that we get an instantiation for the model that describes a given phenomenon.

2. What is maximum likelihood estimation? **[MODEL QUESTION]**

Answer:

In statistics, **maximum likelihood estimation** (**MLE**) is a method of estimating the parameters of an assumed probability distribution, given some observed data. This is achieved by maximizing a likelihood function so that, under the assumed statistical model, the observed data is most probable. The point in the parameter space that maximizes the likelihood function is called the maximum likelihood estimate. The logic of maximum likelihood is both intuitive and flexible, and as such the method has become a dominant means of statistical inference.

If the likelihood function is differentiable, the derivative test for determining maxima can be applied. In some cases, the first-order conditions of the likelihood function can be solved explicitly; for instance, the ordinary least squares estimator maximizes the likelihood of the linear regression model. Under most circumstances, however, numerical methods will be necessary to find the maximum of the likelihood function. The likelihood function can also be non-convex with multiple local minima requiring the use of heuristic global optimization techniques.

3. What is Expectation- Maximization method? **[MODEL QUESTION]**

Answer:

In most of the real-life problem statements of **Machine learning**, it is very common that we have many relevant features available to build our model but only a small portion of them are observable. Since we do not have the values for the not observed (latent) variables, the **Expectation-Maximization** algorithm tries to use the existing data to determine the optimum values for these variables and then finds the model parameters.

Long Answer Type Questions

1. Explain maximum likelihood estimation. **[MODEL QUESTION]**

Answer:

There are many methods for estimating unknown parameters from data. We will first consider the Maximum Likelihood Estimate (MLE), which answers the question:

For which parameter value does the observed data have the biggest probability?

The MLE is an example of a point estimate because it gives a single value for the unknown parameter (later our estimates will involve intervals and probabilities). Two advantages of the MLE are that it is often easy to compute and that it agrees with our intuition in simple examples. We will explain the MLE through a series of examples.

Example 1: A coin is flipped 100 times. Given that there were 55 heads, find the maximum likelihood estimate for the probability p of heads on a single toss.

Before actually solving the problem, let's establish some notation and terms.

We can think of counting the number of heads in 100 tosses as an experiment. For a given value of p, the probability of getting 55 heads in this experiment is the binomial probability

$$P(55 \text{ heads}) = \binom{100}{55} p^{55} (1-p)^{45}$$

The probability of getting 55 heads depends on the value of p, so let's include p in by using the notation of conditional probability:

$$P(55 \text{ heads} \mid p) = \binom{100}{55} p^{55} (1-p)^{45}$$

You should read $P(55 \text{ heads} \mid p)$ as:
'the probability of 55 heads given p.'
or, more precisely as
'the probability of 55 heads given that the probability of heads on a single toss is p'.
Here are some standard teerms we will use as we do statistics.

- **Experiment:** Flip the coin 100 times and count the number of heads.
- **Data:**The data is the result of the experiment. In this case it is '55 heads'.
- **Parameter(s) of interest:** We are interested in the value of the unknown parameter p.
- **Likelihood, or likelihood function:** This is $P(\text{data} \mid p)$. Note it is a function of both the data and the parameter p. In this case the likelihood is

$$P(55 \text{ heads} \mid p) = \binom{100}{55} p^{55} (1-p)^{45}$$

2. Look carefully at the definition. One typical source of confusion is to mistake the likelihood $P(\text{data} \mid p)$ for $P(p \mid \text{data})$. We know from our earlier work with Bayes' theorem that $P(\text{data} \mid p)$ and $P(p \mid \text{data})$ are usually very different.

Definition: Given data the maximum likelihood estimate (MLE) for the parameter p is the value of p that maximizes the likelihood $P(\text{data} \mid p)$. That is, the MLE is the value of p for which the data is most likely.

Answer: For the problem at hand, we saw above that the likelihood

$$P(55 \text{ heads} \mid p) = \binom{100}{55} p^{55} (1-p)^{45}$$

We'll use the notation $\hat{p}$ for the MLE. We use calculus to find it by taking the derivative of the likelihood function and setting it to 0.

$$\frac{d}{dp} P(\text{data} \mid p) = \binom{100}{55} \left(55p^{54}(1-p)^{45} - 45p^{55}(1-p)^{44}\right) = 0$$

Solving this for p we get

$$55p^{54}(1-p)^{45} = 45p^{55}(1-p)^{44}$$
$$55(1-p) = 45p$$
$$55 = 100p$$

The MLE is $\hat{p} = 0.55$

Note:

1. The MLE for p terned out to be exactly the fraction of heads we saw in our data.
2. The MLE is computed from the data. That is, it is a statistic.
3. Officially you should check that the critical point is indeed a maximum. You can do this with the second derivative test.

Log likelihood:

If is often easier to work with the natural log of the likelihood function. For short this is simply called the **log likelihood**. Since $\ln(x)$ is an increasing function, the maxima of the likelihood and log likelihood coincide.

Example 2. Redo the previous example using log likelihood.

Answer: We had the likelihood $P(55 \text{ heads} \mid p) = \binom{100}{55} p^{55}(1-p)^{45}$. Therefore the log likelihood is

$$\ln\left(P(55 \text{ heads} \mid p)\right) = \ln\left(\binom{100}{55}\right) + 55\ln(p) + 45\ln(1-p)$$

Maximizing likelihood is the same as maximizing log likelihood. We check that calculus gives us the same answer as before:

$$\frac{d}{dp}(\text{log likelihood}) = \frac{d}{dp}\left[\ln\left(\binom{100}{55}\right) + 55\ln(p) + 45\ln(1-p)\right]$$

$\Rightarrow \quad 55(1-p) = 45p$

$\Rightarrow \quad \hat{p} = 0.55$

Maximum likelihood for continuous distributions

For continuous distributions, we use the probability density function to define the likelihood. We show this in a few exmples. In the next section we explain how this is analogous to what we did in the discrete case.

Example 3. Light bulbs

Suppose that the lifetime of *Badger* brand light bulbs is modeled by an exponential distribution with (unknown) parameter λ. We test 5 bulbs and find they have lifetimes of 2, 3, 1, 3 and 4 years, respectively. what is the MLE for λ?

Answer: We need to be careful with our notation. With five different values it is best to use subscripts. Let X_j be the lifetime of the ith bulb and let x_i be the value X_i takes. Then each X_i has pdf $f_{X_i}(x_i) = \lambda e^{-\lambda x_i}$. We assume the lifetimes of the bulbs are independent, so the joint pdf is the product of the individual densities:

$$f(x_1, x_2, x_3, x_4, x_5 \mid \lambda) = \left(\lambda e^{-\lambda x_1}\right)\left(\lambda e^{-\lambda x_2}\right)\left(\lambda e^{-\lambda x_3}\right)\left(\lambda e^{-\lambda x_4}\right)\left(\lambda e^{-\lambda x_5}\right) = \lambda^5 e^{-\lambda(x_1+x_2+x_3+x_4+x_5)}$$

Note that we write this as a conditional density, since it depends on λ. Viewing the data as fixed and λ a svariable, this density is the likelihood function. Our data had values

$$x_1 = 2,\ x_2 = 3,\ x_3 = 1,\ x_4 = 3,\ x_5 = 4$$

So, the likelihood and log likelihood functions with this data are

$$f(2,3,1,3,4 \mid \lambda) = \lambda^5 e^{-13\lambda},\ \ \ln f(2,3,1,3,4 \mid \lambda) = 5\ln(\lambda) - 13\lambda$$

Finally we use calculus to find the MLE:

$$\frac{d}{d\lambda}(\text{log likelihood}) = \frac{5}{\lambda} - 13 = 0 \quad \Rightarrow \quad \boxed{\hat{\lambda} = \frac{5}{13}}$$

Example 4. Normal distributions

Suppose the data x $x_1, x_2, \ldots, x_n$ is drawn from a $N(\mu, \sigma^2)$ distribution, where μ and σ are unknown. Find the maximum likelihood estimate for the pair (μ, σ^2).

Answer: Let's be precise and phrase this in terms of random variables and densities. Let uppercase $X_1, \ldots, X_n$ be i.i.d. $N(\mu, \sigma^2)$ random variables and let lowercase x_i be the value X_i takes. The density for each X_i is

$$f_{X_i}(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

Sihce the X_i are independent their joint pdf is the product of the individual pdf's:

$$f(x_1, \ldots, x_n \mid \mu, \sigma) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\sum_{i=1}^n \frac{(x_i-\mu)^2}{2\sigma^2}}$$

For the fixed data $x_1, \ldots, x_n$, the likelihood and log likelihood are

$$f(x_1, \ldots, x_n \mid \mu, \sigma) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\sum_{i=1}^n \frac{(x_i-\mu)^2}{2\sigma^2}},$$

$$\ln\left(f(x_1, \ldots, x_n \mid \mu, \sigma)\right) = -n\ln\left(\sqrt{2\pi}\right) - n\ln(\sigma) - \sum_{i=1}^n \frac{(x_i-\mu)^2}{2\sigma^2}$$

Since $\ln(f(x_1,\ldots,x_n \mid \mu,\sigma))$ is a function of the two variables μ,σ we use partial derivatives to find the MLE. The easy value to find is $\hat{\mu}$:

$$\frac{\partial f(x_1,\ldots,x_n \mid \mu,\sigma)}{\partial \mu} = \sum_{i=1}^{n} \frac{(x_i-\mu)}{\sigma^2} = 0 \Rightarrow \sum_{i=1}^{n} x_i = n\mu \Rightarrow \hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n} = \bar{x}$$

To find $\hat{\sigma}$ we differentiate and solve for σ:

$$\frac{\partial f(x_1,\ldots,x_n \mid \mu,\sigma)}{\partial \sigma} = -\frac{n}{\sigma} + \sum_{i=1}^{n} \frac{(x_i-\mu)^2}{\sigma^3} = 0 \Rightarrow \hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}$$

We already know $\hat{\mu} = \bar{x}$, so we use that as the value for μ in the formula for $\hat{\sigma}$. We get the maximum likelihood estimates

$$\hat{\mu} = \bar{x} \qquad = \text{the mean of the data}$$

$$\hat{\sigma}^2 = \sum_{i=1}^{n} \frac{1}{n}(x_i-\hat{\mu})^2 = \sum_{i=1}^{n} \frac{1}{n}(x_i-\bar{x})^2 = \text{the variance of the data.}$$

Example 5. Uniform distributions

Suppose our data $x_1,\ldots,x_n$ are independently drawn from a uniform distribution $U(a,b)$. Find the MLE estimate for a and b.

Answer: This example is different from the previous ones in that we won't use calculus to find the MLE. The density for $U(a,b)$ is $\frac{1}{b-a}$ on $[a,b]$. Therefore our likelihood function is

$$f(x_1,\ldots,x_n \mid a,b) = \begin{cases} \left(\frac{1}{b-a}\right)^n & \text{if all } x_i \text{ are in the interval } [a,b] \\ 0 & \text{otherwise} \end{cases}$$

This is maximized by making $b-a$ as small as possible. The only restriction is that the interval $[a,b]$ must include all the data. Thus the MLE for the pair (a,b) is

$$\hat{a} = \min(x_1,\ldots,x_n) \qquad \hat{b} = \max(x_1,\ldots,x_n)$$

2. Why we use the density to find the MLE for continuous distributions?

[MODEL QUESTION]

Answer:

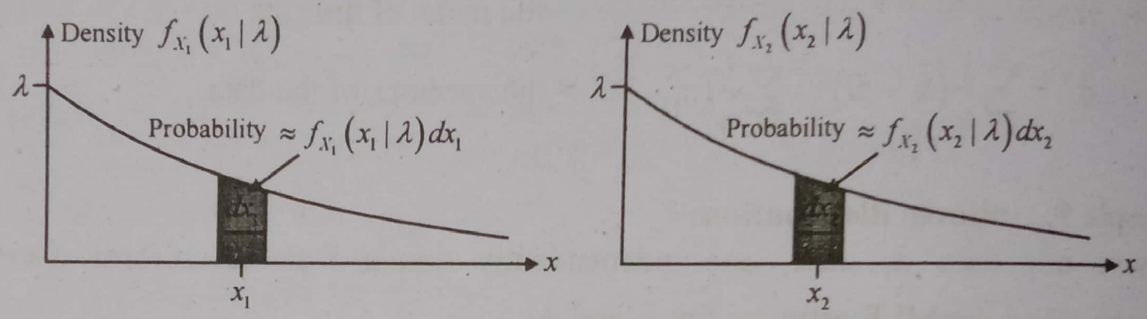
The idea for the maximum likelihood estimate is to find the value of the parameter(s) for which the data has the highest probability. In this section we'll see that we're doing this is really what we are doing with the densities. We will do this by considering a smaller version of the light bulb example.

Example 1. Suppose we have two light bulbs whose lifetimes follow an exponential (λ) distribution. Suppose also that we independently measure their lifetimes and get data

$x_1 = 2$ years and $x_2 = 3$ years. Find the value of λ that maximizes the probability of this data.

Answer: The main paradox to deal with is that for a continuous distribution the probability of a single value, say $x_1 = 2$, is zero. We resolve this paradox by remembering that a single measurement really means a range of values, e.g., in this example we might check the light bulb once a day. So the data $x_1 = 2$ years really means x_1 is somewhere in a range of 1 day around 2 years.

If the range is small we call it dx_1. The probability that X_1 is in the range is approximated by $f_{X_1}(x_1 \mid \lambda)dx_1$. This is illustrated in the figure below. The data value x_2 is treated in exactly the same way.



The usual relationship between density and probability for small ranges

Since the data is collected independently the joint probability is the product of the individual probabilities. Stated carefully

$$P(X_1 \text{ in range}, X_2 \text{ in range} \mid \lambda) \approx f_{X_1}(x_1 \mid \lambda)dx_1 \cdot f_{X_2}(x_2 \mid \lambda)dx_2$$

Finally, using the values $x_1 = 2$ and $x_2 = 3$ and the formula for an exponential pdf we have

$$P(X_1 \text{ in range}, X_2 \text{ in range} \mid \lambda) \approx \lambda e^{-2\lambda}dx_1 \cdot \lambda e^{-3\lambda}dx_2 = \lambda^2 e^{-5\lambda}dx_1 dx_2$$

Now that we have a genuine probability we can look for the value of λ that maximizes it. Looking at the formula above we see that the factor $dx_1 dx_2$ will play no role in finding the maximum. So for the MLE we drop it and simply call the density the likelihood:

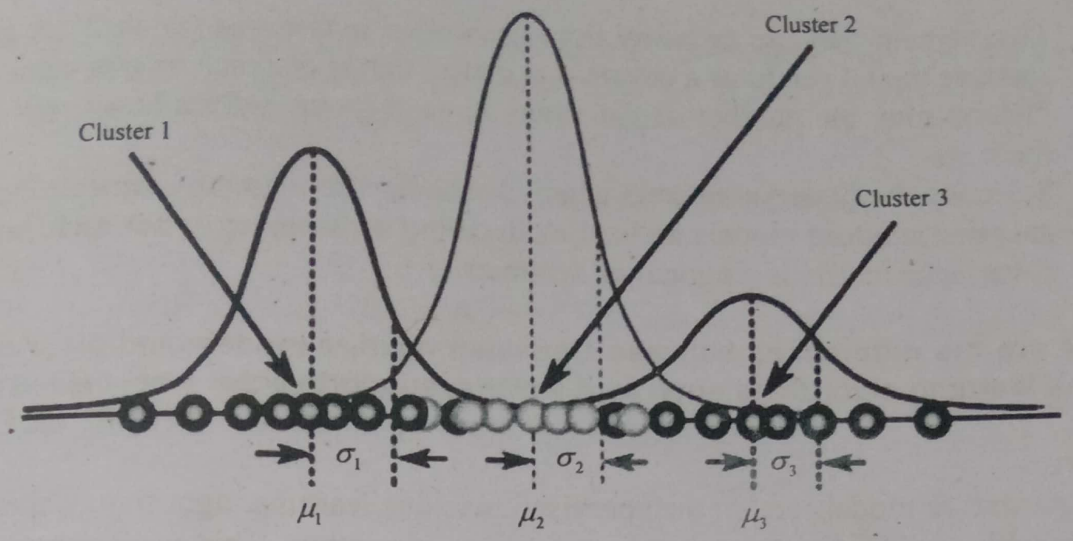
$$\text{likelihood} = f(x_1, x_2 \mid \lambda) = \lambda^2 e^{-5\lambda}$$

The value of λ that maximizes this is found just like in the example above. It is $\hat{\lambda} = 2/5$.

3. What is Gaussian mixture model (GMM)? **[MODEL QUESTION]**

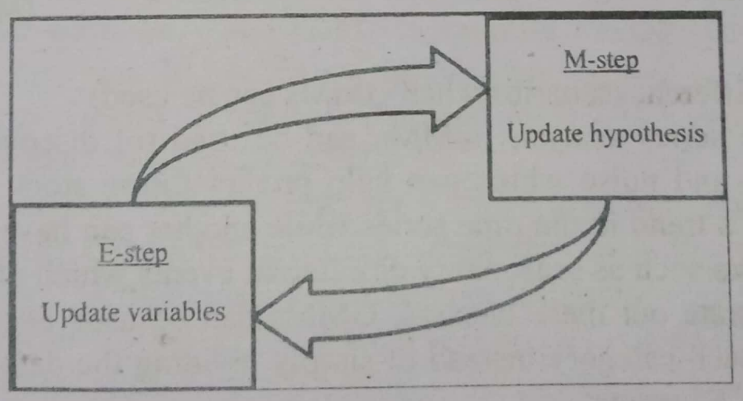
Answer:

The Gaussian mixture model is defined as a clustering algorithm that is used to discover the underlying groups of data. It can be understood as a probabilistic model where Gaussian distributions are assumed for each group and they have means and covariances which define their parameters. GMM consists of two parts – mean vectors (μ) & covariance matrices (Σ). A Gaussian distribution is defined as a continuous probability distribution that takes on a bell-shaped curve. Another name for Gaussian distribution is the normal distribution. Here is a picture of Gaussian mixture models:



4. What is expectation-maximization (EM) method in relation to GMM?
[MODEL QUESTION]

Answer:
In Gaussian mixture models, expectation-maximization method is used to find the gaussian mixture model parameters. Expectation is termed as E and maximization is termed as M. Expectation is used to find the gaussian parameters which are used to represent each component of gaussian mixture models. Maximization is termed as M and it is involved in determining whether new data points can be added or not.
Expectation-maximization method is a two-step iterative algorithm that alternates between performing an expectation step, in which we compute expectations for each data point using current parameter estimates and then maximize these to produce new gaussian, followed by a maximization step where we update our gaussian means based on the maximum likelihood estimate. This iterative process is performed until the gaussians parameters converge. Here is a picture representing the two-step iterative aspect of algorithm:



5. What are the key steps of using Gaussian mixture models?
[MODEL QUESTION]

Answer:
The following are three different steps to using gaussian mixture models:

- Determining a covariance matrix that defines how each Gaussian is related to one another. The more similar two Gaussians are, the closer their means will be and

vice versa if they are far away from each other in terms of similarity. A gaussian mixture model can have a covariance matrix that is diagonal or symmetric.

- Determining the number of gaussians in each group defines how many clusters there are.
- Selecting the hyperparameters which define how to optimally separate data using gaussian mixture models as well as deciding on whether or not each gaussian's covariance matrix is diagonal or symmetric.

6. What are the differences between Gaussian mixture models and other types of machine learning algorithms such as K-means, support vector machines (SVM)? [MODEL QUESTION]

Answer:
Gaussian mixture models are an unsupervised machine learning algorithm, while support vector machines (SVM) is a supervised learning algorithm. This means that gaussian mixture models can be used when there is no labeled data, however, the opposite requires labelled dataset for training the SVM models.
The Gaussian mixture model is different from K-means because Gaussian mixture models discover the underlying groups of data which is different than simply trying to divide data into different parts. Another difference is that Gaussian mixture models provide a probability for each category which can be used to make more accurate decisions and predictions about the data at hand. In addition, Gaussian mixture models have a higher chance of finding the right number of clusters in the data compared to k-means.
Gaussian mixture models have been found to outperform other machine learning algorithms such as artificial neural networks (ANN) when it comes to separating volatility from trend and noise.

7. What are the scenarios when gaussian mixture models can be used? [MODEL QUESTION]

Answer:
The following are different scenarios when GMMs can be used:

- In case of time series analysis, GMMs can be used tol discover how volatility is related to trend and noise which can help predict future stock prices. One cluster could consist of a trend in the time series while another can have noise and volatility from other factors such as seasonality or external events which affect the stock price. In order to separate out these clusters, GMMs can be used because they provide a probability for each category instead of simply dividing the data into two parts such as that in case of K-means.
- Another example is when there are different groups in a dataset and it's hard to label them as belonging to one group or another which makes it difficult for other machine learning algorithms such as K-means clustering algorithm to separate out the data. GMMs can be used in this case because they find gaussian mixture models that best describe each group and provide a probability for each cluster which is helpful when labeling clusters.

- Another example where gaussian mixture model can be useful is when it is desired to discover underlying groups of categories such as types of cancer or risk factors associated with different types of cancer.

8. What are some real-world examples where Gaussian mixture models can be used? **[MODEL QUESTION]**

Answer:

There are many different real-world problems that can be solved with Gaussian mixture models. Gaussian mixture models are very useful when there are large datasets and it is difficult to find clusters. This is where Gaussian mixture models help. It is able to find clusters of Gaussians more efficiently than other clustering algorithms such as k-means.

Here are some real-world problems which can be solved using Gaussian mixture models:

- **Finding patterns in medical datasets**: GMMs can be used for segmenting images into multiple categories based on their content or finding specific patterns in medical datasets.
- **Modeling natural phenomena**: GMM can be used to model natural phenomena where it has been found that noise follows Gaussian distributions. This model of probabilistic modeling relies on the assumption that there exists some underlying continuum of unobserved entities or attributes and that each member is associated with measurements taken at equidistant points in multiple observation sessions.
- **Customer behaviour analysis**: GMMs can be used for performing customer behaviour analysis in marketing to make predictions about future purchases based on historical data.
- **Stock price prediction**: Another area Gaussian mixture models are used is in finance where they can be applied to a stock's price time series. GMMs can be used to detect changepoints in time series data and help find turning points of stock prices or other market movements that are otherwise difficult to spot due to volatility and noise.
- **Gene expression data analysis**: Gaussian mixture models can be used for gene expression data analysis. In particular, GMMs can be used to detect differentially expressed genes between two conditions and identify which genes might contribute towards a certain phenotype or disease state.

Gaussian mixture models are a type of machine learning algorithm that is commonly used in data science. They can be applied to different scenarios, including when there are large datasets and it's difficult to find clusters or groups of Gaussian. Gaussian mixture models provide probability estimates for each cluster which allows you to label the clusters with less effort than k-means clustering algorithms would require. GMMs also offer some other benefits such as finding Gaussian mixture models that best describe each group, helping identify underlying categories of data sets, and predicting future stock prices more accurately by considering volatility and noise factors. If you're looking for an efficient way to find patterns within complicated datasets or need help modeling natural phenomena like natural disasters or customer behaviour analysis in your marketing, Gaussian mixture models could be the right choice.

9. Explain Expectation-maximization method. **[MODEL QUESTION]**

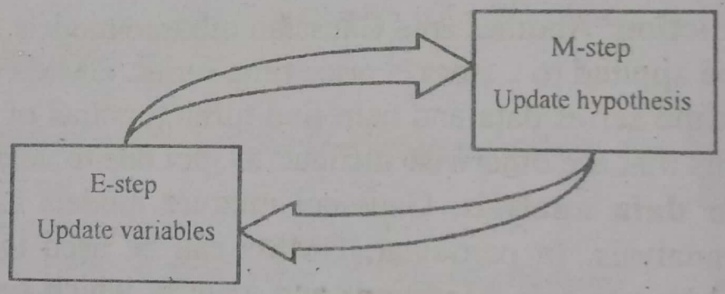
Answer:

A latent variable model consists of **observable** variables along with **unobservable** variables. Observed variables are those variables in the dataset that can be measured whereas unobserved (latent/hidden) variables are inferred from the observed variables.

- It can be used to find the **local maximum likelihood (MLE)** parameters or maximum a posteriori (MAP) parameters for latent variables in a statistical or mathematical model.
- It is used to predict theses missing values in the dataset, provided we know the general form of probability distribution associated with these latent variables.
- In simple words, the basic idea behind this algorithm is to use the observable samples of latent variables to predict the values of samples that are unobservable for learning. This process is repeated until the convergence of the values occurs.

Here is the algorithm you have to follow:

- Given a set of incomplete data, start with a set of initialized parameters.
- **Expectation step (E-step):** In this expectation step, by using the observed available data of the dataset, we can try to estimate or guess the values of the missing data. Finally, after this step, we get complete data having no missing values.
- **Maximization step (M-step):** Now, we have to use the complete data, which is prepared in the expectation step and update the parameters.
- Repeat step 2 and step 3 until we converge to our solution.

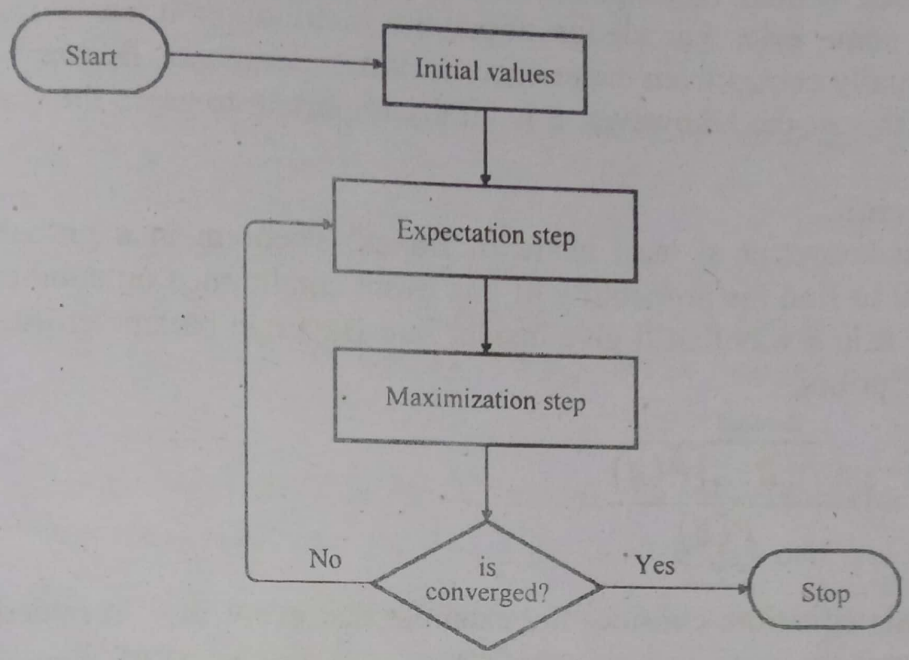


The Expectation-Maximization algorithm aims to use the available observed data of the dataset to estimate the missing data of the latent variables and then using that data to update the values of the parameters in the miximization step.

Let us understand the EM algorithm in a detailed manner:

- **Initialization step:** In this step, we initialized the parameter values with a set of initial values, then give the set of incomplete observed data to the system with the assumption that the observed data comes from a specific model i.e., **probability distribution.**
- **Expectation step:** In this step, by using the observed data to estimate or guess the values of the missing or incomplete data. It is used to update the variables.

- **Maximization step:** In this step, we use the complete data generated in the "**Expectation**" step to update the values of the parameters i.e., **update the hypothesis.**
- **Checking of convergence step:** How, in this step, we checked whether the values are converging or not, if yes, then stop otherwise repeat these two steps i.e., the "**Expectation**" step and "**Maximization**" step until the convergence occurs.



Advantages:

- The basic two steps of the EM algorithm i.e., E-step and M-step are often pretty easy for many of the machine learning problems in terms of implementation.
- The solution to the M-steps often exists in the closed-form.
- It is always guaranteed that the value of likelihood will increase after each iteration.

Disadvantages:

- It has **slow convergence.**
- It converges to the **local optimum** only.
- It takes both forward and backward probabilitie into account. This thing is in contrast to that of numerical optimization which considers only **forward probabilities.**

The latent variable model has several real-life applications in machine learning:

- Used to calculate the **Gaussian density** of a function.
- Helpful to fill in the **missing data** during a sample.
- It finds plenty of use in different domains such as **Natural Language Processing (NLP), Computer Vision**, etc.
- Used in image reconstruction in the field of **Medicine and Structural Engineering.**

- Used for estimating the parameters of the **Hidden Markov Model (HMM)** and also for some other mixed models like **Gaussian Mixture Models**, etc.
- Used for finding the values of latent variables.

10. Explain Bayesian estimation. **[MODEL QUESTION]**

Answer:

The central idea behind Bayesian estimation is that before we've seen any data, we already have some prior knowledge about the distribution it came from. Such prior knowledge usually comes from experience or past experiments. Before jumping into the nitty gritty of this method, however, it is virally important to grasp the concept of Bayes' Theorem.

Bayes' Theorem:

Hopefully you know, or at least heard of Bayes' Theorem in a probabilistic context, where we wish to find the probability of one event conditioned on another event. Here, I hope to frame it in a way that'll give insight into Bayesian parameter estimation and the significance of priors.

$$\overbrace{P(A|B)}^{\text{posterior}} = \frac{\overbrace{P(B|A)}^{\text{likelihood}}\overbrace{P(A)}^{\text{prior}}}{\underbrace{P(B)}_{\text{evidence}}}$$

To illustrate this equation, consider the example that event $A =$ "it rained earlier today" and event $B =$ "the grass is wet" and we wish to calculate $P(A|B)$, the probability that it rained earlier given that the grass is wet. To do this, we must calculate $P(B|A)$, $P(B)$ and $P(A)$. The conditional probability $P(B|A)$ represents the probability that the grass is wet given that it rained. In other words, it is the **likelihood** that the grass would be wet, given it is the case that it rained.

The value of $P(A)$ is known as the **prior**: the probability that it rained regardless of whether the grass is wet or not (prior to knowing the state of the grass). This prior knowledge is key because it determines how strongly we weight the likelihood. If we are somewhere that doesn't rain often, we would be more inclined to attribute wet grass to something other than rain such as dew or sprinklers, which is captured by a low $P(A)$ value. However, if we were somewhere that constantly rains, it is more probable that wet grass is a byproduct of the rain and a high $P(A)$ will reflect that.

All that's left is $P(B)$, also known as the **evidence**: the probability that the grass is wet, this event acting as the evidence for the fact that it rained. An important property of this value is that it's a normalizing constant for the final probability and as you will soon see in Bayesian estimation, we substitute a normalizing factor in place of the traditional "evidence".

The equation used for Bayesian estimation takes on the same form as Bayes' theorem, the key difference being that we are now using models and probability density functions (pdfs) in place of numerical probabilities.

$$\overbrace{p(\theta|D)}^{\text{posterior distribution}} = \frac{\overbrace{p(D|\theta)}^{\text{likelihood function}} \; \overbrace{p(\theta)}^{\text{prior distribution}}}{\underbrace{\int p(D|\theta)\,p(\theta)\,d\theta}_{\text{evidence}}}$$

Bayesian Estimation

Notice that first, the likelihood is equivalent to the likelihood used in MLE and second, the evidence typically used in Bayes' Theorem (which in this case would translate to $P(D)$), is replaced with an integral of the numerator. This is because (1) $P(D)$ is extremely difficult to actually calculate, (2) $P(D)$ doesn't rely on θ, which is what we really care about and (3) its usability as a normalizing factor can be substituted for the integral value, which ensures that the integral of the posterior distribution is 1.

Recall that to solve for parameters in MLE, we took the argmax of the log likelihood function to get numerical solutions for (μ, σ^2). In Bayesian estimation, we instead compute a distribution over the parameter space, called the **posterior pdf**, denoted as $p(\theta|D)$. This distribution represents how strongly we believe each parameter value is the one that generated our data, after taking into account both the observed data and prior knowledge.

The prior, $p(\theta)$, is also a distribution, usually of the same type as the posterior distribution. I won't get into the details of this, but when the distribution of the prior matches that of the posterior, it is known as a **conjugate prior** and comes with many computational benefits. Our example will use conjugate priors.

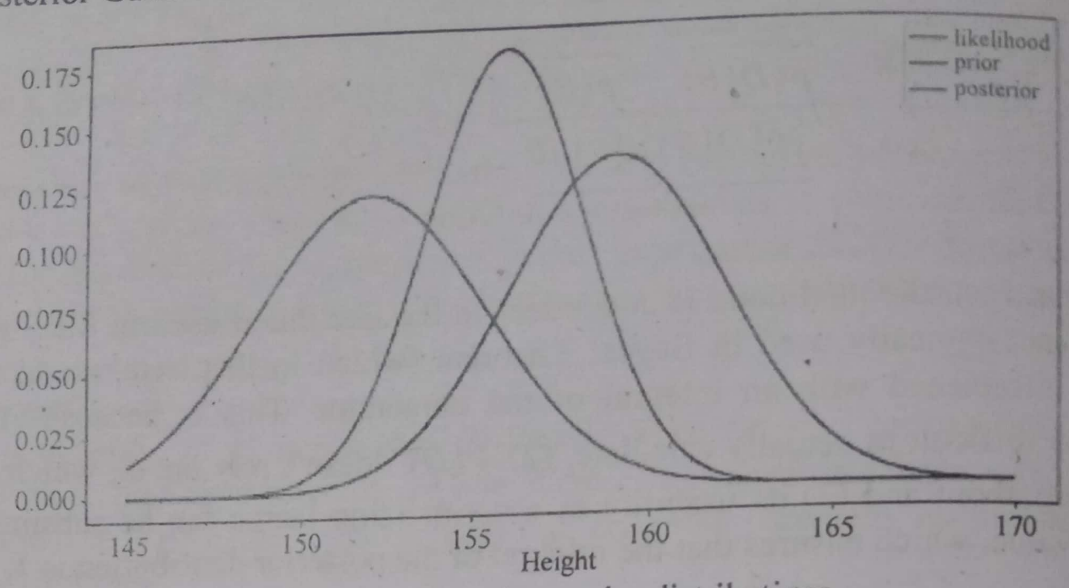
Let's return to our problem concerning tree heights one more time. in addition to the 15 trees recorded by the hiker, we now have means for tree heights over the past 10 years.

Prev. year	1	2	3	4	5	6	7	8	9	10
	160.5	153.8	157.3	160.5	164.0	156.4	157.1	158.1	163.5	160.6

Tree heights for the previous 10 years

Following the assumption that this year, tree heights should fall into the distribution of all the previous year's, our prior is the Gaussian distribution with $\mu = 159.2$ and $\sigma^2 = 9.3$. All that's left is to calculate our posterior pdf. For this calculation, I assume a fixed $\sigma^2 = \sigma^2_MLE = 11.27$. In reality we would not solve Bayesian estimation this way, but multiplying Gaussians of different dimensions, like our likelihood and prior, is extremely complex and I believe simplifying calculations in this case is sufficient for understanding the process and is easier to vicualize. If you'd like more resources on how to execute the full calculation, check out these two links.

Multiplying the univariate likelihood and prior and then normalizing the result, we end up with a posterior Gaussian distribution with $\mu = 155.85$ and $\sigma^2 = 7.05$.



Likelihood, prior and posterior distributions

And there you have the result of Bayesian estimation. As you can see, the posterior distribution takes into account both the prior and likelihood to find a middle ground between them. In light of new observed data, the current posterior becomes the new prior and a new posterior is calculated with the likelihood given by the novel data.

11. When we use MLE or Bayesian estimation technique? [MODEL QUESTION]

Answer:

We've seen the computational differences between the two parameter estimation methods and a natural question now is: When should I use one over the other? While there's no hard and fast fule when selecting a method, I hope you can use the following questions as rough guidelines to steer you in the right direction:

- **How much data are you working with?**

MLE, which depends solely on the outcomes of observed data, is notorious for becoming easily biased when the data is minimal. Consider an experiment where you flip a fair coin 3 times and each flip comes up heads. While you know a fair coin will come up heads 50% of the time, the maximum likelihood estimate tells you that $P(\text{heads}) = 1$ and $P(\text{tails}) = 0$. In situations where observed data is sparse, Bayesian estimation's incorporation of prior knowledge, for instance knowing a fair coin is 50/50, can help in attaining a more accurate model.

- **Do you have reliable prior knowledge about your problem?**

As I just mentioned, prior beliefs can benefit your model in certain situations. However, unreliable priors can lead to a slippery slope of highly **biased models** that require large amounts of seen data to remedy. Make sure that if you are using priors, they are well defined and **contain relevant insight to the problem you're trying to solve.** If you are unsure about the reliability of your priors, MLE may be a better option, especially if you have a sufficient amout of data.

- **Are you limited in computational resources?**

A recurring theme in this article is that Bayesian computations are more complex than those for MLE. With modern computational power, this difference may be inconsequential, however if you do find yourself constrained by resources, MLE may be your best bet.

HIDDEN MARKOV MODELS FOR SEQUENTIAL PATTERN CLASSIFICATION

Multiple Choice Type Questions

1. What is the use of the Hidden Markov Model? **[MODEL QUESTION]**

a) Speech recognition
b) Understanding of the real world
c) Both A and B
d) None of these

Answer: (a)

2. How we can describe the state of the process in HMM? **[MODEL QUESTION]**

a) Literal
b) Single random variable
c) Single discrete random variable
d) None of these

Answer: (c)

Short Answer Type Questions

1. Why it is called hidden Markov model? **[MODEL QUESTION]**

Answer:

In a probabilistic graphical model, the Markov assumption states that the conditional distribution of a variable is independent of all other variables in the graph if the parent nodes of that variable are known.

In the case of HMMs, we are making the Markov assumption in two cases.

1. The observation x in the current state z is only dependent on that state. It is independent of all other states and observations. In other words, the information to generate the current observation is all encoded in the current state. This means,

$$p(x_l \mid x_1, \ldots, x_{l-1}, x_{l+1}, \ldots, x_L, z_1, \ldots, z_{l-1}, z_{l+1}, \ldots, z_L) = p(x_l \mid z_l)$$

2. The current state also encodes the complete information necessary to generate the state in the next step. The next step is independent of all of the previous steps or the upcoming states or observations. This means,

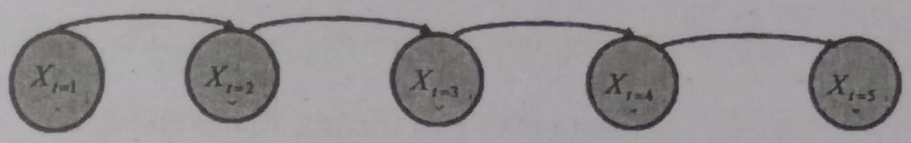
$$p(z_l \mid x_1, \ldots, x_{l-1}, x_{l+1}, \ldots, x_L, z_1, \ldots, z_{l-1}, z_{l+1}, \ldots, z_L) = p(z_l \mid z_{l-1})$$

2. What is a discrete Markov model? **[MODEL QUESTION]**

Answer:

A Markov model is a finite state machine with N distinct states begins at (Time $t=1$) in initial state. It moves from current state to Next state according to the transition probabilities associated with the Current state. This kind of system is called Finite or Discrete Markov model.

The Current state of the system depends only on the previous state of the system. The State of the system at Time $[T+1]$ depends on the state of the system at time T.



Discrete Markov Model example

- A Discrete Markov Model with 5 states.
- Each a_{ij} represents the probability of moving from state 'i' to state 'j'.

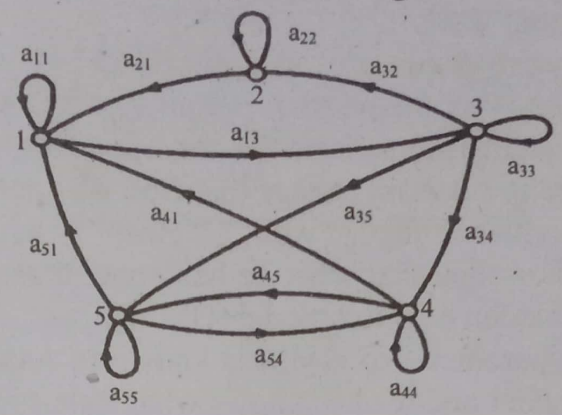


Fig: 1 A Markov chain with five states
(labeled 1 to 5) with selected state transitions

- The probability to start in a given state I is π_i.
- The Vector π represents the start probabilities.
- To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(S_i \mid S_j)$ and initial probabilities $\pi_i = P(S_i)$.

Long Answer Type Questions

1. Describe Hidden Markov model. **[MODEL QUESTION]**

Answer:

A Hidden Markov Model (HMM) is a statistical model which is also used in machine learning. It can be used to describe the evolution of observable events that depend on internal factors, which are not directly observable. These are a class of probabilistic graphical models that allow us to predict a sequence of unknown variables from a set of observed variables.

The Hidden Markov model is a probabilistic model which is used to explain or derive the probabilistic characteristic of any random process. It basically says that an observed event will not be corresponding to its step-by-step status but related to a set of probability distributions. Let's assume a system that is being modelled is assumed to be a Markov chain and in the process, there are some hidden states. In that case, we can say that hidden states are a process that depends on the main Markov process/chain.

The main goal of HMM is to learn about a Markov chain by observing its hidden states. Considering a Markov process X with hidden states Y here the HMM solidifies that for

each time stamp the probability distribution of Y must not depend on the history of X according to that time.

To explain it more we can take the example of two friends, Rahul and Ashok. Now Rahul completes his daily life works according to the weather conditions. Major three activities completed by Rahul are- go jogging, go to the office, and cleaning his residence. What Rahul is doing today depends on whether and whatever Rahul does he tells Ashok and Ashok has no proper information about the weather But Ashok can assume the weather condition according to Rahul work.

Ashok believes that the weather operates as a discrete Markov chain, wherein the chain there are only two states whether the weather is Rainy or it is sunny. The condition of the weather cannot be observed by Ashok, here the conditions of the weather are hidden from Ashok. On each day, there is a certain chance that Bob will perform one activity from the set of the following activities {"jog", "work", "clean"}, which are depending on the weather. Since Rahul tells Ashok that what he has done, those are the observations. The entire system is that of a hidden Markov model (HMM).

Here we can say that the parameter of HMM is known to Ashok because he has general information about the weather and he also knows what Rahul likes to do on average.

So let's consider a day where Rahul called Ashok and told him that he has cleaned his residence. In that scenario, Ashok will have a belief that there are more chances of a rainy day and we can say that belief Ashok has is the start probability of HMM let's say which is like the following:

The states and observation are:

states = ('Rainy', 'Sunny')

observations = ('walk', 'shop', 'clean')

And the start probability is:

start_probability = {'Rainy': 0.6, 'Sunny': 0.4}

Now the distribution of the probability has the weightage more on the rainy day stateside so we can say there will be more chances for a day to being rainy again and the probabilities for next day weather states are as following

```
transition_probability = {

    'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
    'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6},

    }
```

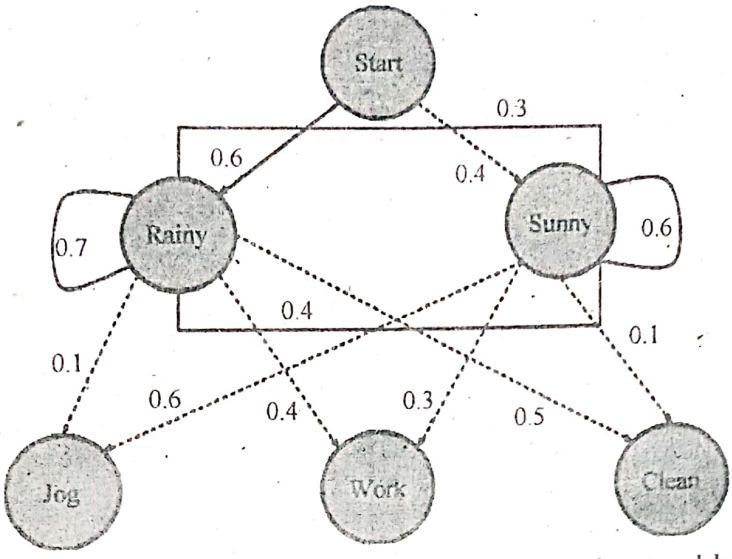
From the above we can say the changes in the probability for a day is transition probabilities and according to the transition probability the emitted results for the probability of work that Rahul will perform is

```
emission_probability = {

    'Rainy' : {'jog': 0.1, 'work': 0.4, 'clean': 0.5},
    'Sunny' : {'jog': 0.6, 'work: 0.3, 'clean': 0.1},

    }
```

This probability can be considered as the emission probability. Using the emission probability Ashok can predict the states of the weather or using the transition probabilities Ashok can predict the work which Rahul is going to perform the next day. Below image shown the HMM process for making probabilities-



So here from the above intuition and the example we can understand how we can use this probabilistic model to make a prediction. Now let's just discuss the applications where it can be used.

An application, where HMM is used, aims to recover the data sequence where the next sequence of the data can not be observed immediately but the next data depends on the old sequences. Taking the above intuition into account the HMM can be used in the following applications:

- Computational finance
- Speed analysis
- Speech recognition
- Speech synthesis
- Part-of-speech tagging
- Document separation in scanning solutions
- Machine translation
- Handwriting recognition
- Time series analysis
- Activity recognition
- Sequence classification
- Transportation forecasting

[MODEL QUESTION]

2. Explain discrete hidden Markov model.

Answer:

The discrete-time Markov model is to some extent restrictive and cannot be applicable to all problems of interest. The extension implies that every state will be probabilistic and not deterministic (e.g., sunny). This means that every state generates an observation at

time t, o_t, according to a probabilistic function $b_j(o_t)$ for each state j. The production of observations in the stochastic approach is characterized by a set of observation probability measures, $B = \{b_j(o_t)\}_{j=1}^{N}$, where, N is the total number of states and the probabilistic function for each state j is

$$b_j(o_t) = P(o_t \mid q_t = j) \qquad \text{.... (1)}$$

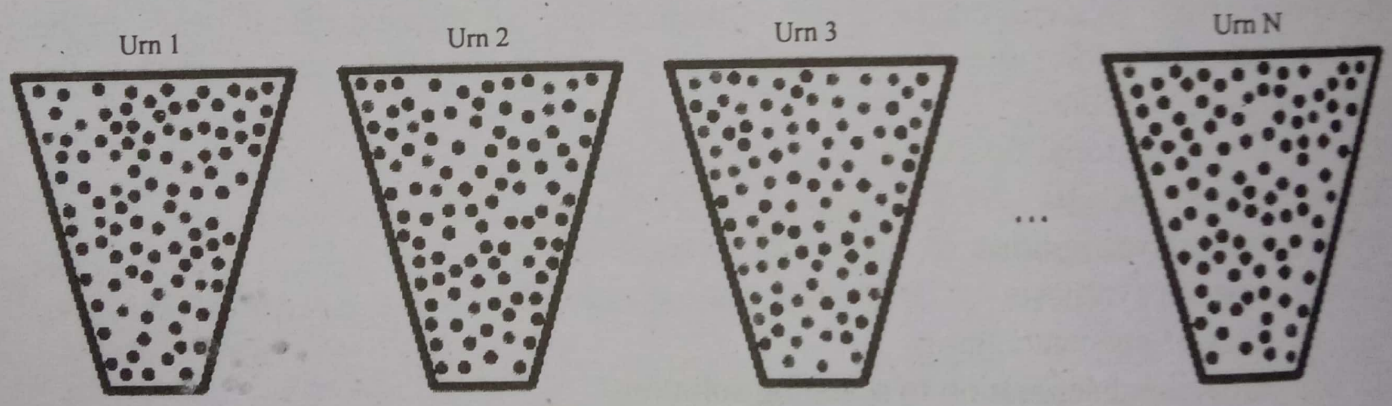
The concept of how the HMM works is introduced by using an example called **the urn and ball model.**

The Urn and Ball Model

Assume that there are N large galss urns with coloured balls. There are M distinct colours of the balls, (Fig. 1).

The steps needed for generating an observation sequence are:

1. Choose an initial state distribution π. In the urn and ball model the urn corresponds to a state.
2. Set $t = 1$ (i.e., clock, $t = 1, 2, \ldots, T$).
3. Choose a ball from the selected urn (state) according to the symbol probability distribution in state $i, b_i(o_t)$. This coloured ball represents the observation o_t. Put the ball back to the urn. For example, the probability for a purple ball is in the first urn 0.61, (Fig. 1).



Urn 1	Urn 2	Urn 3	Urn N
$P(R) = 12\%$	$P(R) = 30\%$	$P(R) = 8\%$	$P(R) = 2\%$
$P(O) = 7\%$	$P(O) = 1\%$	$P(O) = 9\%$	$P(O) = 4\%$
$P(K) = 6\%$	$P(K) = 37\%$	$P(K) = 10\%$	$P(K) = 3\%$
$P(G) = 6\%$	$P(G) = 10\%$	$P(G) = 54\%$	$P(G) = 1\%$
$P(B) = 10\%$	$P(B) = 17\%$	$P(B) = 10\%$	$P(B) = 88\%$
$P(P) = 61\%$	$P(P) = 5\%$	$P(P) = 9\%$	$P(P) = 2\%$

Fig: 1 Urn and ball example with N urns and $M = 6$ different colours
(R = Red, O = Orange, K = Black, G = Green, B = Blue, P = Purple)

4. Transit to a new state (urn) $q_{t+1} = j$ according to the state-transition probability distribution for state i, i.e., a_{ij}.

5. Set $t = t+1$; return to step 3 if $t < T$; otherwise, terminate the procedure.

These steps describe how the HMM works when generating an observation sequence. It should be noted that the urns can contain balls of the same colour and the distinction among various urns is the way the collection of the coloured balls is composed. Therefore, an isolated observation of a ball of a particular colour does not immediately indicate which urn it is drawn from. Furthermre, the link between the urn and ball example and a HMM, is that the urn corresponds to a state and the colour corresponds to a featuere vector (the observation).

3. Explain continuous and discrete density hidden Markov model.

[MODEL QUESTION]

Answer:

Discrete Observation Densities:

The urn and ball example described in the previous section is an example of a discrete observation density HMM with M distinct codewards (colours). In general, the discrete observation density partitions the probability density function (pdf) into M codewords or cells with assigned probabilities. these codewords will be denoted $v_1, v_2, \ldots, v_M$ where one symbol corresponds to one cell. The partitioning is usually called **vector quantization** and for this purpose several analysis methods exist. Vector quantization is performed by creating a codebook based on the mean vectors for every cluster.

The symbol for the observation is determined by the nearest neighbour rule, i.e., the symbol in the cell with the nearest codebook vector is selected. In the urn and ball example this corresponds to the dark grey ball being classified as a black, due to that this is the most probable choice. In this case the symbols $v_1, v_2, \ldots, v_M$ are represented by different colours (e.g., $v_1 =$ red). The probability distributions for the observable symbols are denoted $B = \{b_j(o_t)\}_{j=1}^{N}$, where the symbol distribution in state j is defined as

$$b_j(o_t) = b_j(k) = P(o_t = v_k \mid q_t = j),\ 1 \le k \le M \qquad \ldots (1)$$

This is the first step in the deteermination of the codebook. The second step is to estimate the sets of observation probabilities for each codebook vector in every state respectively. the major problem with discrete output probability is the vector quantization operation partitioning which destroys the original signal structure, when the continuous space is separated into regions. If the output distributions are overlapping the partitioning introduces errors. This is due to a finite training data set being used implying unreliable parameter estimates. The vector quantization errors introduced may cause performance degradation of the discrete density HMM [2]. This will occur when the observed feature vector is intermediate between two codebook symbols. This is why **continuous observation densities** can be used instead of discrete observation densities. The continuous approach avods quntization errors. However, this approach typially require more calculations.

Continuous Observation Densities:

The continuous observation densities $b_j(o_t)$ in the HMM are created by using a parametric probability density function or a mixture of several functions. To be able to reestimate the parameters of the probability density function (pdf) some restrictions for the pdf are needed. A common restriction is that the pdf should belong to the log-concave or elliptically symmetrical density family. The most ganeral representation of the pdf, for which a reestimation procedure has been formulated, is the finite mixture of the form

$$b_j(o_t) = \sum_{k=1}^{M} c_{jk} b_{jk}(o_t), \quad j = 1, 2, \ldots, N \qquad \ldots (2)$$

where, M is the number of mixtures. The stochastic constraints for the mixture weights, c_{jk}, must fulfil

$$\sum_{k=1}^{M} c_{jk} = 1, \quad j = 1, 2, \ldots, N \qquad \ldots (3)$$

$$c_{jk} \geq 0 \qquad j = 1, 2, \ldots, N, \; k = 1, 2, \ldots, M$$

The $b_{jk}(o_t)$ can be either D-dimensional log-concave or of elliptically symmetric density with mean vector μ_{jk} and covariance matrix Σ_{jk}

$$b_{jk}(o_t) = N(o_t, \mu_{jk}, \Sigma_{jk}) \qquad \ldots (4)$$

The most utilized D-dimensional log-concave or elliptically symmetric density function, is the Gaussian density function

$$b_{jk}(o_t) = N(o_t, \mu_{jk}, \Sigma_{jk}) = \frac{1}{(2\pi)^{D/2} |\Sigma_{jk}|^{1/2}} e^{-\frac{1}{2}(o_t - \mu_{jk})^T \Sigma_{jk}^{-1} (o_t - \mu_{jk})} \qquad \ldots (5)$$

To approximate a basic observation source, the Gaussian mixture provides a straighforward solution to gaining a considerable accuracy. This is due to the flexibility and convenient estimation of the pdfs. If the observation souce generates a complicated high-dimensional pdf, the Gaussian mixture becomes computationally difficult to treat, due to the considerable number of parameters to estimate and large covariance matrices to estimate these parameters there is a need of a huge amount of well representative training data, this is, however, hard to achieve in a practical situation. With insufficient training data sets the parameter estimation will be unreliable. Especially the covariance matrix estimation is highly sensitive for the lack of representative training data.

The size of the covariance matrices increases with the square proportional to the vector dimension D. If the feature vectors are designed to avoid redundant components, the elements outside the diagonal of the covariance matrices are usually small. One solution is to use a diagonal covariance matrix approximation. The diagonality provides a simpler and a faster implementation for the probability computation. This is performed by reducing Eqn. (5) to

$$b_{jk}(o_t) = N(o_t, \mu_{jk}, \Sigma_{jk}) = \frac{1}{(2\pi)^{D/2}\left|\Sigma_{jk}\right|^{1/2}} e^{-\frac{1}{2}(o_t-\mu_{jk})^T \Sigma_{jk}^{-1}(o_t-\mu_{jk})}$$

$$= \frac{1}{(2\pi)^{D/2}\left(\Pi_{d=1}^{D}\sigma_{jkd}\right)^{1/2}} e^{-\sum_{d=1}^{D}\frac{(o_{td}-\mu_{jkd})^2}{2\sigma_{jkd}^2}} \quad \text{.... (6)}$$

where, the variance terms $\sigma_{jk1}, \sigma_{jk2}, \ldots, \sigma_{jkD}$, are the diagonal elements of the covariance matrix Σ_{jk}.

4. What are types of hidden Markov models? **[MODEL QUESTION]**

Answer:

Different kinds of structures of HMMs can be used. The structure is defined by the transition matrix, A. The most general structure is the **ergodic** or fully connected HMM. In this model every state can be reached from every other state of the model.

As an example in this section a $N = 4$ state model is introduced, see Fig. 1(a). The ergodic model has the property $0 < a_{ij} < 1$ where the "zero" and the "one" have been excluded in order to fulfill the ergodic property. The state transition matrix, A, for an ergodic model, can be described by

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}_{4\times 4} \quad \text{.... (1)}$$

In some pattern recognition tasks, it is desirable to use a model which models the observations in a successive manner. This can be done by employing the left-right model or Bakis model, see Figs. 1(b), 1(c). The property for a left-right model is

$$a_{ij} = 0, \; j < i \quad \text{.... (2)}$$

This implies that no transitions can be made to previous states. The lengths of the transitions are usually restricted to some maximum length Δ, typically two or three

$$a_{ij} = 0, \; j > i + \Delta \quad \text{.... (3)}$$

Note that for a left-right model, the state transitions coefficients for the last state have the followig property

$$a_{NN} = 1$$
$$a_{Nj} = 0, \; j < N \quad \text{.... (4)}$$

If $\Delta = 1$ yields that

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}_{4\times4} \quad \text{.... (5)}$$

which corresponds to Fig. 1(b).

If $\Delta = 2$ yields that

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}_{4\times4} \quad \text{.... (6)}$$

which corresponds to Fig. 1(c).

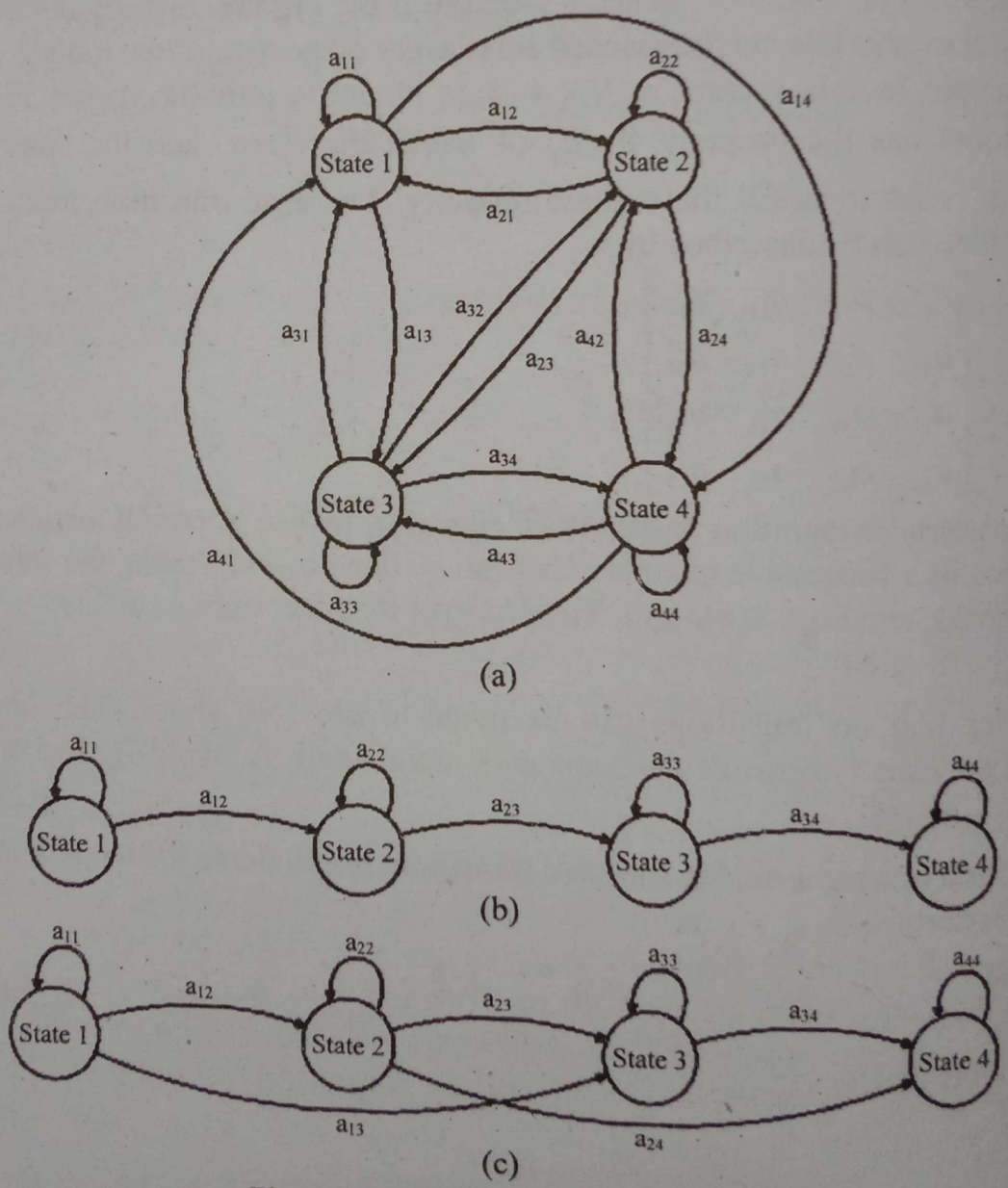


Fig: 1 Different structures for HMMs

The choice of a constrained model structure (for example the left-right model) requires no modification in the training algorithms, due to the fact that an initial state transition probability set to zero will remain zero.

5. What are problems of Hidden Markov model? **[MODEL QUESTION]**

Answer:

Given the basics of a HMM, three central problems arise for applying the model to real-world applications.

Problem 1:

The observation sequence is denoted $O=(o_1, o_2, \ldots, o_T)$ and the HMM $\lambda=(A, B, \pi)$. The question is how to compute the probability of the observation sequence for a given HMM, $P(O|\lambda)$.

Problem 2:

The observation sequence is denoted $O=(o_1, o_2, \ldots, o_T)$ and HMM $\lambda=(A, B, \pi)$. The question is how to choose the corresponding state sequence $q=(q_1, q_2, \ldots, q_T)$, in an optimal sense (i.e., best "explains" the observations).

Problem 3

How can the probability measures, $\lambda=(A, B, \pi)$, be adjusted to maximize $P(O|\lambda)$?

The first problem can be viewed as a recognition problem. With a number of trained models the task is to find a model which best describes the observation sequence given.

In the second problem the task is to uncover the hidden part of the model. In HMMs it is more or less impossible to find the "correct" state sequence, except in the case of degenerated models. The problem should be solved in some optimal sense, which will be explained later.

The third problem can be viewed as the training problem, i.e., models are created for each specific application based on the corresponding training sequences. The training problem is the hardest and most crucial one in most applications.

The HMM should be optimally adapted based on the training data to fit and describe real world phenomena as well as possible.

DIMENSION REDUCTION METHODS

Multiple Choice Type Questions

1. What does dimensionality reduction reduce? **[MODEL QUESTION]**

a) Stochastics b) Collinerity c) Performance d) entropy

Answer: (b)

Short Answer Type Questions

1. Why we need to reduce dimension? **[MODEL QUESTION]**

Answer:

Big Data Analytics is a buzzword nowadays. Everyone is talking about it. Big data Analytics has found application in many sectors like medicine, politics, dating. Though big data analytics is used in bettering many aspects of human life, it comes with its own problems. One of them is '**Curse of dimensionality**'. Curse of dimensionality refers to an exponential increase in the size of data caused by a large number of dimensions. As the number of dimensions of a data increases, it becomes more and more difficult to process it.

Dimension Reduction is a solution to the curse of dimensionality. In layman's terms, dimension reduction methods reduce the size of data by extracting relevant information and disposing rest of data as noise.

Dimensionality Reduction is simply reducing the number of features (columns) while retaining maximum information. Following are reasons for Dimensionality Reduction:

- Dimensionality Reduction helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.
- Removes Correlated Features.
- Reducing the dimensions of data to 2D or 3D may allow us to plot and visualize it precisely. You can then observe patterns more clearly.
- It is helpful in noise removal also and as a result of that, we can improve the performance of models.

2. What are benefits of dimension reduction? **[MODEL QUESTION]**

Answer:

Dimension reduction offers several benefits such as-

- It compresses the data and thus reduces the storage space requirements.
- It reduces the time required for computation since less dimensions require less computation.
- It eliminates the redundant features.
- It improves the model performance.

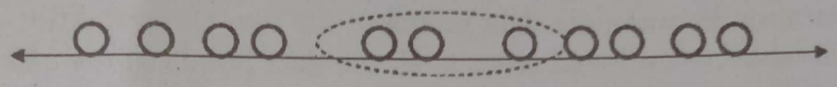
Long Answer Type Questions

1. Describe Fisher linear discriminant analysis. **[MODEL QUESTION]**

Answer:

Linear Discriminant Analysis or **Normal Discriminant Analysis** or **Discriminant Function Analysis** is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space.

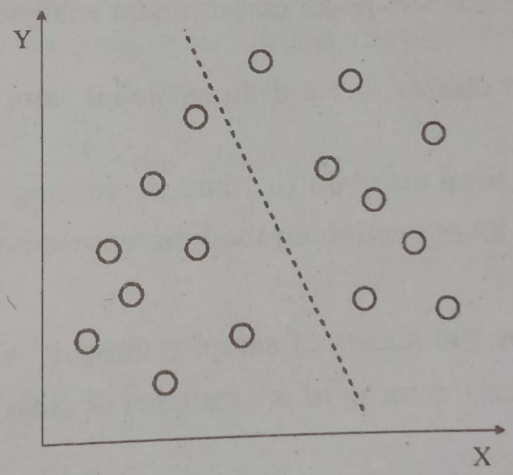
For example, we have two classes and we need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping as shown in the below figure. So, we will keep on increasing the number of features for proper classification.



Overlapping

Example:

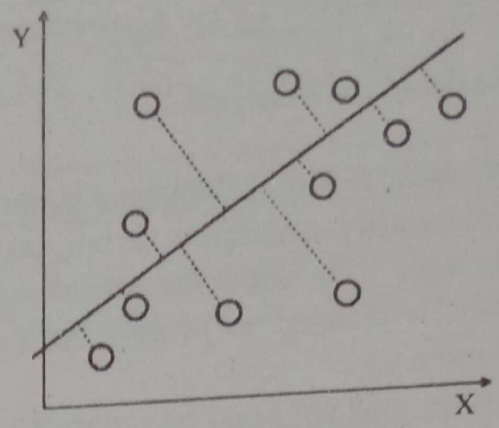
Suppose we have two sets of data points belonging to two different classes that we want to classify. As shown in the given 2D graph, when the data points are plotted on the 2D plane, there's no straight line that can separate the two classes of the data points completely. Hence, in this case, LDA (Linear Discriminant Analysis) is used which reduces the 2D graph into a 1D graph in order to maximize the separability between the two classes.



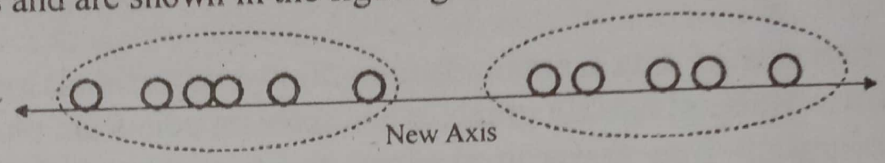
Here, Linear Discriminant Analysis uses both the axes (X and Y) to create a new axis and projects data onto a new axis in a way to maximize the separation of the two categories and hence, reducing the 2D graph into a 1D graph.

Two criteria are used by LDA to create a new axis:

1. Maximize the distance between means of the two classes.
2. Minimize the variation within each class.



In the above graph, it can be seen that a new axis (in red) is generated and plotted in the 2D graph such that it maximizes the distance between the means of the two classes and minimizes the variation within each class. In simple terms, this newly generated axis increases the separation between the data points of the two classes. After generating this new axis using the above-mentioned criteria, all the data points of the classes are plotted on this new axis and are shown in the figure given below.



But Linear Discriminant Analysis fails when the mean of the distributions are shared, as it becomes impossible for LDA to find a new axis that makes both the classes linearly separable. In such cases, we use non-linear discriminant analysis.

Mathematics:

Let's suppose we have two classes and a d-dimensional samples such as $x_1, x_2, \ldots, x_n$, where:

- n_1 samples coming from the class (c_1) and n_2 coming from the class (c_2).

If x_i is the data point, then its projection on the line represented by unit vector v can be written as $v^T x_i$.

Let's consider u_1 and u_2 be the means of samples class c_1 and c_2 respectively before projection and u_1 that denotes the mean of the samples of class after projection and it can be calculated by:

$$\tilde{\mu}_1 = \frac{1}{n_1} \sum_{x_i \in c_1}^{n_1} v^T x_i = v^T \mu_1$$

Similarly,

$$\tilde{\mu}_2 = v^T \mu_2$$

Now, In LDA we need to normalize $\left|\backslash\text{widetilde}\{\backslash\mu_1\}\text{-}\backslash\text{widetilde}\{\backslash\mu_2\}\right|$.

Let $y_i = v^{\{T\}} x_i$ be the projected samples, then scatter for the samples of c_1 is:

$$\tilde{s}_1^2 = \sum_{y_i \in c_1} (y_i - \mu_1)^2$$

Similarly:

$$\tilde{s}_2^2 = \sum_{y_i \in c_1} (y_i - \mu_2)^2$$

Now, we need to project our data on the line having direction v which maximizes

$$J(v) = \frac{\tilde{\mu}_1 - \tilde{\mu}_2}{s_1^2 + s_2^2}$$

For maximizing the above equation we need to find a projection vector that maximizes the difference of means of reduces the scatters of both classes. Now, scatter matrix of s_1 and s_2 of classes c_1 and c_2 are:

$$s_1 = \sum_{x_i \in c_1} (x_i - \mu_1)(x_i - \mu_1)^T$$

and $$s_2 = \sum_{x_i \in c_2} (x_i - \mu_2)(x_i - \mu_2)^T$$

After simplifying the above equation, we get:

Now, we define, scatter withing the classes (s_w) and scatter b/w the classes (s_b):

$$s_w = s_1 + s_2$$

$$s_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

Now, we try to somplify the numerator part of $J(v)$

$$J(v) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|}{s_1^2 + s_2^2} = \frac{v^T s_b v}{v^T s_w v}$$

Now, to maximize the above equation we need to calculate differentiation with respect to v.

$$\frac{dJ(v)}{dv} = s_b v - \frac{v_t s_b v (s_w v)}{v^T s_w v} = s_b v - \lambda s_w v = 0$$

$$s_b v = \lambda s_w v$$

$$s_w^{-1} s_b v - \lambda v$$

$$Mv = \lambda v$$

where, $\lambda = \dfrac{v^T s_b v}{v^T s_w v}$ and $M = s_w^{-1} s_b$

Here, for the maximum value of J(v) we will use the value corresponding to the highest eigenvalue. This will provide us the best solution for LDA.

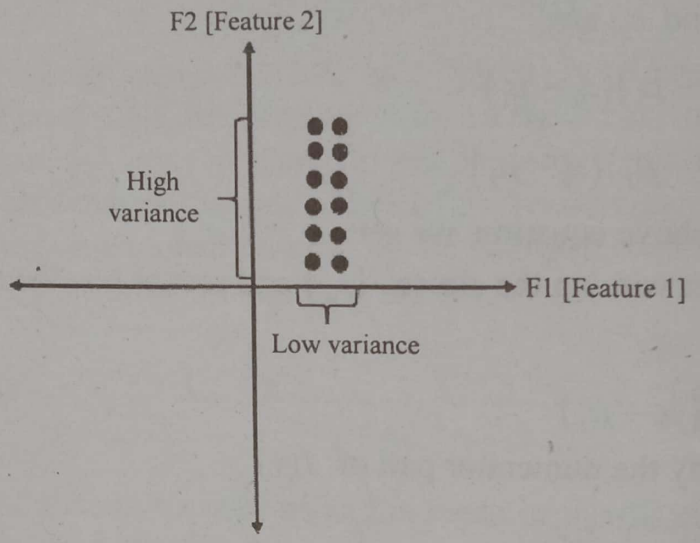
Extensions to LDA:

1. **Quadratic Discriminant Analysis (QDA):** Each class uses its own estimate of variance (or covariance when there are multiple input variables).
2. **Flexible Discriminant Analysis (FDA):** Where non-linear combinations of inputs are used such as splines.
3. **Regularized Discriminant Analysis (RDA):** Introduces regularization into the estimate of the variance (actually covariance), moderating the influence of different variables on LDA.

2. Describe principal component analysis technique. **[MODEL QUESTION]**

Answer:

PCA is a linear dimensionality reduction technique which converts a set of correlated features in the high dimensional space into a series of uncorrelated features in the low dimensional space. These uncorrelated features are also called principal components. PCA is an orthogonal linear transformation which means that all the principal components are perpendicular to each other. It transforms the data in such a way that the first component tries to explain maximum variance from the original data. It is an unsupervised algorithm i.e. it does not take into consideration the class labels.

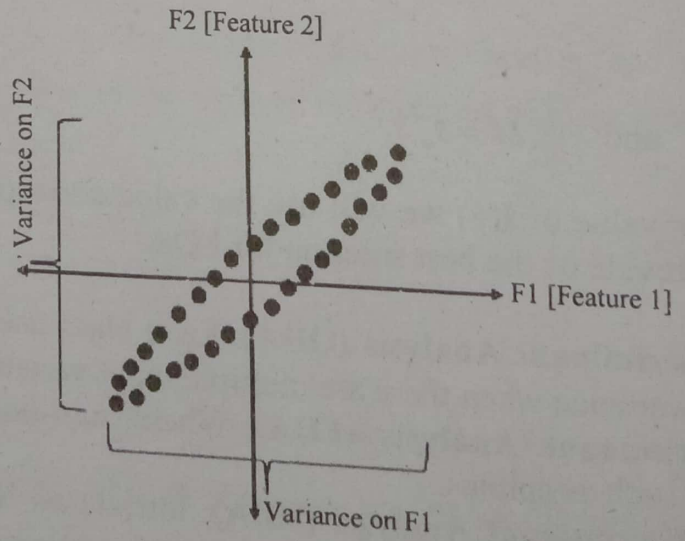


The above figure shows two features F1 and F2 plotted in a two-dimensional space. As we can see from the plot above, the variance on feature F2 is much higher than the variance on feature F1 which means

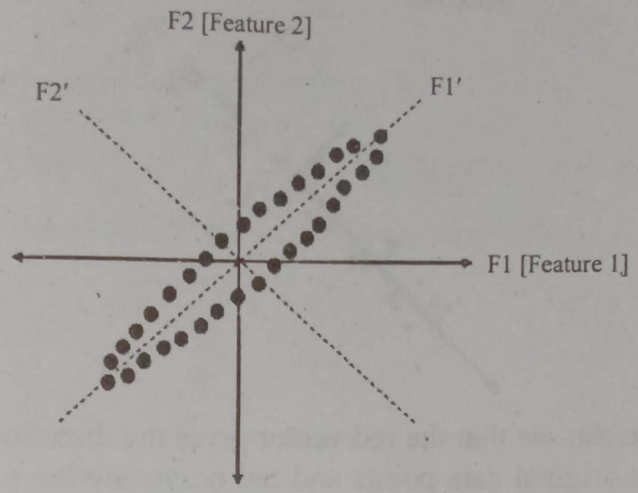
information preserved by F2 >> information preserved by F1

Now suppose we want to convert this 2D data into 1D, we need to drop one feature. We know that feature F1 preserves much less information than F2, so we can safely drop feature F1 and use feature F2 as our final feature.

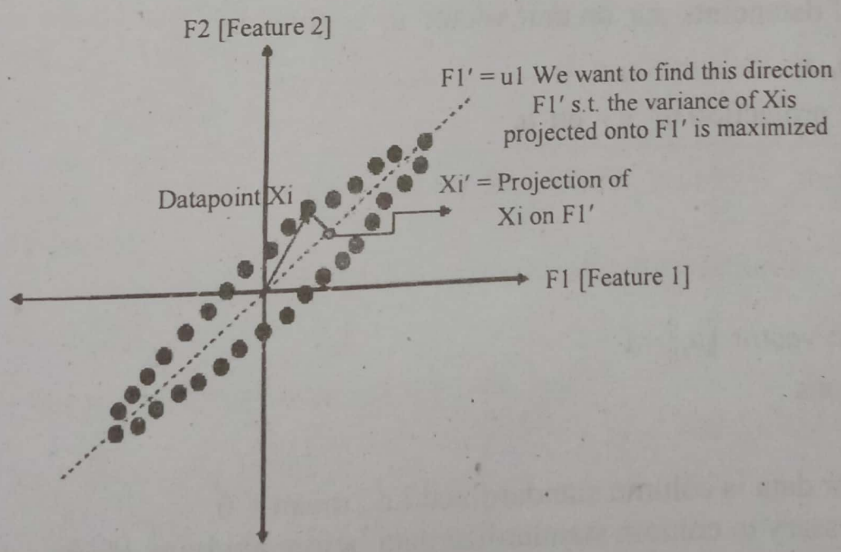
Now, let's look at one more plot



Here, the variance on feature F2 is equal to the variance on feature F1 which means **Information preserved by F2 = information preserved by F1**
now if we want to reduce the dimensions of the data from 2D to 1D we cannot drop one feature as we lose a lot of information.
So, what can we do here?
Here, PCA comes into the picture

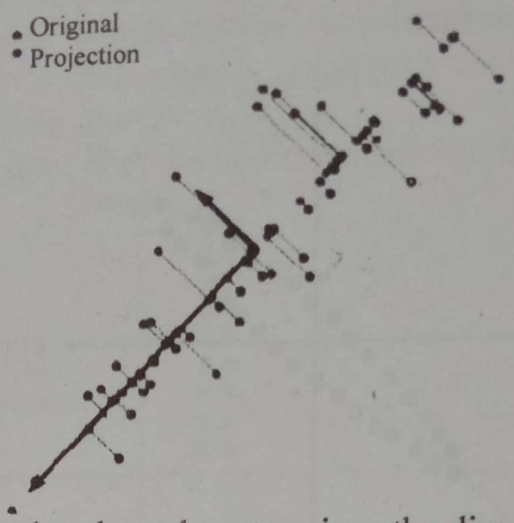


PCA performs linear orthogonal transformation on the data to find features F1′ and F2′ such that the variance on F1′ >> variance on F2′.
These new sets of features are called principal components. Here, F1′ is the first principal component which gives the direction of maximum variance, and F2′ is the second principal component which gives the second direction with most variance. F1′ and F2′ are unit vectors and they are perpendicular to each other. Since F1′ and F2′ are our new set of features we can safely drop F2′ as F1′ preserves maximum information.



PCA tries to find a direction F1′ such that the variance of point Xis projected onto F1′ is maximized.

Here,
Xi = datapoint
u1 = unit vector in the direction of maximum variance
Xi′ = projection of Xi on u1



In the above plot, we can see that the red vector gives the direction of maximum variance. Green points are the original data points and red points are the projection of these points on the axis.

So far, we have looked at the basic intuition of PCA. Now let's have a look at the Maths behind PCA.

PCA Maths.

The goal of PCA is to reduce the dimensionality of data while retaining maximum variance.

"Here, we want to find a unit vactor u_1 which points in the direction of maximum variance, which means if we project our data points on u1 the variance is maximized".

The projection of datapoints $x_i s$ on unit vector u_1 is given by:

$$x_1' = \text{proj}_{u1} x_i$$

where, x_1' is the projection of $x_i s$ on u_1

$$x_i' = \frac{u_1 x_i}{\|u_1\|^2}$$

$$x_i' = u_1^T x_i$$

Since, u_1 is a unit vector $\|u_1\| = 1$ .... (1)

Mean of projections

$$\overline{x_i'} = u_1^T \overline{x_i}$$

Let's consider our data is column standardized i.e., mean = 0

NOTE: It is necessary to column standardize data before applying PCA

The Covariance matrix S for a d-dimensional data with n number of datapoints is given by

$$S_{d\times d} = \frac{1}{n}\left(x_{d\times n}^T x_{n\times d}\right) \qquad \text{.... (2)}$$

where, x is a $n\times d$ data matrix and S is $d\times d$ square symmetric matrix
Now, we need to find a unit vector u_1 such that variance of points $x_i s$ projected on u_1 is maximized i.e., we need to find and maximize –

$$\text{var}\left\{\text{proj}_{u_1} x_i\right\}_{i=1}^n$$

The variance of points $x_i s$ projected on u_1 is given by

$$\text{var}\left\{u_1^T x_i\right\}_{i=1}^n = \frac{1}{n}\sum_{i=1}^n \left(u_1^T x_i - u_1^T \overline{x}_i\right)^2 \qquad \text{since, } \text{proj}_{u_1} x_i = u_1^T x_i$$

Since our data is column standardized the second term in the above equation becomes 0

$$\text{var}\left\{u_1^T x_i\right\}_{i=1}^n = \frac{1}{n}\sum_{i=1}^n \left(u_1^T x_i\right)^2$$

$$\text{var}\left\{u_1^T x_i\right\}_{i=1}^n = \frac{1}{n}\sum_{i=1}^n u_1^T (x_i)(x_i)^T u_1$$

$$\text{var}\left\{u_1^T x_i\right\}_{i=1}^n = u_1^T \left[\frac{1}{n}\sum_{i=1}^n (x_i)(x_i)^T\right] u_1$$

Here, $\left[\frac{1}{n}\sum_{i=1}^n (x_i)(x_i)^T\right]$ this term is the equation for covariance matrix, from Eqn. (2)

So, our final equation becomes:

$$\text{var}\left\{u_1^T x_i\right\}_{i=1}^n = u_1^T S u_1$$

In order to maximize the variance we need to maximize this equation
Here, S is a $d\times d$ Covariance matrix, from Eqn. (2)
Our final objective function for PCA is

$$\boxed{\max_{u_1} u_1^T S u_1 \quad \text{s.t. } u_1^T u_1 = 1 \qquad \text{.... (3)}}$$

Here, S is a known quantity, so we need to find a unit vector u_1 that maximizes the above objective function
Using power of lagrange multipliers we can write the above equation as

$$L(u_1, \lambda) = u_1^T S u_1 + \lambda\left(1 - u_1^T u_1\right)$$

Taking derative of the above equation w.r.t. u_1 we get

$$\frac{d}{du_1} u_1^T S u_1 + \lambda\left(1 - u_1^T u_1\right) = 0$$

$$2Su_1 - 2\lambda u_1 = 0$$

$$Su_1 = \lambda u_1 \qquad \text{.... (4)}$$

- u_1 is the eigen vector of Covariance matrix S and λ is the corresponding eigen value
- u_1 is the unit vector which gives the direction of maximum variance
- For a $d \times d$ covariance matrix, we get d eigen vector and eigen value pairs
- For a given covariance matrix every pair of eigen vectors u_1, u_j is perpendicular to each other. $\therefore u_1^T u_j = 0$

Now let's see, how to calculate eigen vectors and eigen values for the covariance matrix.
To get the eigen values and eigen vectors of a covariance matrix S we use,

$$Su_i = \lambda_i u_i \qquad \text{from Eqn. (4)}$$

where, λ_i = eigen values of S

u_i = eigen vectors of S

We can write the above equation as,

$$Su_i - \lambda_i u_i = 0$$

For each eigenvalue λ_i, we have a specific eigenvalue equation

$$(S - \lambda_i I)u_i = 0, \text{ where, } I = \text{identity matrix} \qquad \text{.... (5)}$$

Now, we need to find the determinant of the matrix $(S - \lambda_i I)$

$$\det|S - \lambda_i I| = 0$$

- After solving for the determinent of the above matrix, we get an equation whose roots are the eigen values λ.
- Once we get the eigen values, we can substitute these eigen values in Eqn. (6) one by one and solve the equation to get the eigen vector for each eigen value

How do we decide which eigen values to choose?

$$u_1^T S u_1 = \lambda u_1 u_1^T \quad \text{multiplying both sides by } u_1^T\text{, from Eqn. (4)}$$

$$u_1^T S u_1 = \lambda \qquad \text{since, } u_1 u_1^T = 1 \qquad \text{.... (6)}$$

- Here, the LHS is the same term from Eqn. (3) we want to maximize
- From the above equation we can see that in order to maximize this term we need to maximize the value of lamda i.e., we need to find the largest eigen value λ,
- Here, the largest eigen value λ corresponds to the eigen vector u_1 which gives the direction of maximum variance.

How do we convert D-dimensions to D'-dimensions such that $D' \ll D$, preserving maximum information (variance).

Here, we are decomposing the original covariance matrix S into its eigen values and eigen vectors, "we know that, for a $d \times d$ covariance matrix we get d eigen vector and eigen value pairs".

$$Su_1 = \lambda_1 u_1 \qquad \text{from Eqn. (4)}$$

$$Su_2 = \lambda_2 u_2$$

$$Su_3 = \lambda_3 u_3$$

$$\vdots$$

$$Su_d = \lambda_d u_d$$

$$U^{-1}SU = U\Lambda U^{-1} \qquad \text{multiplying both sides by } U^{-1}\text{, from Eqn. (4)}$$

$$S = U\Lambda U^{-1}$$

This is the eqution for eigen decomposition of a matrix where,

$$U = [u_1, u_2, u_3, \ldots, u_d] \text{ where, } u_i \in R^d$$

$$\Lambda = \begin{bmatrix} \lambda_1 & . & . & 0 \\ . & . & & . \\ . & & . & . \\ 0 & . & . & \lambda_d \end{bmatrix} \text{ where, } \lambda_i \in R^d$$

- Here, U is a $d \times d$ square matrix containing all eigen vectors of S
- Λ is a diagonal matrix where all the non-diagonal elements are 0
- The diagonals of matrix Λ consists of all eigen values of S
- The number of eigen values and eigen vectors are equal to the dimensions of the data
- Eigen vectors u_i are unit vectors, which means that their length or magnitude is equal to 1.0
- Eigen vectors point in the direction of maximum variance
- Eigenvalues are coefficients applied to eigenvectors that give the vectors their length or magnitude.
- Eigen values stretch / scale the eigen vectors in the direction the vector is pointing or in the opposite direction without rotating it for example: a negative eigen value may reverse the direction of the eigen vector as part of scaling it.
- Eigen values are proportional to the variance.

The total variance is explained by:

$$\text{Total variance} = \sum_{i=1}^{d} \lambda_i \text{, where, } d \text{ is equal to the dimensions of data} \qquad \ldots (7)$$

If we want to convert d-dimensions of d'-dinensions such that $d' \ll d$, we need to select the top d' eigen values which retain maximum information,

$$\text{Retained variance} = \sum_{i=1}^{d'} \lambda_i \text{, where, } d' \text{ is equal to the reduced dimensions} \qquad \ldots (8)$$

Hence, the amount of information retained can be given by

$$\%\text{ info} = \frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \qquad \text{from Eqns. (7) \& (8)}$$

By taking a subset of the eigen vector and eigen value pairs we are able to retain most of the information while using only a fraction of the original dimension.

Which pair of eigen values and eigen vectors to choose?

From Eqn. (6), we know that in order to maximize the variance we need to maximize the value of λ, i.e., we need to select the largest eigen values and the corresponding eigen vectors.

Now, looking back at the equation we got during maximizing the retained variance,

$$Su_i = \lambda_i u_i \qquad \text{from Eqn. (4)}$$

For a $d \times d$ covariance matrix S, we first determine the d eigen vector and eigen value pairs using eigen decomposition.

Now, we sort these pairs based on eigen values in descending order,

$$(\lambda_1, u_1)(\lambda_2, u_2)(\lambda_3, u_3).....(\lambda_d, u_d)$$

where, $\lambda_1 > \lambda_2 > \lambda_3 > ..... > \lambda_d$

Since, the eigen values are proportional to the variance retained we select only the top d' pairs with the largest eigen values.

$$(\lambda_1, u_1)(\lambda_2, u_2)(\lambda_3, u_3).....(\lambda_{d'}, u_{d'})$$

where, d' is the smallest set of eigen vector and eigen value pairs that can retain maximum variance.

How do we perform the transformation?

From Eqn. (1) we know that the projection of $x_i s$ on a unit vector u is given by

$$x_i' = U'^T x_i, \text{ where, } x_i \in R^d \quad x_i' \in R^{d'} \quad d \gg d'$$

Here, matrix U' consists of the top d' eigen vectors of covariance matrix S.

$$U' = [u_1, u_2, u_3, ..., u_{d'}], \text{ where, } u_i \in R^{d'}$$

U' is our final transformation matrix.

- From the above equation, we can transform a d-dimensional vector x into a d'-dimensional vector x' while still retaining maximum information.
- Here, we are using a linear transformation U on input x_i to transform it into x_i' in a reduced feature space.

3. What are steps of PCA algorithm? **[MODEL QUESTION]**

Answer:

The steps involved in PCA Algorithm are as follows-

Step-01: Get data.

Step-02: Compute the mean vector (μ).

Step-03: Subtract mean from the given data.

Step-04: Calculate the covariance matrix.

Step-05: Calculate the eigen vectors and eigen values of the covariance matrix.

Step-06: Choosing components and forming a feature vector.

Step-07: Deriving the new data set.

Example

Compute the principal component of following data-

CLASS 1

X = 2, 3, 4

Y = 1, 5, 3

CLASS 2

X = 5, 6, 7

Y = 6, 7, 8

Answer: We use the above discussed PCA Algorithm –

Step-01:

Get data: The given feature vectors are –

- $x_1 = (2, 1)$
- $x_2 = (3, 5)$
- $x_3 = (4, 3)$
- $x_4 = (5, 6)$
- $x_5 = (6, 7)$
- $x_6 = (7, 8)$

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Step-02:

Calculate the mean vector (μ)

Mean vector $(\mu) = ((2+3+4+5+6+7)/6, (1+5+3+6+7+8)/6) = (4.5, 5)$

Thus, Mean vector $(\mu) = \begin{bmatrix} 4.5 \\ 5 \end{bmatrix}$

Step-03:

Subtract mean vector (μ) from the given feature vectors

- $x_1 - \mu = (2-4.5,\ 1-5) = (-2.5,\ -4)$
- $x_2 - \mu = (3-4.5,\ 5-5) = (-1.5,\ 0)$
- $x_3 - \mu = (4-4.5,\ 3-5) = (-0.5,\ -2)$
- $x_4 - \mu = (5-4.5,\ 6-5) = (0.5,\ 1)$
- $x_5 - \mu = (6-4.5,\ 7-5) = (1.5,\ 2)$
- $x_6 - \mu = (7-4.5,\ 8-5) = (2.5,\ 3)$

Feature vectors (x_i) after subtracting mean vector (μ) are –

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

Step-04:
Calaulate the covariance matrix.
Covariance matrix is given by –

$$\boxed{\text{Covariance Matrix} = \frac{\sum (x_i - \mu)(x_i - \mu)'}{n}}$$

Now, $m_1 = (x_1 - \mu)(x_1 - \mu)' = \begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -2.5 & -4 \end{bmatrix} = \begin{bmatrix} 6.25 & 10 \\ 10 & 16 \end{bmatrix}$

$$m_2 = (x_2 - \mu)(x_2 - \mu)' = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -1.5 & 0 \end{bmatrix} = \begin{bmatrix} 2.25 & 0 \\ 0 & 0 \end{bmatrix}$$

$$m_3 = (x_3 - \mu)(x_3 - \mu)' = \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} -0.5 & -2 \end{bmatrix} = \begin{bmatrix} 0.25 & 1 \\ 1 & 4 \end{bmatrix}$$

$$m_4 = (x_4 - \mu)(x_4 - \mu)' = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$m_5 = (x_5 - \mu)(x_5 - \mu)' = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 1.5 & 2 \end{bmatrix} = \begin{bmatrix} 2.25 & 3 \\ 3 & 4 \end{bmatrix}$$

$$m_6 = (x_6 - \mu)(x_6 - \mu)' = \begin{bmatrix} 2.5 \\ 3 \end{bmatrix} \begin{bmatrix} 2.5 & 3 \end{bmatrix} = \begin{bmatrix} 6.25 & 7.5 \\ 7.5 & 9 \end{bmatrix}$$

Now, covariance matrix $= (m_1 + m_2 + m_3 + m_4 + m_5 + m_6)/6$
On adding the above matrices and dividing by 6, we get –

$$\text{Covariance Matrix} = \frac{1}{6} \begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$$

Step-0.5:
Calculate the eigen values and eigen vectors of the covariance matrix.
λ is an eigen value for a matrix M if it is a solution of the characteristic equation $|M - \lambda| = 0$.
So, we have –

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

From here,

$$(2.92 - \lambda)(5.67 - \lambda) - (3.67 \times 3.67) = 0$$

$$16.56 - 2.92\lambda - 5.67\lambda + \lambda^2 - 13.47 = 0$$
$$\lambda^2 - 8.59\lambda + 3.09 = 0$$

Solving the quadratic equation, we get $\lambda = 8.22,\ 0.38$

Thus, two eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$

Clearly, the second eigen value is very small compared to the first eigen value.
So, the second eigen vector can be left out.
Eigen vector corresponding to the greatest eigen value is the principal component for the given data set.
So, we find the eigen vector corresponding to eigen value λ_1.
We use the following equation to find the eigen vector.

where, M = Covariance Matrix
X = Eigen vector
λ = Eigen value

Substituting the values in the above equation, we get –

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = 8.22 \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

Solving these, we get –

$$2.92X_1 + 3.67X_2 = 8.22X_1$$
$$3.67X_1 + 5.67X_2 = 8.22X_2$$

On simplification, we get –

$$5.3X_1 = 3.67X_2 \qquad \text{.... (1)}$$
$$3.67X_1 = 2.55X_2 \qquad \text{.... (2)}$$

From Eqns. (1) & (2),

$$X_1 = 0.69X_2$$

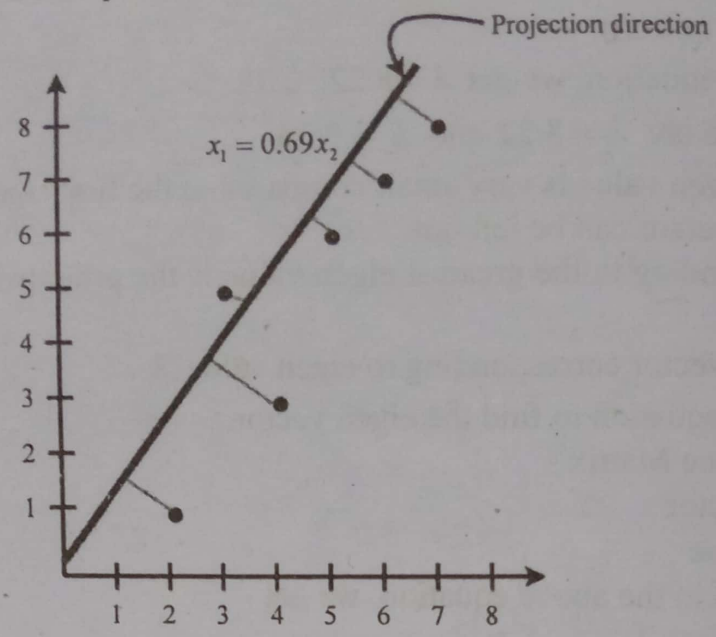
From Eqn. (2), the eigen vector is –

Eigen vector: $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

Thus, principal component for the given data set is –

Principal component: $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$

Lastly, we project the data points onto the new subspace as –



NON-PARAMETRIC TECHNIQUES FOR DENSITY ESTIMATION

Multiple Choice Type Questions

1. **Which of the following statement is False in the case of the KNN Algorithm?** **[MODEL QUESTION]**
   a) **For a very large value of K, points from other classes may be included in the neighbourhood.**
   b) **For the very small value of K, the algorithm is very sensitive to noise.**
   c) **KNN is used only for classification problem statements.**
   d) **KNN is a lazy learner.**

Answer: (c)

2. **Time to classify a new example than with a model in Knn requires?** **[MODEL QUESTION]**
   a) **Depends on Data** b) **More Time**
   c) **None of these** d) **Less time**

Answer: (b)

Short Answer Type Questions

1. **What do you understand by non-parametric techniques for density estimation?** **[MODEL QUESTION]**

Answer:
It is not a very accurate estimate of the density but we are more concerned with making the right decisions rather than estimating the density correctly.
We rely on the fact that

$$P = \text{Probability } \{ x \text{ belongs to } R \}$$

R is a region of feature space

$$P = \int p(X)\,dX$$

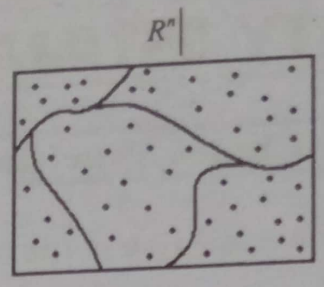
To estimate P:

$$P = \int p(X)\,dX = p(x_0)\,\text{vol}(R)$$

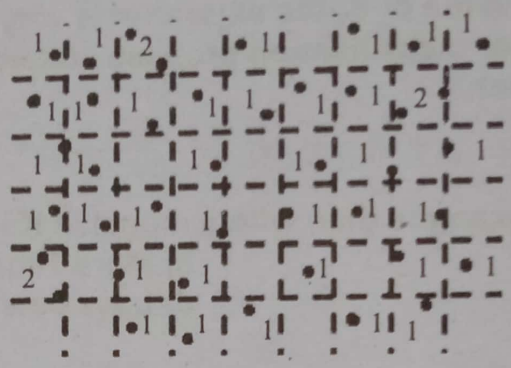
If $p(X)$ is continuous near x_0 and R is small
The idea behind this method is:
Draw samples at random $X_1, X_2, \ldots, X_d$

Divide the feature space into regions in $|khh20|$

$R^n|$



For the sake of simplicity we can divide the region into cubes with side length h.



Count how many samples fall into that region R.

$P = \Pr ob(X \text{ is in } R) = \#\text{of samples in } R/d$ where d is total number of samples

But $\quad P = \int p(X)dX = p(x_0)\text{vol}(R)|$

So, $\quad p(x_0)\text{vol}(R) = \#\text{ of samples in } R/d$

To make $p(\hat{x}_0)|$ converge to $p(x_0)$ we should

1) Fix the volume R and take a large number of samples then #of samples in R/d goes to P.

So, goes to $P/\text{vol}(R)$

Problem: $P/\text{vol}(R)$ is not $p(x_0)$!

So we should also

2) Let $\text{vol}(R)$ go to zero

Once again the problem is if we fix #of samples and let $\text{vol}(R) \to 0$ then for most region #of samples $= 0!\, sp\ p(x_0) = 0!$

So, in 1-D we get a estimate which looks like spikes.

Consider a sequence of regions containing $\vec{x}_0|$

$\{R_i\} i$ belongs to N, x_0 belongs to R_i for all i and an infinite sequence of samples

$X_1, X_2, \ldots = \{X_k\}$, k is in the N

Construct set of samples $S_i = \{X_1, X_2, \ldots, X_i\}$

P_i = Probability that $|khh23|$

$\vec{x} \in R_i|$

V_i = Volume of R_i

K_i = # of samples from S_i falling in R_i

$p_i = K_i / (i \times V_i)$

1) $\lim_{i \to \infty} = 0$ This ensures $\frac{P_i}{V}$ converge stop $(x_0)|$

2) $\lim_{i \to \infty} K_i = \infty|$

3) $\lim_{i \to \infty} K_i / i = 0|$

2. What do you know about parzen window method? **[MODEL QUESTION]**

Answer:

Pick R_i such that $V_i = 1/\sqrt{i}$

Parzen windows classification is a technique for nonparametric density estimation, which can also be used for classification. Using a given kernel function, the technique approximates a given training set distribution via a linear combination of kernels centered on the observed points. In this work, we separately approximate densities for each of the two classes, and we assign a test point to the class with maximal posterior probability.

The resulting algorithm is extremely simple and closely related to support vector machines. The Parzen windows classification algorithm does not require any training phase; however, the lack of sparseness makes the test phase quite slow. Furthermore, although asymptotical convergence guarantees on the performance of Parzen windows classifiers exist, no such guarantees exist for finite sample sizes.

Parzen windows can be regarded as a generalization of k-nearest neighbour techniques. Rather than choosing the k nearest neighbours of a test point and labelling the test point with the weighted majority of its neighbours' votes, one can consider all points in the voting scheme and assign their weight by means of the kernel function. With Gaussian kernels, the weight decreases exponentially with the square of the distance, so far away points are practically irrelevant. The width σ of the Gaussian determines the relative weighting of near and far points. Tuning this parameter controls the predictive power of the system. We have empirically optimized the value.

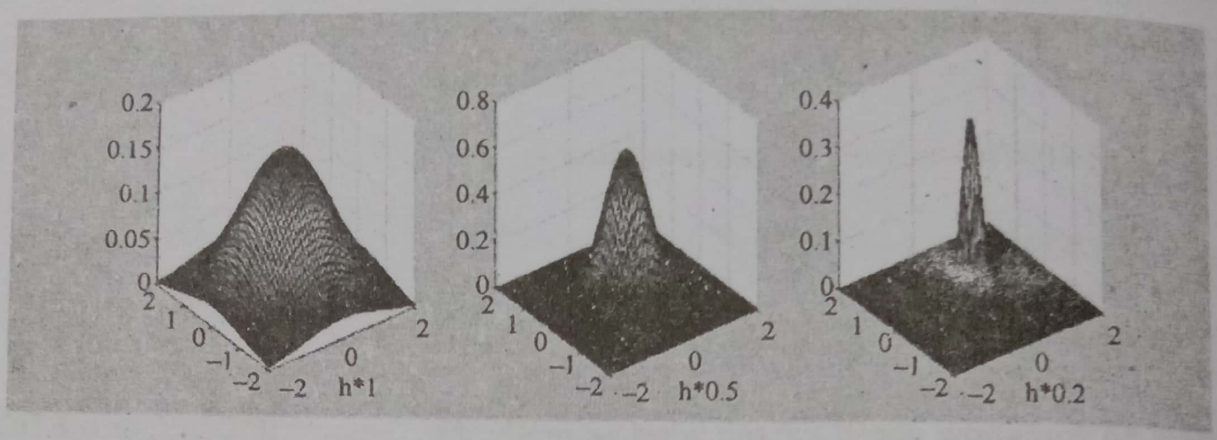
Examples of Parzen Windows

The parzen window is defined by $|\text{parzen_window}|\delta(x) = \frac{1}{V}\varphi\left(\frac{x}{h}\right)|$

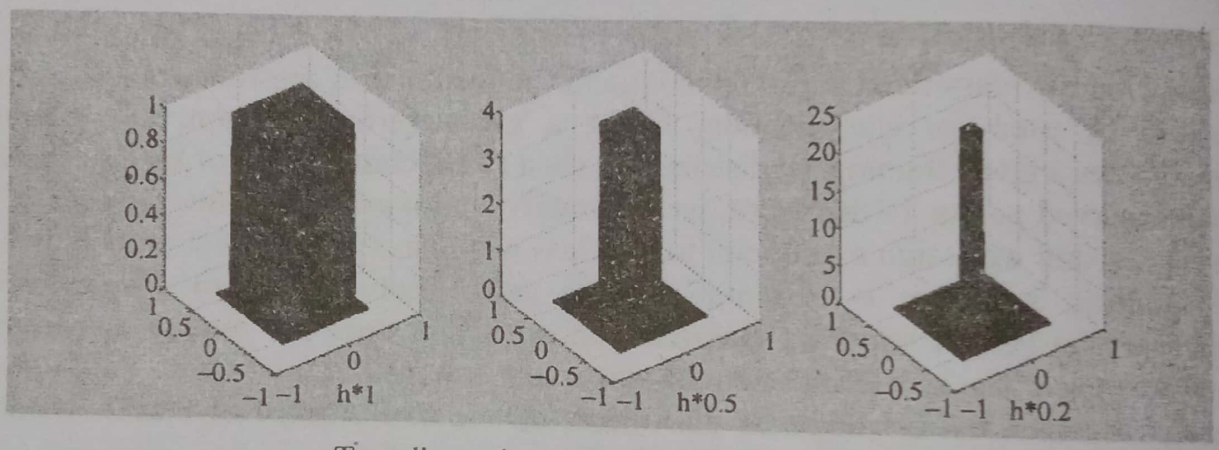
where, $|V|$

$V = h^d|$

d is the number of dimensions.
The values of 'h' parameter are chosen to be 1, 0.5 and 0.2
d is chosen to be 2.



Two dimensional circularly symmetric normal Parzen Window



Two dimensional square Parzen Window

3. What is K-Nearest Neighbours method? **[MODEL QUESTION]**

Answer:

Pick R_i such that $V_i = 1/\sqrt{i}$

The k-nearest neighbour algorithm is amongst the simplest of all machine learning algorithms. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common amongst its k nearest neighbours. k is a positive integer, typically small. If $k = 1$, then the object is simply assigned to the class of its nearest neighbour. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes.

The naive version of the algorithm is easy to implement by computing the distances from the test sample to all stored vectors, but it is computationally intensive, especially when the size of the training set grows. Many nearest neighbour search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed. The nearest neighbour algorithm has some strong consistency results. As the amount of data approaches infinity, the algorithm is

guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data). k-nearest neighbour is guaranteed to approach the Bayes error rate, for some value of k (where k increases as a function of the number of data points).

4. Compare parametric and non-parametric method. [MODEL QUESTION]

Answer:

It's worth to note that it's not one hundred percent right when we say that non-parametric methods are "model-free" or free of distribution assumptions. Take nearest neighbour for example, some kind of distance measure has to be used to identifying the "nearest" neighbour. Although the method didn't assume a specific distribution, the distance measure is distribution-related in some sense. (Euclidean and Mahalanobis distances are closely related to multivariate Gaussian distribution.) Compared to parametric methods, non-parametric ones only "vaguely" or "remotely" related to specific distributions and, therefore, are a lot flexible and less sensitive to violation of distribution assumptions.

Another characteristic found to be helpful in discriminating the two is that: the number of parameters in parametric models is fixed a priori and independent of the size of the dataset, while the number of statistics used for non-parametric models are usually dependent on the size of the dataset (e.g. more statistics for larger datasets).

5. What are advantages and disadvantages of no-parametric method? [MODEL QUESTION]

Answer:

Advantages of Non-parametric Estimation Methods

- Same procedure can be used for unimodal normal and bimodal mixture.
- We do not need to make assumption about the distribution ahead of time.
- With enough samples, we are assured of convergence to an arbitrarily complicated target density

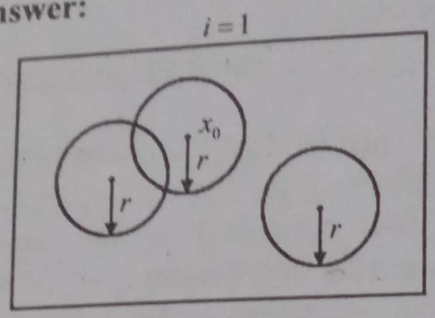
Disadvantages of Non-parametric Estimation Methods

- The cardinality of the sample set should be very large (much larger than the cardinality when the form of density is known) .
- Very high computation time and huge storage required.
- Curse of dimensionality (see [Curse of Dimensionality])
- These methods are very sensitive to the choice of the window size (If too small, most of the volume will be empty, and the estimate will be erratic, if too large: important variations may be lost due to averaging.)
- It may be the case that a cell volume appropriate for one region of the feature space might be entirely unsuitable in a different region.

Long Answer Type Questions

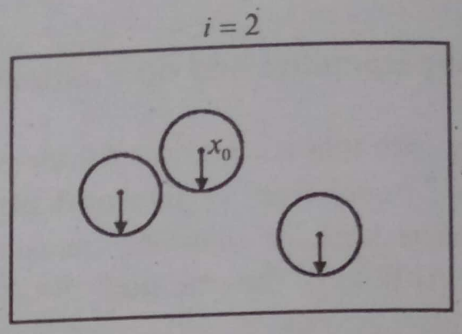
1. Give a visual comparison for Parzen windows. **[MODEL QUESTION]**

Answer:



$$\pi r^2 = \frac{1}{\sqrt{i}}$$

$$r^2 = \frac{1}{\pi}$$



$$r^2 = \frac{1}{\pi\sqrt{2}}$$

2. Explain Parzen-window and k-nearest neighbour method. **[MODEL QUESTION]**

Answer:

$$p(x) \approx \frac{k}{NV} \qquad \text{.... (1)}$$

Parzen Window Method:

The Parzen windown method assumes that the local region around data point x is a hypercube with edge length h, then the volume of region is $V = h^d$. The window function of unit hyercube is defined by

$$H(u) = \begin{cases} 1 & |u_i| < 1/2,\ i = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases} \qquad \text{.... (2)}$$

Given a ser of training data points $X = \{x^1, \ldots, x^N\}$, the number of points falling in the window around a new vector x is

$$k = \sum_{n=1}^{N} H\left(\frac{x - x^n}{h}\right) \qquad \text{.... (3)}$$

Substituting this number into Eqn. (1) leads to the density estimate around x:

$$\hat{p}(x) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{h^d} H\left(\frac{x - x^n}{h}\right) \qquad \text{.... (4)}$$

Expectation of the estimated density in Eqn. (4) is a smoothed version of the true density:

$$\varepsilon\left[\hat{p}(x)\right] = \frac{1}{N}\sum_{n=1}^{N} \varepsilon\left[\frac{1}{h^d} H\left(\frac{x - x^n}{h}\right)\right]$$

$$= \varepsilon\left[\frac{1}{h^d} H\left(\frac{x - x'}{h}\right)\right]$$

$$= \int \frac{1}{h^d} H\left(\frac{x - x'}{h}\right) p(x')\,dx' \qquad \text{.... (5)}$$

The kernel width h plays the role of a smoothing parameter and should be carefully selected to compromise the bias and variance. On finite training sample size, a small

value of h leads to an overfitting of training data, whereas a large value leads to a biased estimation of density.

The window function can be extended to other forms of function provided that $H(x) \geq 0$ and $\int H(x)dx = 1$ are satisfied. The standard Gaussian function is a natural choice of window function of infinite support:

$$H(u) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{\|u\|^2}{2}\right) \quad \text{.... (6)}$$

Acordingly, the density estimate of Eqn. (4) is

$$\hat{p}(x) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{d/2}} \exp\left(-\frac{\|x - x^n\|^2}{2h^2}\right) \quad \text{.... (7)}$$

For classification, the formula Eqn. (4) or (7) can be used to estimate the conditional density around x for each class and then the Bayes decision rule is applied. The value of h can be selected by cross-validation such that the selected value leads to a high classification accuracy on a validation data set disjoint from the training set.

A serious problem with the Parzen sindow method is that all the training samples should be stored for density estimation and classification. A way to alleviate the storage and computation overhead is to model the density using a small number of kernel functions centered at selected pointsl. This way is similar to the semiparametric density estimation method and if the Gaussian window function is used, the center points can be estimated by the EM algorithm.

K-Nearest Neighbour Method:

In the k-NN method, the volume of local region $\Re$ for density estimation is variable whereas the number k of data points falling in $\Re$ is fixed. Given N training data points from M classes, for estimating the local density around a new vector x, a window centered at x is enlarged such that exactly k nearest data points fall in the window. Assume that the k nearest neighbours include k_i points from class $\omega_i, i = 1, \ldots, M, \sum_{i=1}^{M} k_i = k$, the class-conditional density around x is estimated by

$$\hat{p}(x \mid \omega_i) = \frac{k_i}{N_i V}, \; i = 1, \ldots, M \quad \text{.... (8)}$$

where, N_i is the number of training points of class ω_i. The a posteriori probabilities are computed by the Bayes formula:

$$p(\omega_i \mid x) = \frac{P(\omega_i)\hat{p}(x \mid \omega_i)}{\sum_{j=1}^{M} P(\omega_j)\hat{p}(x \mid \omega_j)} = \frac{P(\omega_i)k_i / N_i}{\sum_{j=1}^{M} P(\omega_j)k_i / N_i}, \; i = 1, \ldots, M \quad \text{.... (9)}$$

Considering $\hat{P}(\omega_i) = \frac{N_j}{N}$, the computation of a posteriori probabilities is simplified to

$$p(\omega_i \mid x) = \frac{k_i}{k}, \quad i = 1, \ldots, M \qquad \ldots (10)$$

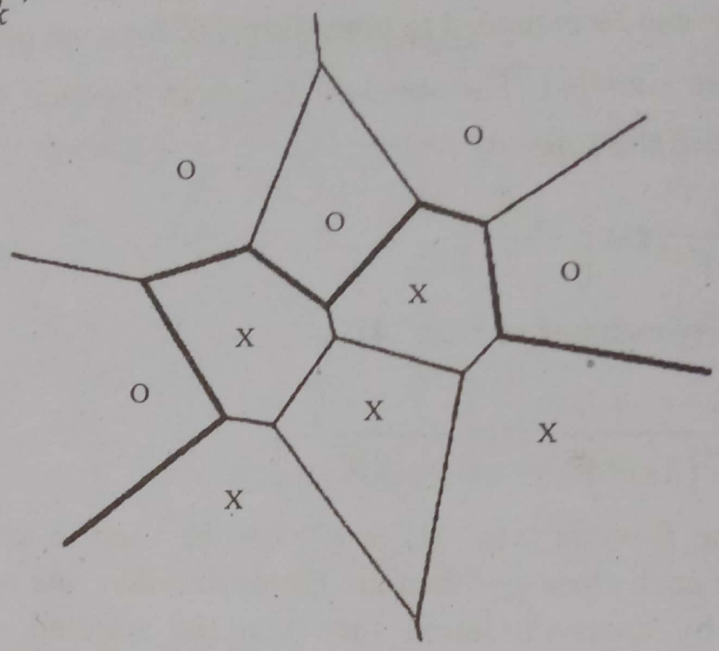


Fig: 1 Voronoi diagram formed by training samples of two classes.
The decision surface is denoted by thick lines

This is to say, the a posteriori probability of a class is simply its fraction of k nearest neighbours in training samples and the decision is to select the class with most nearest neighbours among k. This decision rule is called k-NN rule.

When k is set equal to one, the k-NN rule is reduced to the nearest neighbour (1-NN) rule that classifies the input pattern x to the class of the nearest training sample. When the distance measure is the Euclidean distance, the decision regions of the 1-NN rule in feature space form a Voronoi diagram (Voronoi tessellation), with each cell defined by a training sample and separated from each other by a hyperplane that is equidistant from two samples. An example of Voronoi diagram is shown in Fig. 1.

It shows that when the number of training samples approaches infinity, the classification error rate of 1-NN rule is bounded by two times the Bayes error rate. More exactly, denoting the Bayes error rate by P^*, the error rate of 1-NN rule is bounded by

$$P^* \le P \le P^*\left(2 - \frac{M}{M-1}P^*\right) \qquad \ldots (11)$$

As for the Parzen window method, the drawback of k-NN and 1-NN methods is that all the training samples are required to be stored and matched in classification. Both the Parzen window and nearest neighbour methods are not practical for realtime applications, but because of their fairly high classification performance, they serve as good benchmarks for evaluating other classifiers. There exist many efforts aiming to reduce the storage and computation overhead of nearest neighbour methods.

LINEAR DISCRIMINANT FUNCTION BASED CLASSIFIER

Multiple Choice Type Questions

1. The effectiveness of an SVM depends upon: [MODEL QUESTION]
   a) Selection of Kernel
   b) Kernel Parameters
   c) Soft Margin Parameter C
   d) All of these

Answer: (d)

2. The cost parameter in the SVM means: [MODEL QUESTION]
   a) The number of cross-validations to be made
   b) The kernel to be used
   c) The tradeoff between misclassification and simplicity of the model
   d) None of the above

Answer: (c)

3. Which of the following are real world applications of the SVM? [MODEL QUESTION]
   a) Text and Hypertext Categorization
   b) Image Classification
   c) Clustering of News Articles
   d) All of these

Answer: (d)

4. What is perceptron? [MODEL QUESTION]
   a) a single layer feed-forward neural network with pre-processing
   b) an auto-associative neural network
   c) a double layer auto-associative neural network
   d) a neural network that contains feedback

Answer: (a)

Short Answer Type Questions

1. What is perceptron? [MODEL QUESTION]

Answer:

A perceptron is a simple model of a biological neuron in an artificial neural network. Perceptron is also the name of an early algorithm for supervised learning of binary classifiers.

The perceptron algorithm was designed to classify visual inputs, categorizing subjects into one of two types and separating groups with a line. Classification is an important part of machine learning and image processing. Machine learning algorithms find and classify patterns by many different means. The perceptron algorithm classifies patterns and groups by finding the linear separation between different objects and patterns that are received through numeric or visual input.

The perceptron algorithm was developed at Cornell Aeronautical Laboratory in 1957, funded by the United States Office of Naval Research. The algorithm was the first step planned for a machine implementation for image recognition. The machine, called Mark 1 Perceptron, was physically made up of an array of 400 photocells connected to perceptrons whose weights were recorded in potentiometers, as adjusted by electric motors. The machine was one of the first artificial neural networks ever created.

2. What is perceptron example? **[MODEL QUESTION]**

Answer:

The perceptron is the building block of artificial neural networks , it is a simplified model of the biological neurons in our brain. A perceptron is the simplest neural network, one that is comprised of just one neuron. The perceptron algorithm was invented in 1958 by Frank Rosenblatt.

3. What is a perceptron unit? **[MODEL QUESTION]**

Answer:

A single-layer perceptron is the basic unit of a neural network . A perceptron consists of input values, weights and a bias, a weighted sum and activation function. In the last decade, we have witnessed an explosion in machine learning technology.

4. What are Support Vector Machines? **[MODEL QUESTION]**

Answer:

Support Vector Machine (SVM) is a relatively simple **Supervised Machine Learning Algorithm** used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.

In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems.

Long Answer Type Questions

1. What do you understand by discriminant function? **[MODEL QUESTION]**

Answer:

A classification method always starts with a pair of variables (x, y). The first variable $x \in X \subset \mathbb{R}^d$ is an **input vector**. The set X is called the **input space**. Depending on the application, an input could be intensity of a pixel, a wavelet coefficient, the phase angle of wave, or anything we want to model. We assume that each input vector has a dimensionality d and d is finite. The second variable $y \in \{1, \cdots, k\} = Y$ denotes the

class lebel of the classes $\{C_1, \ldots, C_k\}$. The choice of the class levels is arbitrary. They do not need to be positive integers. For example, in binary classification, one can choose $y \in \{0,1\}$ or, $y \in \{-1,+1\}$, depending which one is more convenient mathematically.

In supervised learning, we assume that we have a **training set** D. A training set is a collection of paired variables (x_j, y_j), for $j = 1, \ldots, n$. The vector x_j denotes the j-th sample input in D and y_j denotes the corresponding class label. The relationship between x_j and y_j is specified by the **target function** $f : X \to Y$ such that $y_j = f(x_j)$. The target function is **unknown**. By unknown we really mean that it is unknown.

The training set D can be generated deterministically or probabilistically. If D is deterministic, then $\{x_j\}_{j=1}^{n}$ is a sequence of fixed vectors. If D is probabilistic, then thee is an underlying generative model that generates $\{(x_j)\}_{j=1}^{n}$. The generative model is specified by the distribution $p_X(x)$. The distinction between a deterministic and a probabilistic model is stable but important.

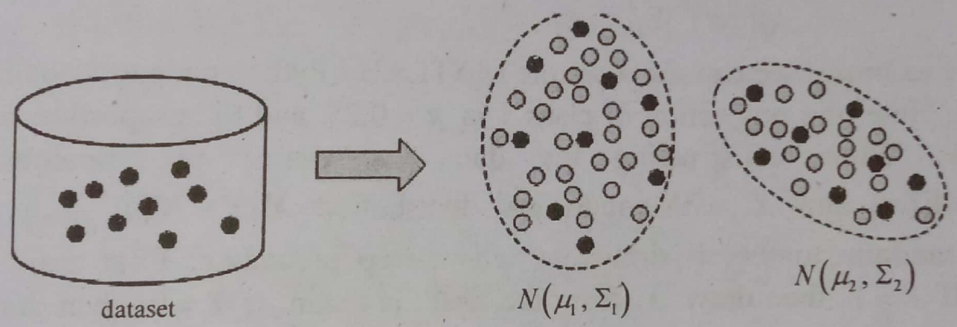


Fig: 1 Visualizing a training data D consisting of two classes C_1 and C_2, with each class being a multidimensional Gaussian. The gray dots represent the samples **outside** the training set, whereas the colour dots represent the samples **inside** the training set.

For now, let us assume that x_j 's are generated probabilistically.

When there is no noise in creating the labels, the class lebel y_j is defined as $y_j = f(x_j)$. However, in the presence of noise, instead of using a fixed but unknown target function f to generate $y_j = f(x_j)$, we assume that y_j is drawn from a posterior distribution $y \sim p_{Y|X}(y|x)$. As a result, the training pair (x_j, y_j) is now drawn according to the **joint distribution** $p_{X,Y}(x, y)$ of random variables X and Y. By Bayes Theorem, it holds that $p_{X,Y}(x, y) = p_{Y|X}(y|x)\, p_X(x)$, which means the joint distribution $p_{X,Y}(x, y)$ can be completely determined by the posterior distribution $p_{Y|X}(y|x)$ and the data generator $p_X(x)$.

Example: [Gaussian]. Consider a training set D consisting of training samples $\{(x_j, y_j)\}$. The input vector x_j is a d-dimensional vector in $\mathbb{R}^2$ and the label y_j is either {1, 2}. The target function f is a mapping which assigns each x_j to a correct y_j. The left hand side of Fig. 1 shows a labeled dataset, marking the data points in red and blue.

We assume that for each class, the input vectors are distributed according to a d-dimensional Gaussian:

$$p_{X|Y}(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left\{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right\} \quad \text{.... (1)}$$

where, $\mu_i \in \mathbb{R}^d$ is the mean vector and $\Sigma_i \in \mathbb{R}^{d\times d}$ is the covariance matrix. If we assume that $p_Y(i) = \pi_i$ for $i = 1, 2$, then the joint distribution of X and Y is defined through the Bayes Theorem.

$$p_{X,Y}(x,i) = p_{X|Y}(x|i)\, p_Y(i)$$

$$= \pi_i \cdot \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left\{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right\}$$

In the above example, we can use a simple MATLAB / Python code to visualize the joint distribution. Here, the proportion of class 1 is $\pi = 0.25$ and the proportion of class 2 is $1-\pi = 0.75$. When constructing the data, we define two random variables $Y \sim \text{Bernoulli}(\pi)$ and X with conditional distribution $X|Y \sim N(\mu_Y, \Sigma_Y)$. Therefore, the actual random number is drawn by a two-step procedure: First draw Y from a Bernoulli. If $Y = 1$, then draw X from the first Gaussian. If $Y = 0$, then draw X from the other Gaussian.

```
mu1    = [0 0];
mu0    = [2 0];
Sigma1 = [.25 .3; .3 1];
Sigma0 = [0.5 0.1; 0.1 0.3];
n      = 1000;              % number of samples
p      = 0.25;              % pi
y      = (rand(n, 1)<=p);% class label
idx1   = (y==1);            % indices of class 1
idx0   = (y==0);            % indices of class 0
x      = mvnrnd(mu1, Sigma1, n).*repmat(y,[1,2]) + ...
         + mvnrnd(mu0, Sigma0, n).*repmat((1-y),[1,2]);
```

The plotting of the data points can be done by a scatter plot. In the code below, we use the indices idx1 and idx0 to identify the class 1 and class 0 data points. This is an afternath labe. When x was first generated, it was unlabeled. Therefore, in an unsupervised learning case, there will not be any colouring of the dots.

```
figure(1);
```

```
scatter(x(idx1,1),x(idx1,2),'rx', 'LineWidth', 1.5); hold on;
scatter(x(idx0,1),x(idx0,2),'bo', 'LineWidth', 1.5); hold off;
xlabel('x'); ylabel('y');
set(gcf, 'Position', [100, 100, 600, 300]);
```

The result of this example is shown in Fig. 2. If we count of the number of red and blue markers, we see that there are approximately 25% red and 75% blue. This is a result of the prior distribution. Shown in this plot is a linearly not separable example, meaning that some data points will never be classified correctly by a linear classifier even if we use the classifier is theoretically optimal.

One thing we need to be careful in the above example is that the training samples $\{(x_j, y_j)\}_{j=1}^{n}$ represent a finite set of n samples drawn from the distribution $p_{X,Y}(x, y)$. The distribution $p_{X,Y}(x, y)$ is "bigger" in the sense that many samples in $p_{X,Y}(x, y)$ are not necessarily part of D. Think about doing a survey in the United States about a person's height. The distribution $p_{X,Y}(x, y)$ covers the entire population, whereas D is only a sub-collection of the data resulting from the survey. Samples that are inside the training set are called the **in-samples** and samples that are ouside the training set are called the **out-samples**. See Fig. 1 for an illustration.

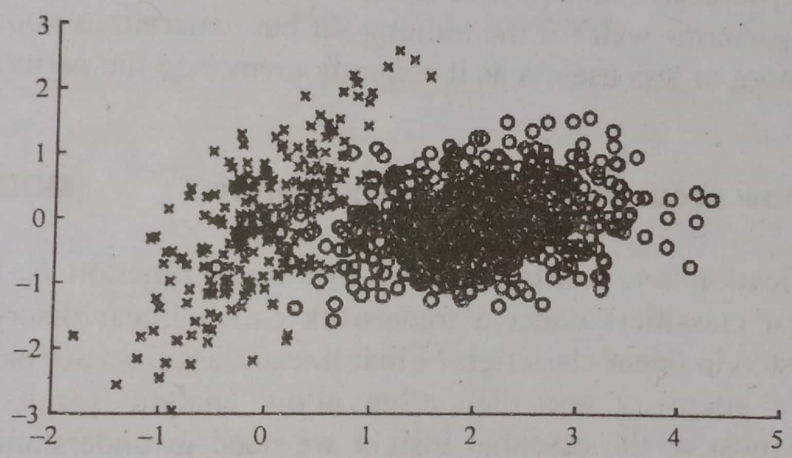


Fig: 2 Generating random samples from two Gaussians where the class label is generated according to a Bernoulli distribution

The difference between in-sample and out-sample is important. Straightly speaking, the Gaussian parameters (μ_1, Σ_1) in the previous example are called the population paramketer and they are unknown no matter how many people we have surveyed unless we have asked everyone. Given the training set D, any model parameter estimated from D is the sample parameter $(\hat{\mu}_1, \hat{\Sigma}_1)$.

Given the training dataset D, the goal of learning (in particular classification) is to pick a mapping $h: X \to Y$ that can minimize the error of misclassification. The mapping h is called a **hypothesis function**. A hypothesis function could be linear, nonlinear, convex, non-convex, expressible as equations or an algorithm like a deep neural network. The set

containing all candidate hypothesis functions is called the **hypothesis set** H. See Fig. 3 for an illustration.

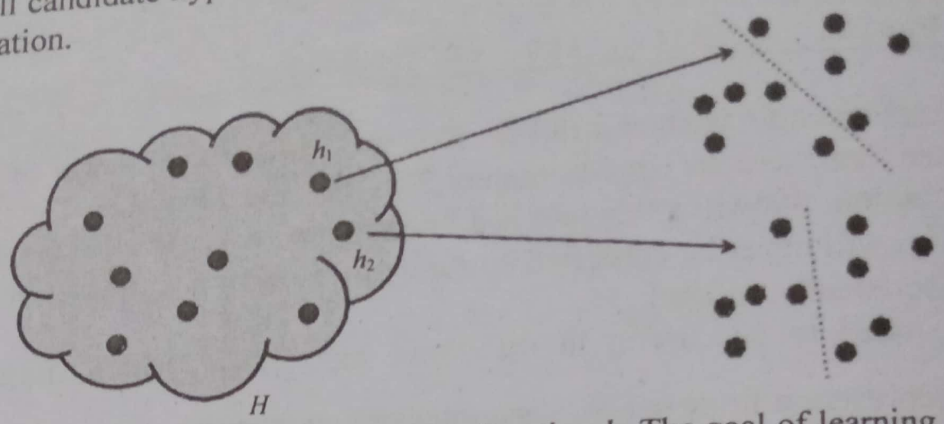


Fig: 3 Hypothesis set H and hypothesis function h. The goal of learning is to use an algorithm to find a hypothesis function or approximate a hypothesis function such that the error of classification is minimized. In this example, h_2 is a better hypothesis because it has a lower misclassification rate.

So what is "error' in classification? To be clear about this, we are fundamentally asking two questions (i) How well does the hypothesis function h do for the training dataset, i.e., the in-samples? (ii) How well does it generalize to the testing dataset, i.e., the out-samples? A gppd hypothesis function should minimize the error for both. If a hypothesis function that only performs well for the training set but generalizes poorly to the testing dataset, then it is more or less useless as it can only memorize the pattern but not predict the pattern.

2. What do you know about linear discriminant analysis? [MODEL QUESTION]

Answer:

The goal of classification is to construct a good hypothesis function h. In this course we will focus on linear classifiers under a framework called linear discriminant analysis. Why are we interested in linear classifiers? First, linear classifiers are easy to understand. They have somple geometry and they often allow analytic results. Second, linear classifiers contain most of the essential insight we need to understand a classification method and the principle is generalizable to nonlinear classifiers.

So what is a linear classifier? Let us consider a simple binary {0, 1} classification problem. The hypothesis function h of a binary classification problem takes the form of

$$h(x)=\begin{cases}1, & g(x)>0\\ 0, & g(x)<0\end{cases}$$

Here, the function $g:X\to\mathbb{R}$ os called a **discriminant function**. The job of the discriminant function g is to map the vector x to a value $g(x)$ such that the class can be determined by checking the sign of $g(x)$. How does g look like? For linear classifiers, g take a linear form and is called a linear discriminant function.

Definition 1: (Linear Discriminant Function). A **linear discriminant function** is a mapping $g: X \to \mathbb{R}$ with

$$g(x) = w^T x + w_0$$

for some vectors $w \in \mathbb{R}^d$ and scalar $w_0 \in \mathbb{R}$.

In this definition, the vector w is called the weight of the linear classifier and w_0 is called the bias of the classifier. To simplify the notation, we sometimes concatenate w and w_0 by defining $\theta = \{w, w_0\}$. We call $\theta \in \mathbb{R}^{d+1}$ the model parameter of the linear discriminant function g. For different dimensionality d, the geometry of $g(x)$ changes. Fig. 1 shows the examples in 1D, 2D and 3D.

A discriminant function does not need to be linear. For example, we can construct a

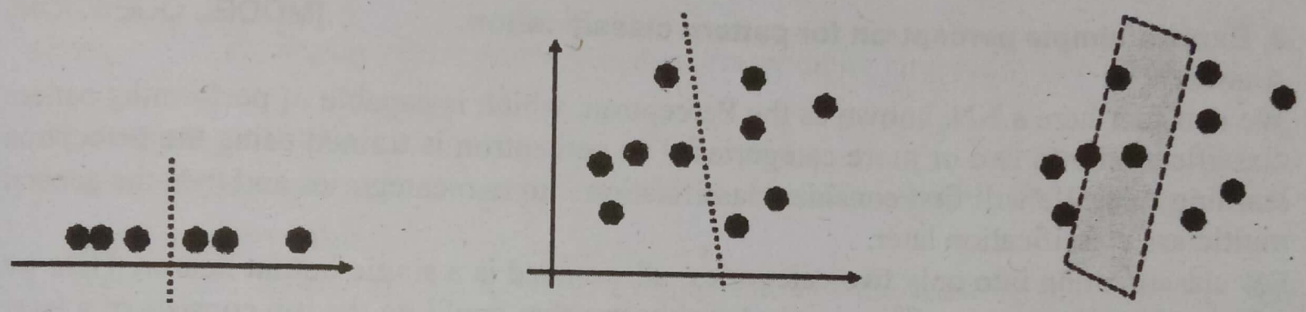


Fig: 1 Discriminant function $g(x)$ at different dimensions. [Left] In 1D, the discriminant function is simply a cutoff value. [Middle] In 2D, the discriminant function is a line. [Right] In 3D, the discriminant function is a plane.

Definition 2: (Quadratic Discriminant Function). A **quadratic discriminant function** is a mapping $g: X \to \mathbb{R}$ with

$$g(x) = \frac{1}{2} x^T W x + + w^T x + w_0$$

for some matrix $W \in \mathbb{R}^{d \times d}$, some vector $w \in \mathbb{R}^d$ and some scalar $w_0 \in \mathbb{R}$.

In quadratic discriminant function, the model parameter is $\theta = \{W, w, w_0\}$. Depending on W, the geometry of g could be convex, concave or neither. Fig. 2 below shows a quadratic discriminant function separating an inner and an outer cluster of data points.

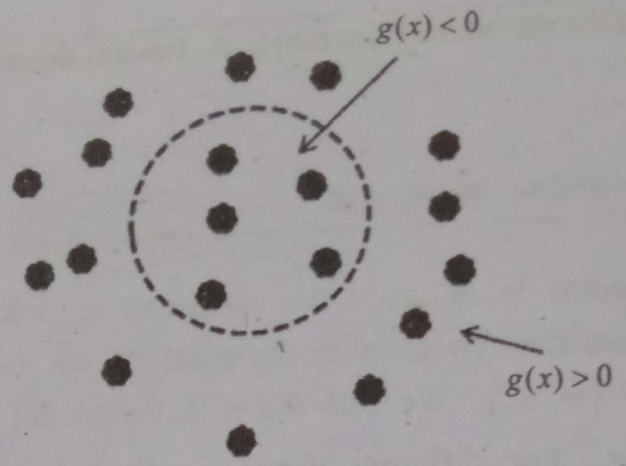


Fig: 2 A quadratic discriminant function is able to classify data using quadratic surfaces. This example shows an ellipsoid surface for separating an inner and outer cluster of data points.

3. Explain simple perceptron for pattern classification. **[MODEL QUESTION]**

Answer:

We consider here a NN, known as the Perceptron, which is capable of performing pattern classification into two or more categories. The perceptron is trained using the perceptron learning rule. We will first consider classification into two categories and then the general multiclass classification later.

For classification into only two categories, all we need is a single output neuron. Here we will use bipolar neurons. The simplest architectre that could do the job consists of a layer of N input neurons, an output layer with a single output neuron and no hidden layers. This is the same architecture as we saw before the Hebb learning.

However, we will use a different transfer function here for the output neuron:

$$f_\theta(x) = \begin{cases} +1, & \text{if } x > \theta, \\ 0, & \text{if } -\theta \le x \le \theta, \\ -1 & \text{if } x < -\theta. \end{cases}$$

This transfer function has an undecided band of width 2θ. The value of θ is held fixed at a relatively small value. The undecided case has an output of 0, exactly half way between 1 and –1.

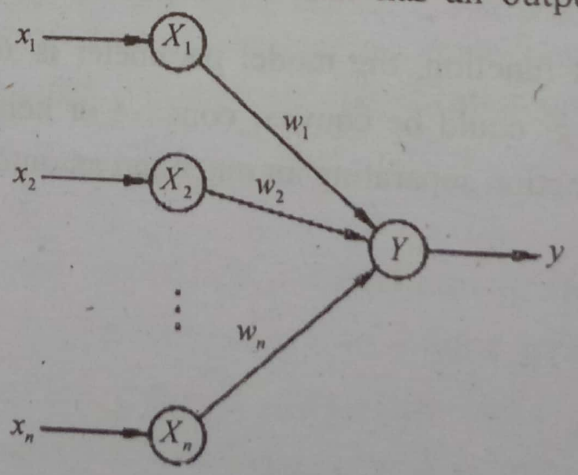


Fig: 1 A feed forward NN having 1 output neuron

Thus if we represent the N components of the input vector by x, the N components of the weight vector by w and the bias by b, the output is then given by

$$y = f_\theta(xw + b)$$

Notice that because of the form of the transfer function, the threshold, θ, can no longer be absorbed by the bias, b. However, we can still introduce a fictitious input neuron X_0 whose activation is always given by $x_0 = 1$ and its connecting weight w_0 with the output neuron Y is defined to be b.

The final question is how are we going to find the set of weights and bias to solve a given problem? For this prpose we will be using the Perceptron learning rule. Just like Hebb's rule, the Perceptron learning rule is also a supervised learning rule. Thus we assume that we are given a training set:

$$\left\{s^{(q)}, t^{(q)}\right\},\ q = 1, 2, \ldots, Q$$

where, $s^{(q)}$ is a training vector and $t^{(q)}$ is its corresponding targeted output value.

The Perceptron rule is:

1. Set learning rate $\alpha(>0)$ and threshold θ. Initialize weights and bias to zero (or some random values).
2. Repeat the following steps, while cycling through the training set $q = 1, 2, \ldots, Q$, until all training patterns are classified correctly
   (a) Set activations for input vector $x = s^{(q)}$.
   (b) Compute total input for the output neuron:
   $$y_{\text{in}} = xw + b$$
   (c) Compute the output
   $$y = f_\theta(y_{\text{in}})$$
   (d) Update the weights and bias only if that pattern is misclassified
   $$w^{\text{new}} = w^{\text{old}} + \alpha t^{(q)} x$$
   $$b^{\text{new}} = b^{\text{old}} + \alpha t^{(q)}$$

Notice that when a training vector is presented to the NN and the output agrees with the targeted value, then there is no change to the weights and bias. There is a common saying: when things are working don't change them. Also notice that by the design of the algorithm, the NN always correctly classify all the training vectors using the weights and bias obtained from the algorithm.

Application: Bipolar Logical Function: AND

The training set is given by the following table:

q	$s^{(q)}$	$t^{(q)}$
1	$[1 \quad 1]$	1
2	$[1 \quad -1]$	-1
3	$[-1 \quad 1]$	-1
4	$[-1 \quad -1]$	-1

We assume that the weights and bias are initially zero and the threshold is 0.2. We choose a learning rate $\alpha = 1$. We obtain the following results by applying the Perceptron learning rule.

Step	$s^{(q)}$	y_{in}	y	$t^{(q)}$	Δw	Δb	w	b
1	$[1 \quad 1]$	0	0	1	$[1 \quad 1]$	1	$[1 \quad 1]$	1
2	$[1 \quad -1]$	1	1	-1	$[-1 \quad 1]$	-1	$[0 \quad 2]$	0
3	$[-1 \quad 1]$	2	1	-1	$[1 \quad -1]$	-1	$[1 \quad 1]$	-1
4	$[-1 \quad -1]$	-3	-1	-1	$[0 \quad 0]$	0	$[1 \quad 1]$	-1
5	$[1 \quad 1]$	1	1	1	$[0 \quad 0]$	0	$[1 \quad 1]$	-1
6	$[1 \quad -1]$	-1	-1	-1	$[0 \quad 0]$	0	$[1 \quad 1]$	-1
7	$[-1 \quad 1]$	-1	-1	-1	$[0 \quad 0]$	0	$[1 \quad 1]$	-1

Notice that in step 4 of the loop, the weights and bias do not change. However, we still have to present the other 3 training vectors to the NN to make sure that they are also classified correctly and no change in the weights and bias need to be made. After step 7, it is clear that the situation will repeat itself in groups of 4 and thus we can exit the loop. Therefore, in general the loop in the Perceptron algorithm should be stopped when the weights and bias do not change for Q consecutive times.

The resulting set of weights, $w = [1 \quad 1]$ and bias, $b = -1$, given the decision boundary determined by

$$x_1 w_1 + x_2 w_2 = x_1 + x_2 = 1$$

This boundary is a straight line given by

$$x_2 = 1 - x_1$$

which has a slope of -1 and passes through the point $[1 \quad 0]$. This is the best decision boundary for this problem in terms of robustness (the training vectors are farthest from the decision boundary). Thus in this example, the Perceptron learning algorithm converges to a set of weights and bias that is the best choice for this NN.

However, this is all quite fortuitous. In general we cannot expect the Perceptron learning algorithm to converge to a set of weights and bias that is the best choice for any given

NN. It is easy to see that because if we has assumed some none zero values for the initial weights or bias, in all likelihood we would not have gotten the best decision boundary for this problem.

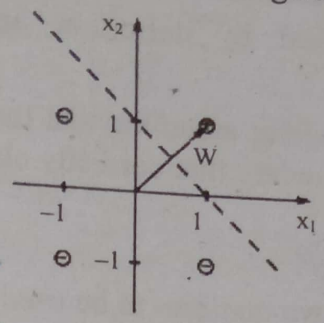


Fig: 2 The Perceptron rule finds the best decision boundary for the case of the AND function if the initial weights and bias were chosen to be zero.

4. Explain convergence theorem for perceptron learning rule. [MODEL QUESTION]

Answer:

To consider the convergence theorem for the Perceptron Learning Rule, it is convenient to absorb the bias by introducing an extra input neuron, X_0, whose signal is always fixed to be unity.

Convergence Theorem for the Perceptron Learning Rule: For a Perceptron, if there is a correct weight vector $w*$ that gives correct response for all training patterns, that is

$$f_\theta\left(s^{(q)} \cdot w*\right) = t^{(q)}, \qquad \forall q = 1, \ldots, Q$$

then for any starting weight vector w, the Perceptron learning rule will converge in a finite number of steps to a correct weight vector.

According to the theorem, a table like the one given above for the NN functioning as an AND gate, must terminate after a finite number of steps. Note that the vectors appearing in the second column are all chosen from the training set. The same vector may appear more that once in that column. First we can ignore from that column those vectors that are classified correctly at the particular point in the loop, since they lead to no changes in the weights or the bias. Next, we consider those vectors in that column, say $s^{(q)}$, whose target output is $t^{(q)} = -1$. This means that we want $s^{(q)} \cdot w < -\theta$ so that the output $y = -1$. Of we multiply this inequality by minus one, we have $-s^{(q)} \cdot w > \theta$. We can reverse the signs of all these vectors so that we want their target output to be +1.

We denote the list of these vectors by F. The individual vectors are denoted by $x(k)$, so that

$$F = \{x(0), x(1), \ldots, x(n), \ldots\}.$$

Out goal is to prove that this list must terminate after a finite number of steps and therefore, contains only a finite number of vectrs. By design, all these vectors have target outputs of +1 and they are all incorrectly classified at their own step in the loop.

The initial weight is $w(0)$ and at step n (not counting those steps where the training vectors presented to the NN are classified correctly) the weight vector is $w(n)$. Therefore for any $n, x(n)$ is misclassified by definition and therefore we always have $x(n) \cdot w(n) < 0$.

We let w^* be a correct weight vector assuming that there is such a vector. Then by the fact that w^* gives a decision boundary that correctly classifies all the training vectors in F, we have

$$x(n) \cdot w^* > 0, \quad \forall x(n) \in F$$

We need to define the following two quntities to be used in the proof:

$$m = \min_{x(n) \in F} x(n) \cdot w^*,$$

$$M = \max_{x(n) \in F} \|x(n)\|^2.$$

Notice that both quantities are strictly positive.

Now we want to prove that the number of steps in the Perceptron learning rule is always finite provide that w^* exists.

For this proof, we assume that $\theta = 0$.

The first step in the loop of the Perceptron learning rule gives

$$w(1) = w(0) + \alpha x(0)$$

and the second step gives

$$w(2) = w(1) + \alpha x(1) = w(0) + \alpha x(0) + \alpha x(1)$$

Thus after k steps, we have

$$w(k) = w(0) + \alpha \sum_{n=0}^{k-1} x(n)$$

We take the dot-product of this equation with w^* to give

$$w(k) \cdot w^* = w(0) \cdot w^* + \alpha \sum_{n=0}^{k-1} x(n) \cdot w^*$$

But by the defintion of m, we have $x(n) \cdot w^* \geq m$ for any n. So, we have the inequality

$$w(k) \cdot w^* > w(0) \cdot w^* + \alpha \sum_{n=0}^{k-1} m = w(0) \cdot w^* + \alpha k m$$

Notice that in the above relation, equality is not possible because there is at least one vector $x(p)$ in the sum such that $x(p) \cdot w^* > m$.

Next, we make use of the Cauchy-Schwartz inequality:

$$\|a\|^2 \|b\|^2 \geq (a \cdot b)^2$$

for any vectors a and b. Therefore

$$\|a\|^2 \geq \frac{(a \cdot b)^2}{\|b\|^2}$$

for any nonzero vector b. Identifying a as $w(k)$ and b as $w*$ yields

$$\|w(k)\|^2 \geq \frac{(w(k)\cdot w*)^2}{\|w*\|^2} > \frac{(w(0)\cdot w* + \alpha km)^2}{\|w*\|^2}$$

Thus the square of the length of the weight vector at the k-th step, $w(k)$, grows faster than $(\alpha km)^2$ (quadratic in k) for large k.

The right-hand side of the inequality provides a lower bound for $\|w(k)\|^2$. Note that $w*$ clearly cannot be a zero vector.

What we want next is to find an upper bound for $\|w(k)\|^2$. Since $w(k)$ is given by

$$w(k) = w(k-1) + \alpha x(k-1)$$

taking the dot-product with itself gives

$$\|w(k)\|^2 = \|w(k-1)\|^2 + \alpha^2 \|x(k-1)\|^2 + 2\alpha x(k-1)\cdot w(k-1)$$

Since $x(k-1)\cdot w(k-1)$ must be negative because $x(k-1)$ is classified incorrectly, the last term on the right-hand side of the above equation can be omitted to obtain the inequality

$$\|w(k)\|^2 < \|w(k-1)\|^2 + \alpha^2 \|x(k-1)\|^2.$$

Replacing k by $k-1$ gives

$$\|w(k-1)\|^2 < \|w(k-2)\|^2 + \alpha^2 \|x(k-2)\|^2$$

Using this above inequality to eliminate $\|w(k-1)\|^2$ yields

$$\|w(k)\|^2 < \|w(k-2)\|^2 + \alpha^2 \|x(k-2)\|^2 + \alpha^2 \|x(k-1)\|^2$$

This procedue can be iterated to give

$$\|w(k)\|^2 < \|w(0)\|^2 + \alpha^2 \|x(0)\|^2 + \ldots + \alpha^2 \|x(k-2)\|^2 + \alpha^2 \|x(k-1)\|^2$$

Using the definition of M, we have the inequality

$$\|w(k)\|^2 < \|w(0)\|^2 + \alpha^2 kM$$

This inequality gives an upper bound for the square of $w(k)$ and shows that it cannot grow faster than $\alpha^2 kM$ (linear in k).

Combining this upper bound with the lower bound for $\|w(k)\|^2$, we have

$$\frac{(w(0)\cdot w* + \alpha km)^2}{\|w*\|^2} \leq \|w(k)\|^2 \leq \|w(0)\|^2 + \alpha^2 kM$$

This bound is clearly valid initially when $k = 0$, becaue in that case the bound is

$$(w(0)\cdot w*)^2 \leq \|w(0)\|^2 \leq \|w(0)\|^2$$

Notice that the square of the projection of $w(0)$ in any direction cannot be larger than $\|w(0)\|^2$.

Next consider what happens as k increases. Since the lower bound grows faster (quadratically with k) than the upper bound, which grows only linearly with k, there must be a value of k^* such that this condition is violated for all $k \geq k^*$. This means the iteration cannot continue forever and must therefore terminate after k^* steps.

To determine k^*, let κ be a solution of the following quadratic equation

$$\frac{\left(w(0)\cdot w^* + \alpha\kappa m\right)^2}{\|w^*\|^2} = \|w(0)\|^2 + \alpha^2\kappa M$$

We find that κ is given by

$$\kappa = \frac{R}{2} - D \pm \sqrt{\frac{R^2}{4} - RD + L^2}$$

where we have defined

$$R = \frac{\max_{x(n)\in F}\{x(n)\cdot x(n)\}}{\left(\min_{x(n)\in F}\{x(n)\cdot \hat{w}^*\}\right)^2}$$

$$D = \frac{w(0)\cdot w^*}{\alpha \min_{x(n)\in F}\{x(n)\cdot \hat{w}^*\}}$$

$$L^2 = \frac{\|w(0)\|^2}{\alpha^2\left(\min_{x(n)\in F}\{x(n)\cdot \hat{w}^*\}\right)^2}$$

In these expressions, $\hat{w}^*$ is a unit vector (having unit magnitude) pointing in the direction of w^*.

We want to show that only the upper sign leads to acceptable solution. First we assume that $\frac{R}{2} - D$ is positive. Clearly the upper sign will give a positive κ as required. The lower sign will lead to an unacceptable negative κ. We can prove that by showing that

$$\frac{R}{2} - D < \sqrt{\frac{R^2}{4} - RD + L^2}$$

Since both sides of this inequality are positive, we can square both sides to give

$$\left(\frac{R}{2} - D\right)^2 = \frac{R^2}{4} - RD + D^2 < \frac{R^2}{4} - RD + L^2$$

This relation becomes $D^2 < L^2$ or equivalently $\left(w(0)\cdot \hat{w}^*\right)^2 < \|w(0)\|^2$. This final relation is obviously true because the square of the projection of any vector in any direction cannot be larger than the magnitude square of the vector itself.

Next we assume that $\frac{R}{2} - D$ is negative. Then the lower sign will always give a negative κ. On the other hand, the upper sign will always give an acceptable positive κ because

$$\sqrt{\frac{R^2}{4} - RD + L^2} > -\left(\frac{R}{2} - D\right)$$

as can be seen by squaring both sides.

Our conclusion is that $k*$ is given by $\lceil \kappa \rceil$ where

$$\kappa = \frac{R}{2} - D + \sqrt{\frac{R^2}{4} - RD + L^2}$$

If the weight vector is initialized to zero, then $D = L = 0$ and therefore, $k* = \lceil \kappa \rceil = \lceil R \rceil$. From the definition of R, we see that the number of iteration is determined by the training vector that lie closest to the decision boundary. the smaller that distance is the higher the number of iterations has to be for convergence.

5. How we can use perceptron for multi category classification?

[MODEL QUESTION]

Answer:

We will now extend to the case where a NN has to be able to classify input vectors, each having N components, into more than 2 categories.

We will continue to use only bipolar neurons whose transfer functions are given by

$$f_\theta(x) = \begin{cases} +1, & \text{if } x > \theta, \\ 0, & \text{if } -\theta \le x \le \theta, \\ -1, & \text{if } x < -\theta. \end{cases}$$

Clearly the NN now needs to have multiple neurons, $Y_1, Y_2, \ldots, Y_M$, in the output layer. The number of output neurons is secified by M. Each input neuron is connected to each of these output neuron with a weight factor. The weight associated with the connection between input neuron X_i with output neuron Y_j is denoted by w_{ij}. The figure shows a diagram of this NN.

What was the weight vector now becomes the weight matrix having two separate indices: i goes from 1 to N and j goes from 1 to M. The weight matrix, W, is in general a rectangular one, unjess it happens that $N = M$. We follow common mathematical convention to use boldface capital letters to denote matrices. The total number of these weight components is given by NM.

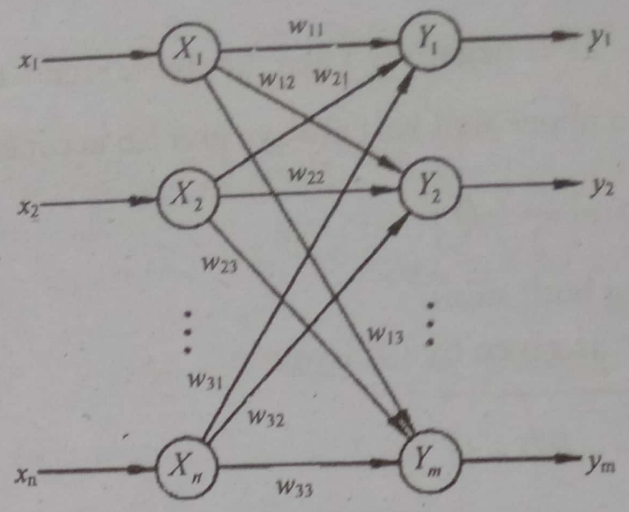


Fig: 1 A single-layer feed-forward *NN* having multiple output neurons

Each output neuron also has its own bias, denoted by b_j, where $j = 1, \ldots, M$. Thus the total input into neuron Y_j is $\quad y_{\text{in},j} = b_j + \sum_{i=1}^{N} x_i w_{ij}$

As before we can absorb a bias by introdicing an extra input neuron, X_0, so that the weight connecting it to Y_j is $w_{0j} = b_j$, provided that the signal in X_0 is always fixed to be $x_0 = 1$. Then the total input into neuron Y_j can be rewritten as $\quad y_{\text{in},j} = \sum_{i=0}^{N} x_i w_{ij}$

The output of neuron Y_j is then given by $\quad y_j = f_\theta\left(y_{\text{in},j}\right)$

We need to be able to train this NN using a more general form of the Perceptron learning rule. To train the NN, we assume that we are given a training set:

$$\left\{s^{(q)}, t^{(q)}\right\}, \qquad q = 1, 2, \ldots, Q$$

where, $s^{(q)}$ is a training vector, each having N components and $t^{(q)}$ is an M-component vector, representing the corresponding targeted output values. Notice that the target is no longer a scalar quantity. Since each component of the output can take on two different values, a NN with M output neuron is capable of classifying input vectors into 2^M categories (under practical situation where $N \gg M$).

The general Perceptron rule is:

1. Set $\alpha(>0)$ and θ. Initialize W and b.
2. Repeat steps, while cycling through training set, until all training patterns are classified correctly

(a) Set input vector $x_i = s_i^{(q)}$, $i = 1, \ldots, N$.

(b) Compute total input for each output neuron:

$$y_{\text{in},j} = \sum_{i=1}^{N} x_i w_{ij} + b_j, \qquad j = 1, \ldots, M$$

(c) Compute the output

$$y_j = f_\theta\left(y_{\text{in},j}\right), \quad j = 1, \ldots, M$$

(d) Update W and b only if pattern is misclassified

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \alpha x_i t_j^{(q)}$$

$$b_j^{\text{new}} = b_j^{\text{old}} + \alpha t_j^{(q)}$$

For $\theta = 0$, there is a total of M decision hyperplanes determined by

$$\sum_{i=1}^{N} x_i w_{ij} + b_j = \sum_{i=0}^{N} x_i w_{ij} = 0, \qquad j = 1, \ldots, M$$

and for $N \geq M$ these hyperpanes divide the N dimensional input space into 2^M distinct regions. Thus each column of the weight matrix, W, gives the weight vector that determines the normal direction of each of the M hyperplanes. if we do not absorb the biases by introducing a zeroth neuron, then we can define weight vectors $w_j = \left[w_{1j}, w_{2j}, \ldots, w_{Nj}\right]$ so that the equation for the j-th hyperplane can be written as

$$x \cdot \hat{w}_j = \frac{-b_j}{\|w_j\|}$$

However, if we absorb the biases by introducing a zeroth neuron, then we can define weight vectors $w_j = \left[w_{0j} w_{1j} w_{2j} \ldots w_{Nj}\right]$ so that the equation for the j-th hyperplane can be written as

$$x \cdot \hat{w}_j = 0$$

For finite value of θ, these hyperplanes become slabs of width $2d$, where the value of d will be determined below. Each slab divides the input vector space into three regions:

$$\sum_{i=1}^{N} x_i w_{ij} + b_j > \theta \qquad \Rightarrow \qquad y = +1,$$

$$-\theta \leq \sum_{i=1}^{N} x_i w_{ij} + b_j \leq \theta \qquad \Rightarrow \qquad y = 0,$$

$$\sum_{i=1}^{N} x_i w_{ij} + b_j < -\theta \qquad \Rightarrow \qquad y = -1$$

The slab is bounded on each side by hyperplanes determined by

$$x \cdot \hat{w}_j = \frac{-b_j + \theta}{\|w_j\|},$$

$$x \cdot \hat{w}_j = \frac{-b_j - \theta}{\|w_j\|}$$

where the weight vectors ae defined by $w_j = \left[w_{1j} w_{2j} \ldots w_{Nj}\right]$. Therefore the slab has a thichness $2d_j$, where

$$d_j = \frac{\theta}{\|w_j\|}$$

However, if we absorb the biases by introducing a zeroth neuron, then the slab is bounded on each side by hyperplanes determined by

$$x \cdot \hat{w}_j = \frac{\theta}{\|w_j\|}, \quad x \cdot \hat{w}_j = \frac{-\theta}{\|w_j\|},$$

where the weight vectors are now defined by $w_j = \left[w_{0j} w_{1j} w_{2j} \ldots w_{Nj} \right]$. Therefore, the slab has a thickness $2d_j$, where

$$d_j = \frac{\theta}{\|w_j\|}$$

Although this expression for d_j looks identical to the one above, the weight vectors are defined slightly differently and the slab is located in an input vector space one higher than the one before. The figure shows one of the hyperphanes with a safety margin.

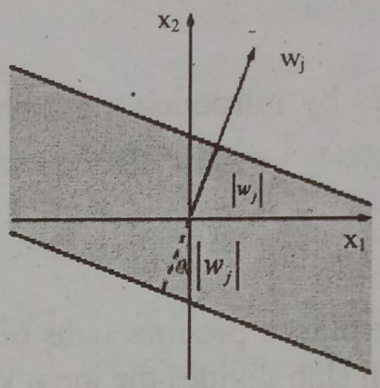


Fig: 2 A hyperplane with a safety margin

6. What is geometry of linear discriminant function? **[MODEL QUESTION]**

Answer:

The equation $g(x) = 0$ defines a hyperplane where on one side of the plane we have C_1 and on the other side of the plane we have C_2. The geometry is summarized below.

Theorem 1: (Geometry of a linear discriminant function). Let g be a linear discriminant function and let $H = \{x \mid g(x) = 0\}$ be the separating hyperplane. Then,

(i) $\frac{w}{\|w\|_2}$ is the normal vector of H, i.e., for any $x_1, x_2 \in H$, we have

$$w^T (x_1 - x_2) = 0$$

(ii) For any x_0, the distance between x_0 and H is

$$d(x_0, H) = \frac{g(x_0)}{\|w\|_2}$$

Proof: To prove the first statement, we note consider two points $x_1 \in H$ and $x_2 \in H$. Since, both points are on the hyperplane, we have $g(x_1)=0$ and $g(x_2)=0$ and hence $w^T x_1 + w_0 = w^T x_2 + w_0 = 0$. Rearranging the equations yields $w^T(x_1 - x_2) = 0$. Therefore, w is a normal vector to the surface H and $\frac{w}{\|w\|_2}$ is a scaled version of w so that the norm of $\frac{w}{\|w\|_2}$ is 1.

To prove the second statement, we can decompose x_0 as $x_0 = x_p + \eta \frac{w}{\|w\|_2}$, where x_p is the orthogonal projection of x_0 onto H such that $g(x_p)=0$. In other words, we decompose x_0 into its two orthogonal components: one along the normal vector w and the other along the tangent. Since

$$g(x_0) = w^T x_0 + w_0 = w^T\left(x_p + \eta \frac{w}{\|w\|_2}\right) + w_0$$

$$= g(x_p) + \eta \|w\|_2 = \eta \|w\|_2$$

We have that $\eta = \frac{g(x_0)}{\|w\|_2}$.

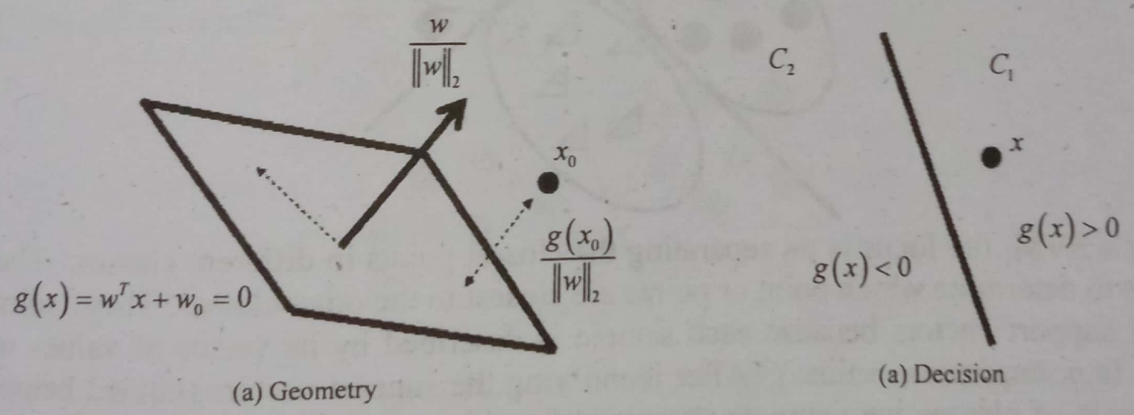


Fig: 1 The geometry of linear discriminant function

The geometry of the above theorem is illustrated in Fig. 1. The theorem tells us two things about the hyperplane: (i) **Orientation of *H*:** The orientation of H is specified by $\frac{w}{\|w\|_2}$, as it is the normal vector of H. (ii) **Distance to *H*:** The distance of a point x_0 to H is $\frac{g(x_0)}{\|w\|_2}$. Therefore, the bigger the value $g(x_0)$ is the farther away the point x_0 is

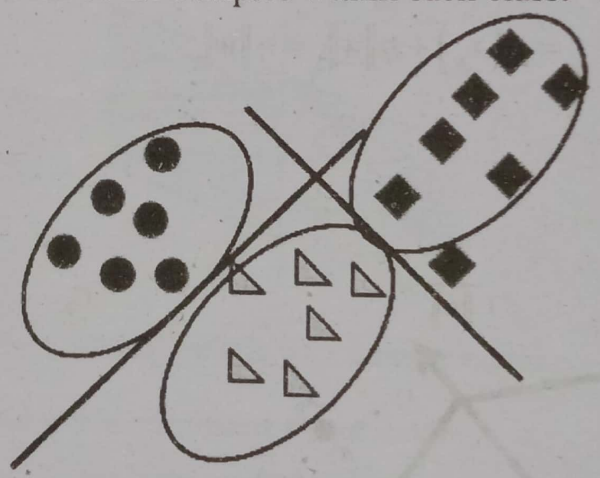
from the decision boundary and hence the more "confident" x_0 is belonging to one class but not the other. To classify a data point x, we check whether x belongs to C_1 or C_2 by checking if $g(x) > 0$.

7. Explain the idea of support vector machine. **[MODEL QUESTION]**

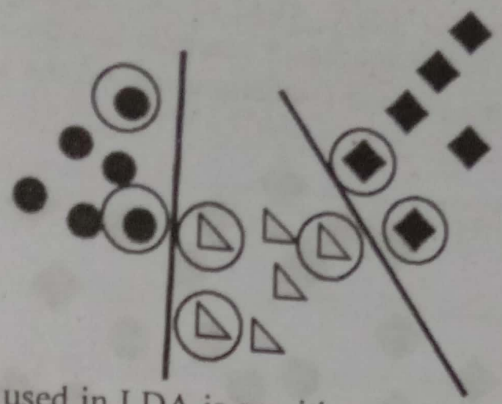
Answer:

The idea behind the basic support vector machine (SVM) is similar to LDA - find a hyperplane separating the classes. In LDA, the separating hyperplane separates the means of the samples in each class, which is suitable when the data are sitting inside an ellipsoid or other convex set. However, intuitively what we really want is a rule that separates the samples in the different classes which appear to be closest. When the classes are separable (i.e. can be separated by a hyperplane without misclassification), the support vector machine (SVM) finds the hyperplane that maximizes the minimum distance from the hyperplane to the nearest point in each set. The nearest points are called the support vectors.

In discriminant analysis, the assumption is that the samples cluster more tightly around the mean of the within class distribution. Using this method the focus is on separating the ellipsoids containing the bulk of the samples within each class.



Using a SVM, the focus is on separating the closest points in different classes. The first step is to determine which point or points are closest to the other classes. These points are called support vectors because each sample is described by its vector of values on the genes (e.g. expression values). After identifying the support vectors (circled below) for each pair of classes we compute the separating hyperplane by computing the shortest distance from each support vector to the line. We then select the line the maximizes the sum of distances. If it is impossible to completely separate the classes, then a negative penalty is applied for misclassification - the selected line then maximizes the sum of distances plus penalty, which balances the distances from the support vectors to the line and the misclassifications.



The separating hyperplane used in LDA is sensitive to all the data in each class because you are using the entire ellipse of data and its center. The separating hyperplane used in SVM is sensitive only to the data on the boundary which provides computational savings and makes it more to robust outliers.

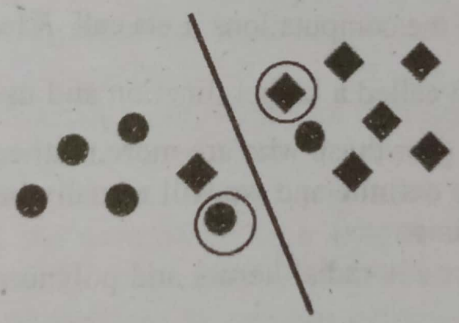
This is how it is applied:

Suppose the data vector for the i-th sample is x_i, which is the expression vector for several genes. Also, suppose we have only 2 classes denoted by $y_i = +/-1$. The hyperplane is defined by a vector W and a scalar b such that for all i.

$y_i(x_i W + b) - 1 \geq 0$ for all i with equality for the support vectors and $\|W\|$ is minimal.

After computing W and b using the training set, new samples are classified by determining the value of y satisfying the inequality.

If the clusters overlap, the "obvious" extension is to find the hyperplane that minimizes the distance between the main samples and the hyperplane, while penalizing for misclassified samples.

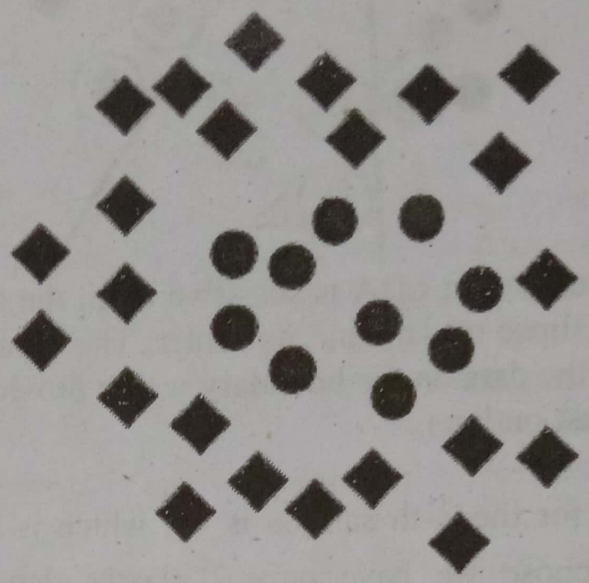


This is the solution to the optimization problem: $y_i(x_i W + b) - 1 + z_i \geq 0$ for all i with equality for the support vectors where all $z_i \geq 0$ and $\|W\| + C\Sigma_i z_i$ is minimal for some C.

The idea is that some points can be one the wrong side of the separating hyperplane, but there is a penalty for each misclassification. The hyperplane is now found by minimizing a criterion that still separates the support vectors as much as possible in this orthogonal direction, but also minimizes the number of misclassifications.

It is not clear that SVM does much better than linear discriminant analysis when the groups are actually elliptical. When the groups are not elliptical because of outliers SVM does better because the criterion is only sensitive to the boundaries and not sensitive to

the entire shape. However, like LDA, SVM requires groups that can be separated by planes. Neither method will do well for an example like the one below. However, there is a method called the kernel method (sometimes called the kernel trick) that works for more general examples.



The idea is simple - the implementation is very clever.
Suppose that we suspected that the clusters were in concentric circles (or spheres) as seen above. If we designate the very center of the points as the origin, we can compute the distance from each point to the center as a new variable. In this case, the clusters separate very nicely on the radius, creating a circular boundary.
The general idea is to add more variables that are nonlinear functions of the original variables until we have enough dimensions to separate the groups.
The kernel trick uses the fact that for both LDA and SVM, only the dot product of the data vectors $x_i'x_j$ are used in the computations. Lets call $K(x_i, x_j) = x_i'x_j$. We replace K with a more general function called a kernel function and use $K(x_i, x_j)$ as if it were the dot product in ur algorithm. (For those who are more mathematically inclined, the kernel function needs to be positive definite and we will actually be using the dot product in the basis described by its eigenfunctions).
The most popular kernels are the radial kernel and polynomial kernel. The radial kernel (sometimes called the Gaussian kernel) is $k(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)$.

NON METRIC METHOD FOR PATTERN CLASSIFICATION

Multiple Choice Type Questions

1. Which one of the following statements is TRUE for a Decision Tree? **[MODEL QUESTION]**

a) Decision tree is only suitable for the classification problem statement.
b) In a decision tree, the entropy of a node decreases as we go down a decision tree
c) in a decision tree, entropy determines purity
d) Decision tree can only be used for only numeric valued and continuous attributes

Answer: (b)

2. How do you choose the right node while constructing a decision tree? **[MODEL QUESTION]**

a) An attribute having high entropy
b) An attribute having high entropy and information gain
c) An attribute having the lowest information gain
d) An attribute having the highest information gain

Answer: (d)

Short Answer Type Questions

1. What do you understand by Nominal data? **[MODEL QUESTION]**

Answer:

We have considered pattern recognition based on feature vectors of real-valued and discrete-valued numbers and in all cases there has been a natural measure of distance between vectors. For instance in the nearest-neighbour classifier the notion figures conspicuously – indeed it is the core of the technique – while for neural networks the notion of similarity appears when two input vectors sufficiently "close" lead to similar outputs. Most practical pattern recognition methods address problems of this sort, where feature vectors are real-valued and there exists some notion of metric.

But suppose a classification problem involves nominal data – for instance descriptions that are discrete and without any natural notion of similarity or even ordering. Consider the use of information about teeth in the classification of fish and sea mammals. Some teeth are small and fine (as in baleen whales) for straining tiny prey from the sea. Others (as in sharks) come in multiple rows. Some sea creatures, such as walruses, have tusks. Yet othes, such as squid, lack teeth altogether. There is no clear notion of similarity (or metric) for this information about teeth. For instance, the teeth of a baleen whale and the tusks of a walrus are no more "similar" than are the teeth of a shark to the teeth in a flounder.

Thus, our attention turns away from describing patterns by vectors of real numbers and toward using lists of attributes. A common aproach is to specify the values of a fixed number of properties by a property d-tuple. For example, consider describing a piece of fruit by four properties: colour, texture, taste and size. Then a particular piece of fruit might be described by the 4-tuple (red, shiny, sweet, small), which is a shorthand for colour = red, texture = shiny, taste = sweet and size = small. Another common approach is to describe the pattern by a variable length string of nominal attributes, such as a sequence of base pairs in a segment of DNA – for example, "AGCTTCAGATTCCA". Such lists or strings might themselves by the output of other comjponeent classifiers of the type we have seen elsewhere. For instance, we might train a neural network to recognize different component brush strokes used in Chinese and Japanese characters (roughly a dozen basic forms); a classifier would then accept as inputs a list of these nominal attributes and make the final, full character classification.

How can we best use such nominal data for classification? Most importantly, how can we efficiently learn categories using such nonmetric data? If there is structure in strings, how can it be represented? In considering such problems we move beyond the notion of continuous probability distributions and metrics toward discrete problems that are addressed by rule-base or syntactic pattern recognition methods.

2. What category of algorithms does CART belong to? **[MODEL QUESTION]**

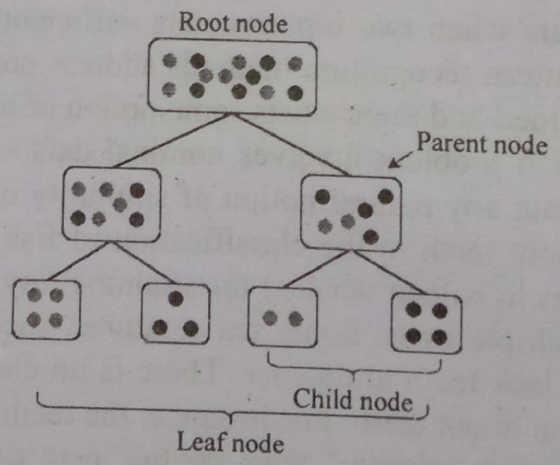
Answer:

As the name suggests, CART (Classification and Regression Trees) can be used for both classification and regression problems. The difference lies in the target variable:

- With **classification**, we attempt to predict a class label. In other words, classification is used for problems where the output (target variable) takes a finite set of values, e.g., whether it will rain tomorrow or not.
- Meanwhile, **regression** is used to predict a numerical label. This means your output can take an infinite set of values, e.g., a house price.

3. What are basic decision tree terminologies? **[MODEL QUESTION]**

Answer:



- **Parent and Child Node:** A node that gets divided into sub-nodes is known as Parent Node, and these sub-nodes are known as Child Nodes. Since a node can be divided

into multiple sub-nodes, therefore a node can act as a parent node of numerous child nodes

- **Root Node:** The top-most node of a decision tree. It does not have any parent node. It represents the entire population or sample
- **Leaf / Terminal Nodes:** Nodes that do not have any child node are known as Terminal/Leaf Nodes

4. What is Node Splitting in a Decision Tree & Why is it done? [MODEL QUESTION]

Answer:

Before learning any topic, I believe it is essential to understand why you're learning it. That helps in understanding the goal of learning a concept. So let's understand why to learn about node splitting in decision trees.

Since you all know how extensively decision trees are used, there is no denying the fact that learning about decision trees is a must. A decision tree makes decisions by splitting nodes into sub-nodes. This process is performed multiple times during the training process until only homogenous nodes are left. And it is the only reason why a decision tree can perform so well. Therefore, node splitting is a key concept that everyone should know.

Node splitting, or simply splitting, is the process of dividing a node into multiple sub-nodes to create relatively pure nodes. There are multiple ways of doing this, which can be broadly divided into two categories based on the type of target variable:

1. Continuous Target Variable
2. Reduction in Variance
3. Categorical Target Variable
   - Gini Impurity
   - Information Gain
   - Chi-Square

In the upcoming sections, we'll look at each splitting method in detail. Let's start with the first method of splitting – reduction in variance.

Long Answer Type Questions

1. Explain decision tree. [MODEL QUESTION]

Answer:

It is natural and intuitive to classify a pattern through a sequence of questions, in which the next question asked depends on the answer to the current question. This "20-questions" approach is particularly useful for non-metric data, since all of the questions can be asked in a "yes/no" or "true/false" or "value(property) ∈ set_of_values" style that does not require any notion of metric.

Such a sequence of questions is displayed in a directed **decision tree** or **simply tree**, where by convention the first or **root node** is displayed at the top, connected by successive (directional) **links** or **branches** to other nodes. These are similarly connected until we reach terminal or **leaf** nodes, which have no further links (Fig. 1). The

classification of a particular pattern begins at the root node, which asks for the value of a particular property of the pattern. The different links from the root node correspond to the different possible values. Based on the answer we follow the apropriate link to a subsequent or **descendent** node. In the trees we shall discuss, the links must be mutually distinct and exhaustive, i.e., one and only one link will be followed. The next step is to make the decision at the appropriate subsequent node, which can be considered the root of a **sub-tree**. We continue this way until we reach a leaf node, which has no further question. Each leaf node bears a category label and the test pattern is assigned the category of the leaf node reached.

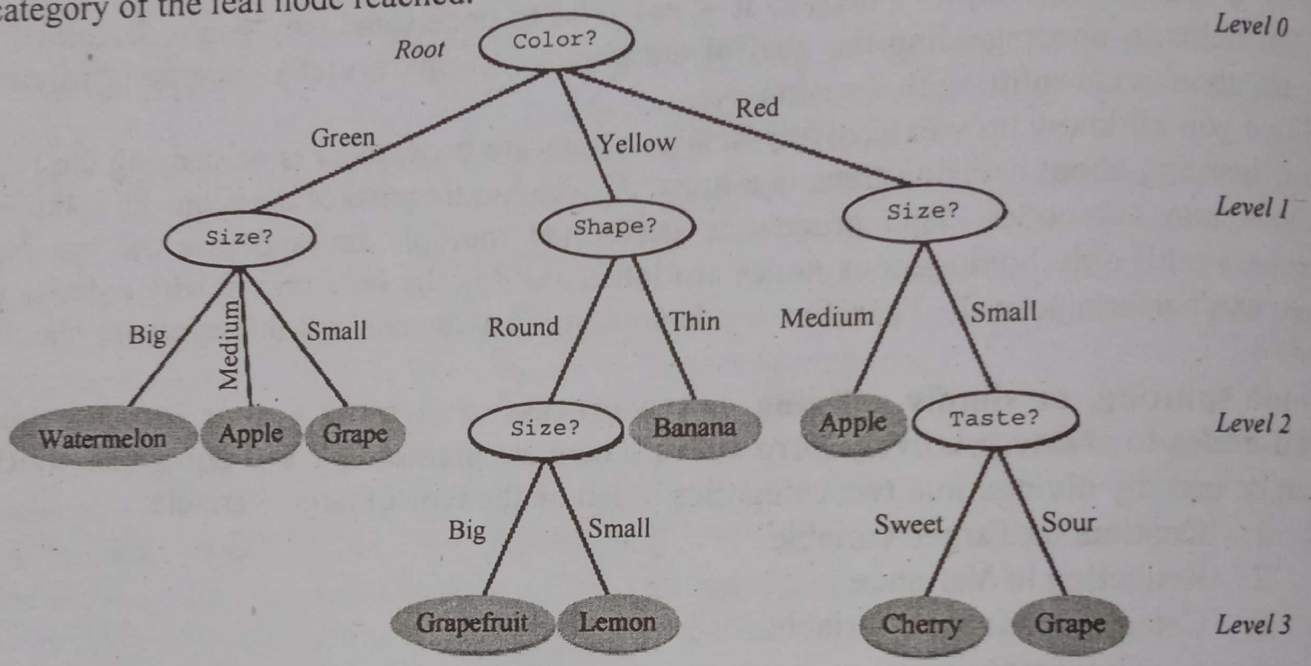


Fig: 1 Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category label is read. Note that the same question, **Size?**, appears in different places in the tree and that different questions can have different numbers of branches. Moreover, different leaf nodes, shown in pink, can be labeled by the same category (e.g., **Apple**)

The simple decision tree in Fig. 1 illustrates one benefit of trees over many other classifiers such as neutrla networks: interpretability. It is a straightforward matter to render the information in such a tree as logical expressions. Such interpretability has two manifestations. First, we can easily interpret the decision for any particular test pattern as the conjunction of decisions along the path to its corresponding leaf node. Thus, if the properties are `{taste, color, shape, size}`, the pattern `x = {sweet, yellow, thin, medium}` is classified as **Banana** because it is `(color = yellow)` AND `(shape = thin)`. Second, we can occasionally get clear interpretations of the **categories** themselves, by creating logical descriptions using conjunctions and disjunctions. For instance the tree shows `Apple = (green AND medium) OR (red AND medium)`.

Rules derived from trees – especially large trees – are often quite complicated and must be reduced to aid interpretation. For our example, one simple rule describes `Apple = (medium AND NOT yellow)`. Another benefit of trees is that they lead to rapid classification, emloying a sequence of typically simple queries. Finally, we note that trees provide a natural way to incorporate prior knowledge from human experts. In practie, though, such expert knowledge if of greatest use when the classification problem is fairly simple and the training set is small.

2. How do classification and regression trees work? **[MODEL QUESTION]**

Answer:

Let's start with a simple example. Assume you have a bunch of oranges and mandrins with labels on them, and you want to identify a set of simple rules that you can use in the future to distinguish between these two types of fruit.

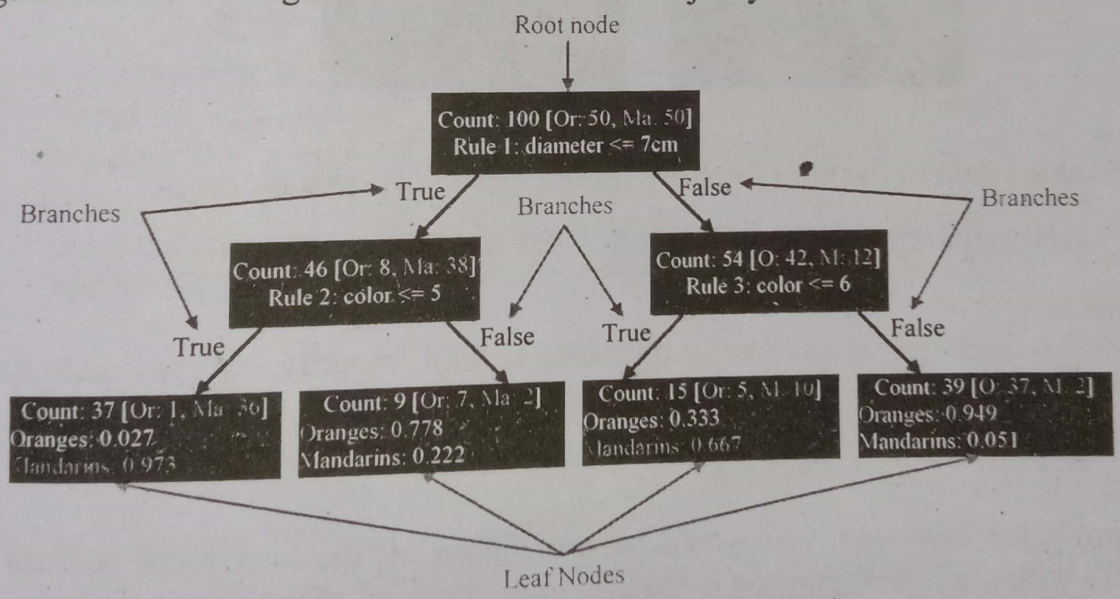
Typically, oranges (diameter 6–10cm) are bigger than mandarins (diameter 4–8cm), so the first rule found by your algorithm might be based on size:

- Diameter ≤ 7cm.

Next, you may notice that mandarins tend to be slightly darker in color than oranges. So, you use a color scale (1=dark to 10=light) to split your tree further:

- Color ≤5 for the left side of the sub-tree
- Color ≤6 for the right side of the sub-tree

Your final result is a tree that consists of 3 simple rules that help you to correctly distinguish between oranges and mandarins in the majority of the cases:



Decision tree for identifying oranges vs. mandarins

3. How does CART find the best split? **[MODEL QUESTION]**

Answer:

Several methods can be used in CART to identify the best splits. Here are two of the most common ones for classification trees:

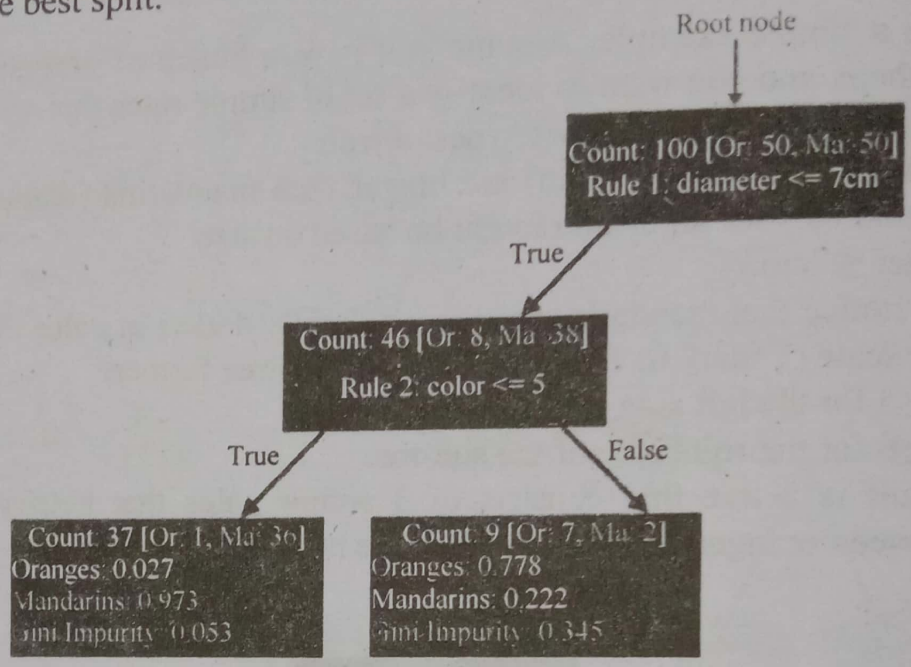
$$\text{Gini Impurity} = 1 - \text{Gini} = 1 - \sum_{i=1}^{n} p_i^2$$

where, p_i is the fraction of items in the class i.

Using the above tree as an example, Gini Impurity for the leftmost leaf node would be:

$$1 - \left(0.027^2 + 0.973^2\right) = 0.053$$

To find the best split, we need to calculate the weighted sum of Gini Impurity for both child nodes. We do this for all possible splits and then take the one with the **lowest Gini Impurity** as the best split.



The weighted sum of the Gini Impurity for the two child nodes is:

$$\text{Gini Impurity} = \frac{37}{46} * 0.053 + \frac{9}{46} * 0.345 = 0.110$$

Entropy:

The entropy approach is essentially the same as Gini Impurity, except it uses a slightly different formula:

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log_2\left(p_i\right)$$

To identify the best split, you would have to follow all the same steps outlined above. The split with the lowest entropy is the best one. Similarly, if the entropy of the two child nodes is not lower than that of a parent bode, you should not split any further.

4. Explain the methods for decision tree splitting. **[MODEL QUESTION]**

Answer:

1: Reduction in Variance

Reduction in Variance is a method for splitting the node used when the target variable is continuous, i.e., regression problems. It is so-called because it uses variance as a measure for deciding the feature on which node is split into child nodes.

$$\text{Variance} = \frac{\sum (X-\mu)^2}{N}$$

Variance is used for calculating the homogeneity of a node. If a node is entirely homogeneous, then the variance is zero.

Here are the steps to split a decision tree using reduction in variance:

1. For each split, individually calculate the variance of each child node
2. Calculate the variance of each split as the weighted average variance of child nodes
3. Select the split with the lowest variance
4. Perform steps 1-3 until completely homogeneous nodes are achieved

2: Information Gain

Well, the answer to that is Information Gain. Information Gain is used for splitting the nodes when the target variable is categorical. It works on the concept of the entropy and is given by:

$$\text{Information Gain} = 1 - \text{Entropy}$$

Entropy is used for calculating the purity of a node. **Lower the value of entropy, higher is the purity of the node.** The entropy of a homogeneous node is zero. Since we subtract entropy from 1, the Information Gain is higher for the purer nodes with a maximum value of 1. Now, let's take a look at the formula for calculating the entropy:

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Steps to split a decision tree using Information Gain:

1. For each split, individually calculate the entropy of each child node
2. Calculate the entropy of each split as the weighted average entropy of child nodes
3. Select the split with the lowest entropy or highest information gain
4. Until you achieve homogeneous nodes, repeat steps 1-3

3: Gini Impurity

Gini Impurity is a method for splitting the nodes when the target variable is categorical. It is the most popular and the easiest way to split a decision tree. The Gini Impurity value is:

$$\text{Gini Impurity} = 1 - \text{Gini}$$

Gini is the probability of correctly labeling a randomly chosen element if it was randomly labeled according to the distribution of labels in the node. The formula for Gini is:

$$\text{Gini} = \sum_{i=1}^{n} p_i^2$$

And Gini Impurity is:

Lower the Gini Impurity, higher is the homogeneity of the node. **The Gini Impurity of a pure node is zero.** Now, you might be thinking we already know about Information Gain then, why do we need Gini Impurity?

Gini Impurity is preferred to Information Gain because it does not contain logarithms which are computationally intensive.

Here are the steps to split a decision tree using Gini Impurity:

1. Similar to what we did in information gain. For each split, individually calculate the Gini Impurity of each child node
2. Calculate the Gini Impurity of each split as the weighted average Gini Impurity of child nodes
3. Select the split with the lowest value of Gini Impurity
4. Until you achieve homogeneous nodes, repeat steps 1-3

4: Chi-Square

Chi-square is another method of splitting nodes in a decision tree for datasets having categorical target values. It can make two or more than two splits. It works on the statistical significance of differences between the parent node and child nodes.

Chi-Square value is:

$$\text{Chi-Square} = \sqrt{\frac{(\text{Actual} - \text{Expected})^2}{\text{Expected}}}$$

Here, the *Expected* is the expected value for a class in a child node based on the distribution of classes in the parent node, and **Actual** is the actual value for a class in a child node.

The above formula gives us the value of Chi-Square for a class. Take the sum of Chi-Square values for all the classes in a node to calculate the Chi-Square for that node. Higher the value, higher will be the differences between parent and child nodes, i.e., higher will be the homogeneity.

Here are the steps to split a decision tree using Chi-Square:

1. For each split, individually calculate the Chi-Square value of each child node by taking the sum of Chi-Square values for each class in a node
2. Calculate the Chi-Square value of each split as the sum of Chi-Square values for all the child nodes
3. Select the split with higher Chi-Square value
4. Until you achieve homogeneous nodes, repeat steps 1-3

UNSUPERVISED LEARNING AND CLUSTERING

Multiple Choice Type Questions

1. Which of the following refers to the problem of finding abstracted patterns (or structures) in the unlabeled data? [MODEL QUESTION]
   a) Supervised learning
   b) Unsupervised learning
   c) Hybrid learning
   d) Reinforcement learning

Answer: (b)

2. What is the minimum no. of variables / features required to perform clustering? [MODEL QUESTION]
   a) 0
   b) 1
   c) 2
   d) 3

Answer: (b)

3. Is it possible that Assignment of observations to clusters does not change between successive iterations in K-Means [MODEL QUESTION]
   a) Yes
   b) No
   c) Can't say
   d) None of these

Answer: (a)

4. Which of the following algorithm is most sensitive to outliers? [MODEL QUESTION]
   a) K-means clustering algorithm
   b) K-medians clustering algorithm
   c) K-modes clustering algorithm
   d) K-medoids clustering algorithm

Answer: (a)

Short Answer Type Questions

1. What is unsupervised learning? [MODEL QUESTION]

Answer:

In some pattern recognition problems, the training data consists of a set of input vectors x without any corresponding target values. The goal in such unsupervised learning problems may be to discover groups of similar examples within the data, where it is called **clustering**, or to determine how the data is distributed in the space, known as **density estimation**. To put forward in simpler terms, for a n-sampled space x1 to xn, true class labels are not provided for each sample, hence known as **learning without teacher**.

2. What are the issues with unsupervised learning? [MODEL QUESTION]

Answer:

- Unsupervised Learning is harder as compared to Supervised Learning tasks.
- How do we know if results are meaningful since no answer labels are available?
- Let the expert look at the results (external evaluation)

- Define an objective function on clustering (internal evaluation)

3. Why Unsupervised Learning is needed despite of these issues? **[MODEL QUESTION]**

Answer:

- Annotating large datasets is very costly and hence we can label only a few examples manually. Example: Speech Recognition
- There may be cases where we don't know how many/what classes is the data divided into. Example: Data Mining
- We may want to use clustering to gain some insight into the structure of the data before designing a classifier.

4. What are problems associated with clustering? **[MODEL QUESTION]**

Answer:

There are a number of problems with clustering. Among them:

- dealing with large number of dimensions and large number of data items can be problematic because of time complexity;
- the effectiveness of the method depends on the definition of "distance" (for distance-based clustering). If an *obvious* distance measure doesn't exist we must "define" it, which is not always easy, especially in multidimensional spaces;
- the result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.

5. What are possible applications of clustering algorithm? **[MODEL QUESTION]**

Answer:

Clustering algorithms can be applied in many fields, for instance:

- **Marketing:** finding groups of customers with similar behaviour given a large database of customer data containing their properties and past buying records;
- **Biology:** classification of plants and animals given their features;
- **Insurance:** identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds;
- **Earthquake studies:** clustering observed earthquake epicenters to identify dangerous zones;
- **World Wide Web:** document classification; clustering weblog data to discover groups of similar access patterns.

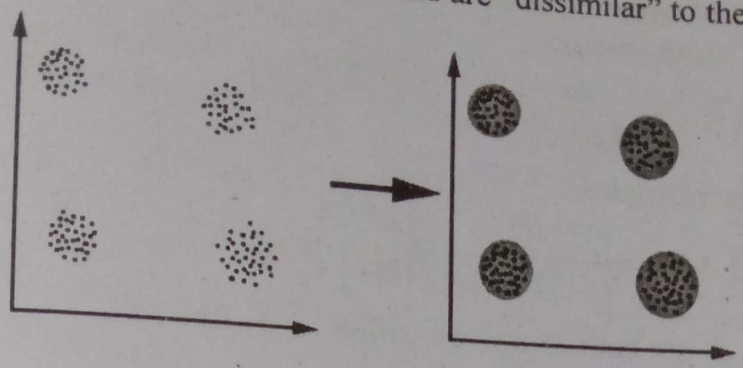
Long Answer Type Questions

1. What is Clustering? **[MODEL QUESTION]**

Answer:

Clustering can be considered the most important **unsupervised learning** problem; so, as every other problem of this kind, it deals with finding a **structure** in a collection of unlabeled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A **cluster** is therefore a collection

of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.



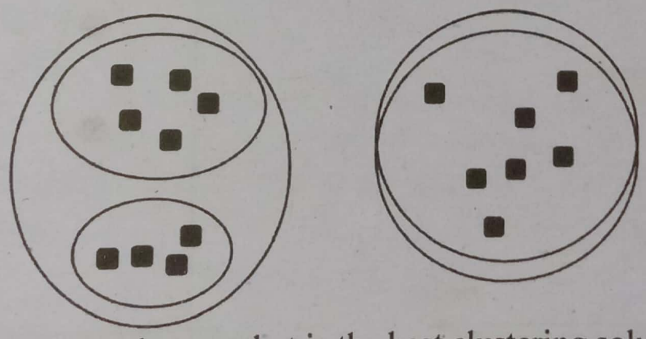
Distance-based clustering:

Given a set of points, with a notion of distance between points, grouping the points into some number of **clusters**, such that

- internal (within the cluster) distances should be small i.e members of clusters are close/similar to each other.
- external (intra-cluster) distances should be large i.e. members of different clusters are dissimilar.

The Goals of Clustering:

The goal of clustering is to determine the internal grouping in a set of unlabeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the clustering. Consequently, it is the user who should supply this criterion, in such a way that the result of the clustering will suit their needs.



In the above image, how do we know what is the best clustering solution?

To find a particular clustering solution , we need to define the similarity measures for the clusters.

Proximity Measures:

For clustering, we need to define a proximity measure for two data points. Proximity here means how similar / dissimilar the samples are with respect to each other.

- Similarity measure $S(x_i, x_k)$: large if x_i, x_k are similar
- Dissimilarity (or distance) measure $D(x_i, x_k)$: small if x_i, x_k are similar

There are various similarity measures which can be used.

- Vectors: Cosine Distance

$$s(x, x') = \frac{x'x'}{\|x\|\|x'\|}$$

- Sets: Jaccard Distance

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

(If A and B are both empty, we define $J(A, B) = 1$)

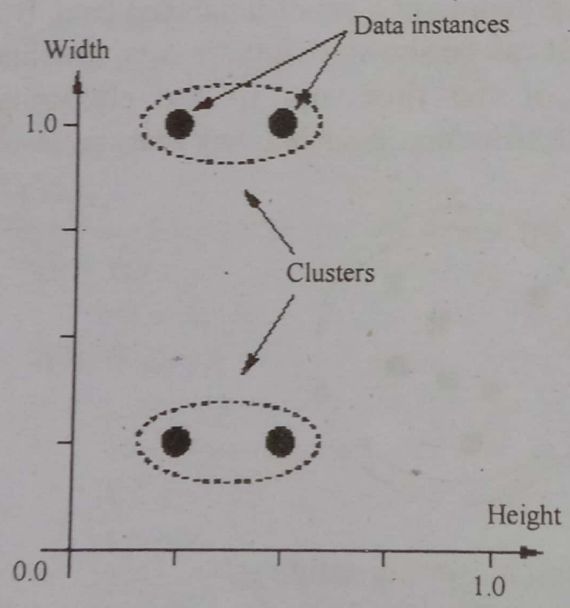
$$0 \le J(A, B) \le 1$$

- Points: Euclidean Distance

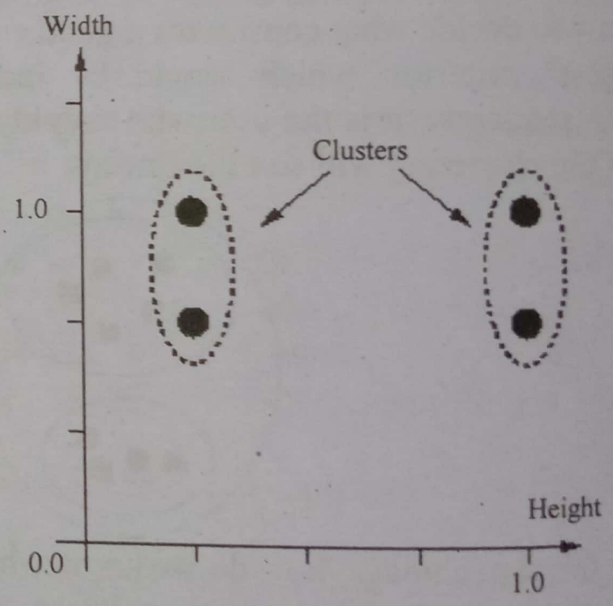
$$q = 2$$

$$d(x, x') = \left(\sum_{k=1}^{d} |x_k - x'_k|^q \right)^{1/q}$$

A "good" proximity measure is VERY application dependent. The clusters should be invariant under the transformations "natural" to the problem. Also, while clustering it is not advised to normalize data that are drawn from multiple distributions.



Before scaling



After scaling

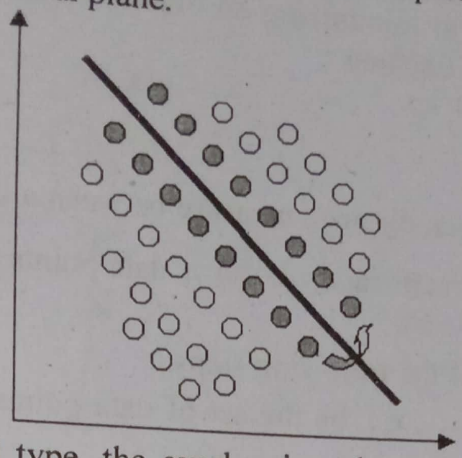
2. What are different clustering algorithms? **[MODEL QUESTION]**

Answer:

Clustering algorithms may be classified as listed below:

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

In the first case data are grouped in an exclusive way, so that if a certain data point belongs to a definite cluster then it could not be included in another cluster. A simple example of that is shown in the figure below, where the separation of points is achieved by a straight line on a bi-dimensional plane.



On the contrary, the second type, the overlapping clustering, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value.
A hierarchical clustering algorithm is based on the union between the two nearest clusters. The beginning condition is realized by setting every data point as a cluster. After a few iterations it reaches the final clusters wanted.
Finally, the last kind of clustering uses a completely probabilistic approach.
Below there are four of the most used clustering algorithms:

- K-means
- Fuzzy K-means
- Hierarchical clustering
- Mixture of Gaussians

Each of these algorithms belongs to one of the clustering types listed above. While, K-means is an **exclusive clustering** algorithm, Fuzzy K-means is an **overlapping clustering** algorithm, Hierarchical clustering is obvious and lastly Mixture of Gaussians is a **probabilistic clustering** algorithm. We will discuss about each clustering method in the following paragraphs.

3. Describe K-Means algorithm. **[MODEL QUESTION]**

Answer:

K-means is one of the simplest unsupervised learning algorithms that solves the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centres, one for each cluster. These centroids should be placed in a smart way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same

data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.
Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

$$J = \sum_{i=1}^{k} \sum_{i=1}^{n} \left\| X_i^{(j)} - c_j \right\|^2$$

where, $\left\| X_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point xi and the cluster centre cj, is an indicator of the distance of the n data points from their respective cluster centres.
The algorithm is composed of the following steps:

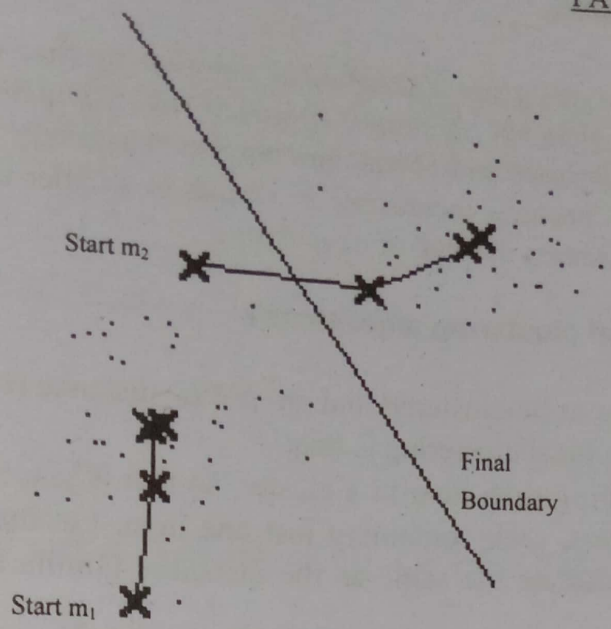
- Let, $X = \{x_1, x_2, x_3, \ldots, x_n\}$ be the set of data points and $V = \{v_1, v_2, \ldots, v_c\}$ be the set of centers.
- Randomly select 'c' cluster centers.
- Calculate the distance between each data point and cluster centers.
- Assign the data point to the cluster center whose distance from the cluster center is minimum of al the cluster centers.
- Recalculate the new cluster center using:

$$v_i = \left(\frac{1}{c_i}\right) \sum_{j=1}^{c_i} x_i$$

where, 'c_i' represents the number of data points in i-th cluster.

- Recalculate the distance between each data point and new obtained cluster centers.
- If no data point was reassigned then step, otherwise repeat from step 3).

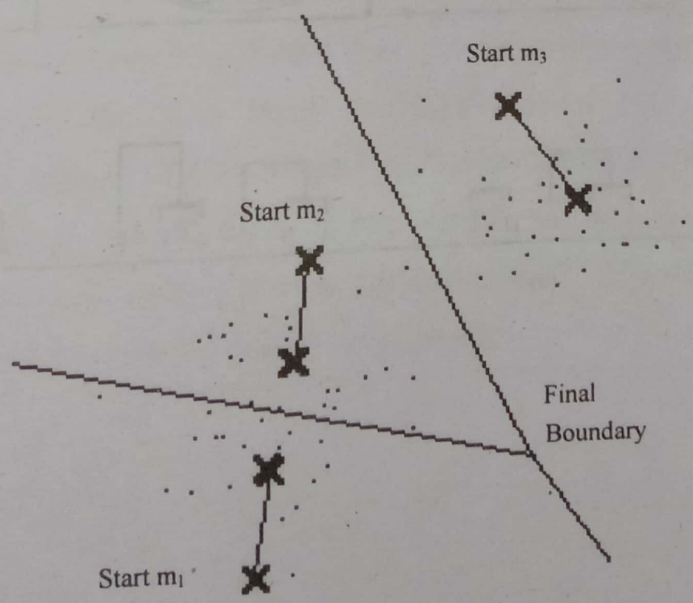
Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centres. The k-means algorithm can be run multiple times to reduce this effect.
k-means is a simple algorithm that has been adapted to many problem domains. As we are going to see, it is a good candidate for extension to work with fuzzy feature vectors.



The k-means procedure can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. It does have some weaknesses:

- The way to initialise the means was not specified. One popular way to start is to randomly choose k of the samples.
- It can happen that the set of samples closest to m_i is empty, so that m_i cannot be updated. This is a problem which needs to be handled during the implementation, but is generally ignored.
- The results depend on the value of k and there is no optimal way to describe a best "k".

This last problem is particularly troublesome, since we often have no way of knowing how many clusters exist. In the example shown above, the same algorithm applied to the same data produces the following 3-means clustering. Is it better or worse than the 2-means clustering?

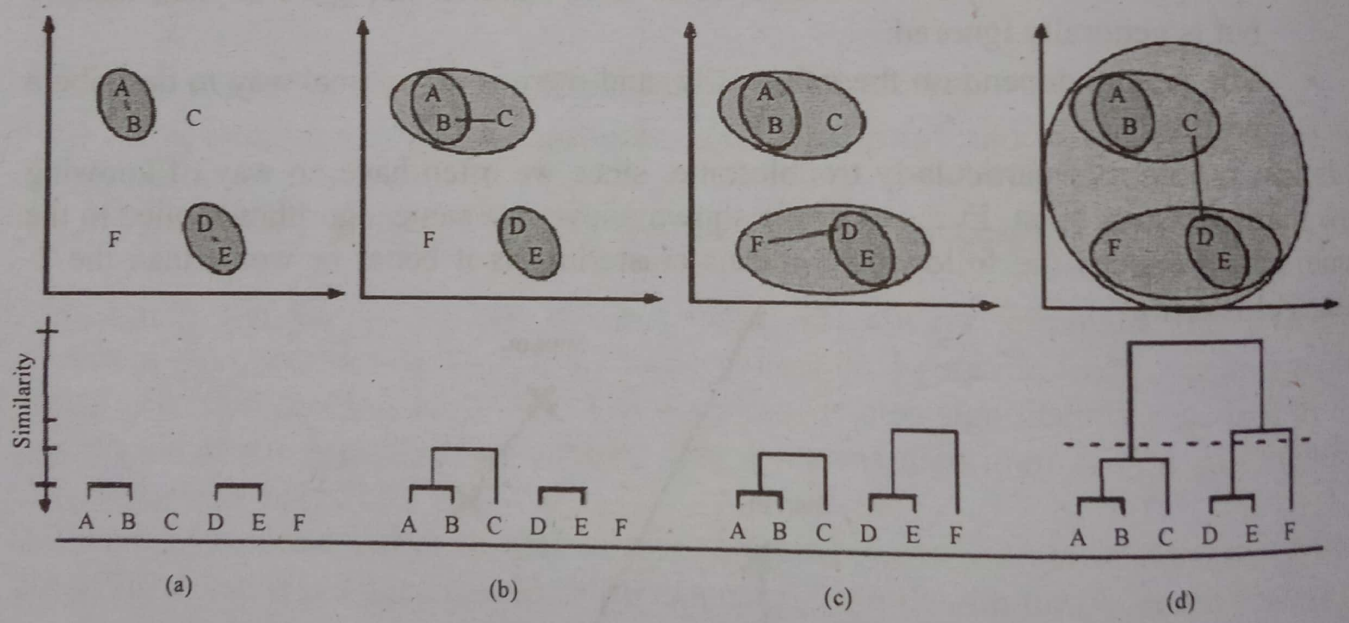


Unfortunately there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given criterion, but we need to be careful because increasing k results in smaller error function values by definition, but also increases the risk of over-fitting.

4. What is hierarchical clustering algorithms? **[MODEL QUESTION]**

Answer:

Given a set of N items to be clustered and an $N*N$ distance (or similarity) matrix, the basic process of hierarchical clustering is this:

- Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters the same as the distances (similarities) between the items they contain.
- Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.
- Compute distances (similarities) between the new cluster and each of the old clusters.
- Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Example: Hierarchical Agglomerative Clustering