1. **Bayesian Classifier (Refer Class notes)**

- **We defined prior, conditional and joint probability for random variables**
  - **Prior probability:** $P(X)$
  - **Conditional probability:** $P(X_1 \mid X_2), P(X_2 \mid X_1)$
  - **Joint probability:** $\mathbf{X} = (X_1, X_2),\ P(\mathbf{X}) = P(X_1, X_2)$
  - **Relationship:** $P(X_1, X_2) = P(X_2 \mid X_1)P(X_1) = P(X_1 \mid X_2)P(X_2)$
  - **Independence:** $P(X_2 \mid X_1) = P(X_2),\ P(X_1 \mid X_2) = P(X_1),\ P(X_1, X_2) = P(X_1)P(X_2)$

Bayesian Classifier Algorithm

Let us say there are, C1, C2,… , Ck classes

During Testing

Compute $P(C_i|X)$ for each class C_i (i=1:k) using the equation

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

During Training

Compute $P(X|C_i)$ and $P(C_i)$ for all the features and for all the classes

Bayesian Classifier Problems

Problem 1

1 Given:

- A doctor knows that meningitis causes stiff neck 50% of the time
- Prior probability of any patient having meningitis is 1/50,000
- Prior probability of any patient having stiff neck is 1/20

If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Problem 2

Consider the "Cancer test kit" problem, which has the following features:
Given that the subject has Cancer "C", the probability of the test kit producing a positive decision "+" is = 0.98.
Probability of the kit producing a negative decision "-" given that the subject is healthy "H" is=0.97
The prior probability of Cancer in the population=0.01.

We would like to know the probability that the subject has Cancer given that the test kit generated a positive decision?

$$P(C \mid +) = \frac{P(+ \mid C)P(C)}{P(+)} = \frac{P(+ \mid C)P(C)}{P(+ \mid C)P(C) + P(+ \mid H)P(H)} = 0.266$$

Problem 3

Given the following dataset,

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Given a new instance of variable values,

x'=(Outlook=*Sunny,* Temperature=*Cool,* Humidity=*High,* Wind=*Strong*)

PlayTennis is Yes or No?

P(*Yes*|**x**'): [P(*Sunny*|*Yes*)P(*Cool*|*Yes*)P(*High*|*Yes*)P(*Strong*|*Yes*)]P(Play=*Yes*) = ?
P(*No*|**x**'): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = ?

Answer

$P(\text{Play}=\textit{Yes}) = 9/14$

$P(\text{Play}=\textit{No}) = 5/14$

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

P(Outlook=*Sunny* | Play=**Yes**) = 2/9
P(Temperature=*Cool* | Play=*Yes*) = 3/9
P(Huminity=*High* | Play=*Yes*) = 3/9
P(Wind=*Strong* | Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=*Sunny* | Play=**No**) = 3/5
P(Temperature=*Cool* | Play==*No*) = 1/5
P(Huminity=*High* | Play=*No*) = 4/5
P(Wind=*Strong* | Play=*No*) = 3/5
P(Play=*No*) = 5/14

P(*Yes*|**x**'): [P(*Sunny*|*Yes*)P(*Cool*|*Yes*)P(*High*|*Yes*)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|**x**'): [P(*Sunny*|No) P(*Cool*|No)P(*High*|No)P(*Strong*|No)]P(Play=*No*) = 0.0206

Answer=No

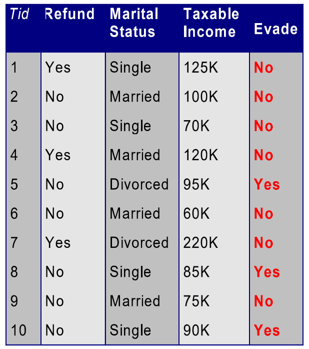
If input X is Continuous-valued Input Attributes

$$\hat{P}(X_j|C=c_i)=\frac{1}{\sqrt{2\pi}\sigma_{ji}}\exp\left(-\frac{(X_j-\mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C=c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C=c_i$

Given a dataset,



Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

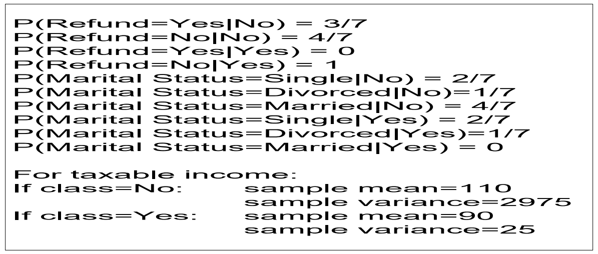
Given a test record

$$X=(\text{Refund}=\text{No}, \text{Married}, \text{Income}=120\text{K})$$

1 (Income, Class=No):
- ◆ Sample mean = 110, sample variance = 2975

2. (Income, Class=yes)
- ◆ Sample mean = 90, sample variance = 25

naive Bayes Classifier:



```
P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No:        sample mean=110
                    sample variance=2975
If class=Yes:       sample mean=90
                    sample variance=25
```

- $P(X|Class=No) = P(Refund=No|Class=No) \times P(Married|\ Class=No) \times P(Income=120K|\ Class=No) = 4/7 \times 4/7 \times 0.0072 = 0.0024$
- $P(X|Class=Yes) = P(Refund=No|\ Class=Yes) \times P(Married|\ Class=Yes) \times P(Income=120K|\ Class=Yes) = 1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since $P(X|No)P(No) > P(X|Yes)P(Yes)$

Therefore $P(No|X) > P(Yes|X)$

=> Class = No

Problem 4

(HW)

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

2. Decision Trees

A decision tree is a tree whose internal nodes are tests (on input patterns) and whose leaf nodes are categories (of patterns).
A decision tree is a hierarchical model for supervised learning whereby local region is identified in a sequence of recursive splits in a smaller number of steps.

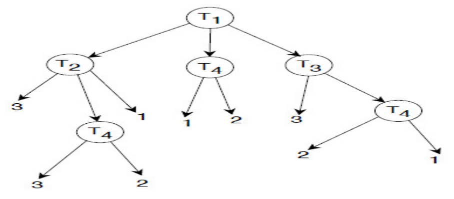


Figure 6.1: A Decision Tree

Here T1, T2, .. represents the tests and 1,2,3 represents the class labels.

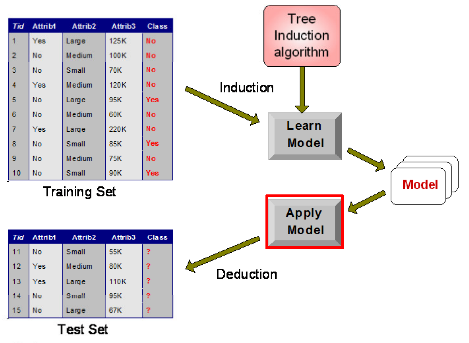
A decision is tree is composed of

1. Internal decision nodes – Each decision node m implements a test function $f_m(x)$ with discrete outcomes labeling the branches.
   Given an input, at each node, a test is applied and one of the branches is taken depending on the outcome.
   The process starts at the root and is repeated recursively until a leaf node is hit, at which point the value written in the leaf constitutes the output.
2. Leaf nodes --- It is labeled with the class label.

There are several dimensions along which decision trees might differ:

a. The tests might be multivariate (testing on several features of the input at once) or univariate (testing on only one of the features).
b. The tests might have two outcomes or more than two. (If all of the tests have two outcomes, we have a binary decision tree.)
c. The features or attributes might be categorical or numeric. (Binary-valued ones can be regarded as either.)
d. We might have two classes or more than two. If we have two classes and binary inputs, the tree implements a Boolean function, and is called a Boolean decision tree.

2.1 Decision Tree Learning



Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

During Training:
Construct a decision tree using the training dataset. This process is called as Tree induction.

During testing:
Evaluate the given input against the decision tree to compute the class label.

2.2 Tree Induction

There exist many decision trees for a given training set. We can select a smallest among them, where tree size is measured as the number of nodes in the tree.

Tree construction is with a greedy algorithm.

1. At each step, starting at the root with the complete training data, we look for the best split.
2. This splits the training data into two or more depending on whether the chosen attribute is numeric or discrete.
3. We continue splitting recursively with the corresponding subset until we do not need to split anymore, at which point leaf node is created and labeled.

If x_j is discrete, taking one of n possible values, the decision node checks the value of x_j and takes the corresponding branch, implementing a n-way split.

If x_j is numeric (ordered), the test is a comparison

$$f_m(x): x_j \geq w_{m0}$$

w_{m0} is a suitably chosen threshold value. In such case, Input space is divided into two regions,

$L_m: \{x | x_j \geq w_{m0}\}$ and $R_m: \{x | x_j < w_{m0}\}$ called as binary split.

Splitting Criteria

The splitting criteria is determined so that, ideally, the resulting partitions at each branch are as " pure " as possible as pure as possible.

- A partition is pure if all of the tuples in it belong to the same class

Hunts Algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. Let Dt be the set of training records that are associated with node t and
Y ={y1,y2,...,yc} be the class labels. The following is a recursive definition of Hunt's algorithm.

Step 1:
If all the records in Dt belong to the same class yt, then t is a leaf node labeled as yt.
Step 2:
If Dt contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in Dt are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

Measures for selecting the best split

A goodness of split is measured by an impurity measure. A split is pure if after the split, for all branches, all the instances choosing a branch belong to the same class. Among all, we look for the split that minimizes the impurity after the tree because we want to generate the smallest tree.

Let us say for node m, N_m is the number of training instances reaching node m. For the root node , it is N (size of dataset).

N_m^i of N_m belong to class C_i, with

$\sum N_m^i = N_m$

Given that an instance reaches node m, the estimate for the probability of class C_i is

$$p_m^i = \frac{N_m^i}{N_m}$$

Various measures of impurity/purity

1. **Entropy**
   A possible measure of impurity is Entropy which is calculated as,

Entropy gives a measure of randomness in data. A high value indicates the dataset is more random and hence contains the records belonging to many classes (uncertainty). For K classes at node m, entropy is defined as,

$$Entropy(D) = \sum_{i=1}^{K} -p_m^i log_2 p_m^i$$

Where **0logo≡1**

2. **Gini Index**
   It is given as,

$$G = \sum_{i=1}^{K} p_m^i (1 - p_m^i)$$

   Similar to Entropy, a high value indicates more randomness(uncertainty) in data.

Information Gain

Using the Entropy, we can compute Information gain. The attribute which has the highest gain will be selected as the attribute to be tested. For a dataset D and the attribute A, gain can be computed as,

$$Gain(D, A) = Entropy(D) - \sum_{j=1}^{v} \frac{|D_j|}{|D|} Entropy(D_j)$$

A -→ Refers to the attribute considered
v -- > total number of outcome for an attribute
For example Marital Status can be Single, Married or Divorced. So A=Marital Status, v=3
D_j is the subset of D satisfying the corresponding attribute condition.
For ex, D_1 refers to subset of D in which education is SSLC.

3. K-nearest neighbor Classification

All the above classification techniques are known as Eager learners and KNN is called as Lazy learner. Given a training set D of m labeled patterns (points),a nearest neighborhood procedure decides that some new pattern x', belongs to the same category as do its closest neighbors in D. More precisely, as k-nearest neighbor method assigns a new pattern x' , to that category to which the plurality of its k closest neighbors belong.
Using large value of k increases the chance of getting influenced by the noise pattern, but estimate would be better.

Distance metric
Simple Euclidean distance can be computed between any two points.

Consider a dataset (X, Y) where X is p-dimensional.
Distance between two points, x_i and x_j where x_i can be elaborated as , $(x_{i1}, x_{i2}, .., x_{ip})$ and x_j can be elaborated as $(x_{j1}, x_{j2}, .., x_{jp})$.
Distance is,

$$d_{ij} = \sqrt{\sum_{l=1}^{p} (x_{il} - x_{jl})^2}$$

Sometimes it could be scaled by an amount a_l(scale factor)
So the equation will be,

$$d_{ij} = \sqrt{\sum_{l=1}^{p} a_l^2 (x_{il} - x_{jl})^2}$$

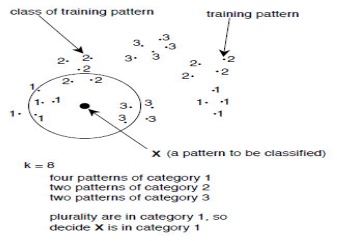

class of training pattern
training pattern
X (a pattern to be classified)
k = 8
four patterns of category 1
two patterns of category 2
two patterns of category 3
plurality are in category 1, so
decide X is in category 1

Figure 5.3: An 8-Nearest-Neighbor Decision

Disadvantage: Memory intensive as large dataset need to be stored in memory to achieve good performance.

Algorithm

Steps:

1. Let k be the number of nearest neighbor and D be the set of training examples
2. For each test example z=(x', y') do
    1. Compute d(x', x), the distance between z and every example , (x, y) ∈ D.
    2. Select $D_z \subseteq D$, the set of k closest training examples to z
    3. $y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \varepsilon D_z} I(v = y_i)$

Example 1

Example	x	y
1	0.5	-
2	3.0	-
3	4.5	+
4	4.6	+
5	4.9	+
6	5.2	-
7	5.3	-
8	5.5	+
9	7.0	-
10	9.5	-

Test point 5.0

Compute distance

	X	y	Distance
1	0.5	-	4.5
2	3.0	-	2
3	4.5	+	0.5
4	4.6	+	0.4
5	4.9	+	0.1
6	5.2	-	0.2
7	5.3	-	0.3
8	5.5	+	0.5
9	7.0	-	2
10	9.5	-	4.5

1. **1- nearest neighbor – 5th , +**
2. **3 nearest neighbor – 5, 6, 7 , -**
3. **5 nearest neighbor – 5,6 ,7, 4, 3, +**
4. **9 nearest neighbor – 5,6,7,4,3,8, 2, 9, 1 , -**

Example 2

Example	x1 (Acid Durability in seconds)	x2 (Strength)	Classification
1	7	7	Bad
2	7	4	Bad
3	3	4	Good
4	1	4	Good

Test data=(3, 7) k=3

Solution: Distance is 1=4, 2=5 ,3= 3,4= 3.6
Ascending order 3, 4, 1, 5
Consider the first 3 points {3, 4,1} and the corresponding class labels {Good, Good, Bad}

The assigned label is through majority voting --- **Good**

Unit 4

Un Supervised Learning

1. Clustering – Introduction

The aim of unsupervised learning is to find some regularity in the given data. There is no supervisor in this technique. Here 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,

i. discover groups of similar examples within the data, where it is called *clustering*,
ii. to determine the distribution of data within the input space, known as *density estimation*,
iii. to project the data from a high-dimensional space down to two or three dimensions for the purpose of *visualization*.

Clustering
Cluster analysis groups data based only on information found in the data that describes the objects and their relationships. The goal is that the objects within the group be similar (or related) to one another and different from (or unrelated to) the objects in other groups.
The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.

Types of clustering
Clustering can be categorized into two types based on whether the clusters are nested or not.

1. Partitional clustering: It is simply a division of the set of objects into non-overlapping set of subsets (clusters) such that each data point (object) I in exactly one subset.
2. Hierarchical clustering: It is a set of nested clusters organized as a tree. Each node in (cluster) in the tree (except for the leaf nodes) is the union of its children (subclusters) and the root of the tree containing all the objects.

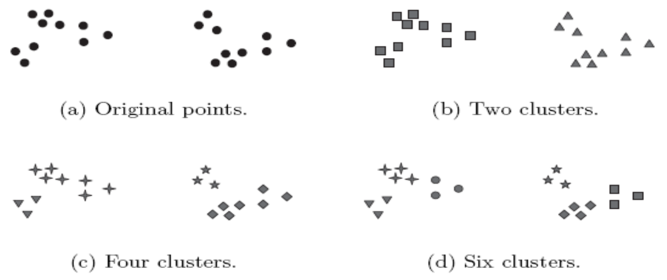


Figure 8.1. Different ways of clustering the same set of points.

1.1 K-means Clustering
It is a type of partitional clustering.

```
1: Select $K$ points as the initial centroids.
2: repeat
3:    Form $K$ clusters by assigning all points to the closest centroid.
4:    Recompute the centroid of each cluster.
5: until The centroids don't change
```

Formal explanation
Suppose we have a dataset $\{x_1, x_2, \ldots, x_N\}$ consisting of N observations, of a p-dimensional input. Our goal is to partition the dataset into K clusters. Suppose we have a vector μ_k where k=1, …, K which represents each cluster with cluster centroid. Our goal is then to find an assignment of data points to clusters, as well as a set of vectors $\{\mu_k\}$ such that the sum of the squares of the distances of each data point to its closest vector μ_k, is a minimum.

For each data point x_n, we introduce a corresponding set of binary vectors $r_{nk} \in \{0,1\}$, describing which of K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then $r_{nk} = 1$

The objective function, called as a distortion measure is defined as

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

which represents the sum of the squares of the distances of each data point to its assigned vector μ_k . Our goal is to find values for the r_{nk} and the μ_k so as to minimize *J*. It means that every data point should be assigned to its closest center as much as possible.
More formally r_{nk} can be represented as,

$$r_{nk} = \begin{cases} 1, & if\ k = \underset{j}{\operatorname{argmin}} \|x_n - \mu_j\|^2 \\ 0, & otherwise \end{cases}$$

The value μ_k is defined as

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

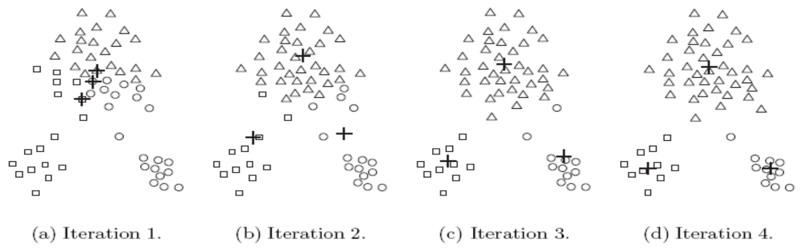


Figure 8.3. Using the K-means algorithm to find three clusters in sample data.

Problem: Consider the dataset given below,

Data Point	X1	X2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

Suppose A (Cluster 1) and C (Cluster 2) chosen as initial clusters means randomly for cluster 1 and cluster2 respectively.

1. Compute the distance from each point to cluster centers

Data Point	**Cluster 1**	**Cluster 2**
A	0	1.4
B	1	2.2
C	1.4	0
D	3.2	2.8
E	4.5	4.2

2. Assign each point to the closest centroid

Data Point	**distance with Cluster 1**	**Distance with Cluster 2**	**Cluster**
A	0	1.4	**1**
B	1	2.2	**1**
C	1.4	0	**2**
D	3.2	2.8	**2**
E	4.5	4.2	**2**

3. Re-compute the cluster centroid
   Cluster 1: Includes the points A (1,1) and B (1,0) so
   (1+1)/2=1, (1+0)/2=0.5
   Cluster 1 Centroid : (1, 0.5)

   Cluster 2: includes the points C (0,2), D (2,4), E(3,5)

   (0+2+3)/3=1.666 (2+4+5)/3=3.6666
   Cluster 2 centroid: (1.7, 3.7)

   Repeat step 1 to 3

   Distance Calculation and assignment

Data Point	**Distance Cluster 1**	**Distance Cluster 2**	**Assigned Cluster**
A	0.5	2.7	1
B	0.5	3.7	1
C	1.8	2.4	1
D	3.6	0.5	2
E	4.9	1.9	2

   Re-compute cluster means
   Cluster1 : (0.7, 1)
   Cluster 2: (2.5, 4.5)

   Repeat until converge

1.2 Hierarchical Clustering

It works based on the similarities of instances, without any other requirement on the data. The aim is to find groups such that instances in a group are similar to each other than instances in different groups.

There are two types,

1. Agglomerative clustering
   a. It starts with N groups, each initially containing one instance, merging smaller groups to form larger groups, until there is single one.
2. Divisive Clustering
   a. It goes in the other direction, starting with a single group and dividing large groups into smaller groups, until each group contains a single instance.

Agglomerative Clustering

At each step, agglomerative clustering algorithm, chooses the closest groups to merge. There are two types,

1. Single-link clustering: Distance is defined as the smallest distance between all possible pair of elements of the two groups

$$d(G_i, G_j) = \min_{X^r \in G_i, X^s \in G_j} d(x^r, x^s)$$

2. Complete link clustering: Distance between two group is taken as the largest distance between all possible pairs.

$$d(G_i, G_j) = \max_{X^r \in G_i, X^s \in G_j} d(x^r, x^s)$$

The result after the agglomerative clustering is usually drawn as a hierarchical structure known as, dendrogram.

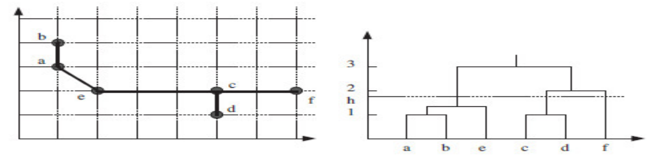


Figure 7.5 A two-dimensional dataset and the dendrogram showing the result of single-link clustering is shown. Note that leaves of the tree are ordered so that no branches cross. The tree is then intersected at a desired value of *h* to get the clusters.

Example

Data Point	X1	X2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

1. Each of the point is an individual cluster

Compute the distance between each points

	A	B	C	D	E
A	0	1	1.41	3.16	4.47
B	1	0	2.23	4.12	5.38
C	1.41	2.23	0	2.82	4.24
D	3.16	4.12	2.82	0	1.41
E	4.47	5.38	4.24	1.41	0

2. Group the points based on nearest distance
   A & B are nearer to each other , so it will merged together to form a cluster , we call G-AB. Rewrite the distance matrix

	G-AB	C	D	E
G-AB	0	1.41	3.16	4.47
C	1.41	0	2.82	4.24
D	3.16	2.82	0	1.41
E	4.47	4.24	1.41	0

Step 2 is continued until we get a single cluster

Here, G-AB and C is merged into one group , call it as G-ABC
Group D & E is merged into one group , call it as G – DE

Rewrite the distance matrix

	G-ABC	G-DE
G-ABC	0	2.82
G-DE	2.82	0

Example 2: Apply Hierarchical clustering

	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.41	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

3. Evaluation Measures and Combining Learners

Evaluation Measures: Cross-validation and Re-sampling, Measuring Error, Hypothesis Testing,

Combining Learners: Voting, Bagging, Boosting

Reference (Textbook : Ethem Alpaydin)

After designing a classifier model, how can we assess the expected error rate of a classifier algorithm on a problem?

1. So, we need a validation set (test set) different from the training set
2. Even over a validation set, just one run may not be enough. We would like to have several runs to average over sources of randomness.

Cross validation & Resampling methods

From a given dataset X, we generate multiple training and validation set pairs.

1. K-Fold cross validation:
   a. The dataset is divided into K equal sized parts.
2. 5 X 2Cross validation
   a. The dataset is divided into two subsets and then swapped again. Repeated for 5 times.
3. Bootstrapping:
   a. Generates new samples from the original sample with replacement
   b. Best for small dataset.

Measuring Error

Confusion matrix for a two class classification is given as ,

	Predicted Class	
True Class	Yes	No
Yes	TP: True Positive	FN: False Negative
No	FP: False Positive	TN: True Negative

The error rate can be defined as,
Error rate = (|FN| + |FP|) / N
N is the total number of examples in the validation set
N= |FN| + |FP|+|TN| + |TP|
If there are more than two (K>2) classes, then confusion matrix is a,
K X K matrix , such that the entry (i, j) contains the number of instances that belong to Ci, but are recognized as Cj.
The diagonal entries indicates the number of correct classification
Receiver Operating Curve (ROC)
It is a curve which shows the hit rate versus false alarm rate, namely,
|TP|/(|TP|+|FN|) vs |FP|/(|FP|+|TN|)

Hypothesis Testing

Here, certain claim is made about the population. We define a statistic that obeys a certain distribution if the hypothesis is correct. If the statistic calculated from the sample has a high enough probability of being drawn from this distribution, then we accept the hypothesis; otherwise reject it.

Let us say, we have a population from a normal distribution with unknown mean μ and known variance σ^2. We want to test a specific hypothesis about μ, for example, whether it is equal to a specified constant μ_0. It is denoted as H_0 and called the null hypothesis.

$$H_0: \mu = \mu_0$$

Against the alternative hypothesis

$$H_1: \mu \neq \mu_0$$

Level of significance: It is the level of confidence with which we can accept the hypothesis. Denote

s as α(0.1, 0.05, 0.01 etc). We accept the hypothesis with level of significance α, if μ_0 lies in the 100(1- α)[90%, 95%,..] percent confidence interval.

Type 1 error: When we reject the hypothesis when it is actually true.

Type 2 error: When we accept the hypothesis when it is actually false.

One sided Test & Two sided test

In two sided test the null hypothesis and alternative hypothesis will be of the form,

$$H_0: \mu = \mu_0$$

Against the alternative hypothesis

$$H_1: \mu \neq \mu_0$$

In one sided test, the null hypothesis and alternative hypothesis will be of the form,

$$H_0: \mu = \mu_0$$

Against the alternative hypothesis

$$H_1: \mu > \mu_0$$

or

$$H_1: < \mu_0$$

Combining Multiple Learners

We discussed many different learning algorithms in the previous chapters. Though these are generally successful, no one single algorithm is always the most accurate. Now, we are going to discuss models composed of multiple learners that complement each other so that by combining them, we attain higher accuracy.

Generating Diverse Learners
Since there is no point in combining learners that always make similar decisions, the aim is to be able to find a set of *diverse* learners who differ in their decisions so that they complement each other. At the same time, there cannot be a gain in overall success unless the learners are accurate, at least in their domain of expertise. We therefore have this double task of maximizing individual accuracies and the diversity between learners.

1. Different algorithms
2. Different Hyperparameters
3. Different input representation
4. Different training sets

Model Combination Schemes
There are also different ways the multiple base-learners are combined to generate the final output:

1. ***Multiexpert combination*** methods have base-learners that work in *parallel*. These methods can in turn be divided into two:
   _ In the ***global*** approach, also called *learner fusion*, given an input, all base-learners generate an output and all these outputs are used. Examples are *voting* and *stacking*.
   _ In the ***local*** approach, or *learner selection*, for example, in *mixture of experts*, there is a *gating* model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.

2. ***Multistage combination*** methods use a *serial* approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. The idea is that the base-learners (or the different representations they use) are sorted in increasing complexity so that a complex base-learner is not used (or its complex representation is not extracted) unless the preceding simpler base-learners are not confident. An example is *cascading*.

1. **Voting**
   The simplest way to combine multiple classifiers is by *voting*, which corresponds to taking a linear combination of the learners

$$y_i = \sum_j w_j d_{ji}$$

d_{ji} is the ith output from jth learner.

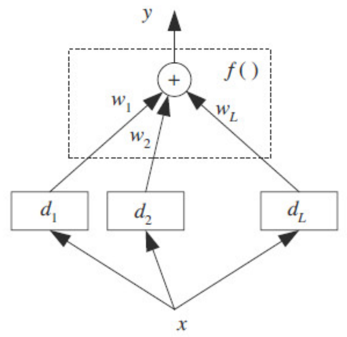


Figure 17.1 Base-learners are d_j and their outputs are combined using $f(\cdot)$. This is for a single output; in the case of classification, each base-learner has K outputs that are separately used to calculate y_i, and then we choose the maximum. Note that here, all learners observe the same input; it may be the case that different learners observe different representations of the same input object or event.

In the simplest case, all learners are given equal weight and we have *simple voting* that corresponds to taking an average.

2. **Bagging**

Bagging is a voting method whereby base-learners are made different by training them over slightly different training sets.

Generating *L* slightly different samples from a given sample is done by bootstrap, where given a training set X of size *N*, we draw *N* instances randomly from X *with replacement*.

Because sampling is done with replacement, it is possible that some instances are drawn more than once and that certain instances are not drawn at all. When this is done to generate *L* samples X*j, j* = 1*, . . ., L*, these samples are similar because they are all drawn from the same original sample, but they are also slightly different due to chance. The base-learners *dj* are trained with these *L* samples X*j* .

3. **Boosting**

In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method.
In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the previous learners.
Given a large training set, we randomly divide it into three.
We use X1 and train *d*1.
We then, take X2 and feed it to *d*1.
We take all instances misclassified by *d*1 and also as many instances on which *d*1 is correct from X2, and these together form the training set of *d*2.

We then take X3 and feed it to *d*1 and *d*2.
The instances on which *d*1 and *d*2 disagree form the training set of *d*3. During testing, given an instance, we give it to *d*1 and *d*2; if they agree, that is the response, otherwise the response of *d*3 is taken as the output.

Disadvantage is that it requires large training set.