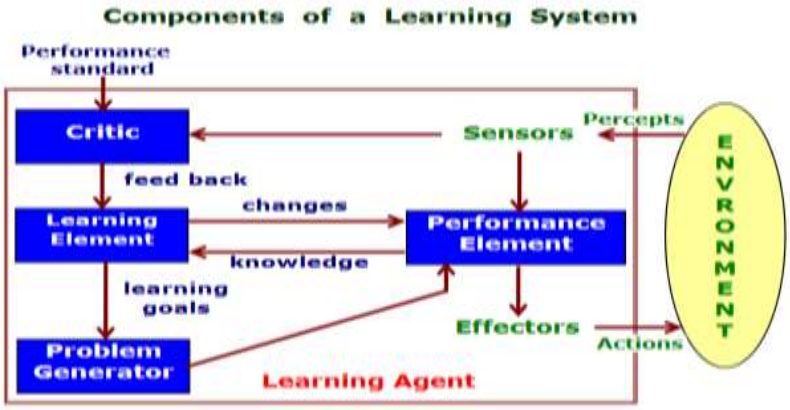
Learning

Learning denotes changes in a system that enables the system to do the same task more efficiently next time.

Definition of Learning System/Agent

A learning agent is an entity that is capable of perceiving and do action. An agent can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. A learning agent has a learning component which improves its action to the perceived inputs as time proceeds.



Components of a Learning System

- **Performance Element:** The Performance Element is the agent itself that acts in the world. It takes in percepts and decides on external actions.
- **Learning Element:** It responsible for making improvements, takes knowledge about performance element and some feedback, determines how to modify performance element.
- **Critic:** Tells the Learning Element how agent is doing (success or failure) by comparing with a fixed standard of performance.
- **Problem Generator:** Suggests problems or actions that will generate new examples or experiences that will aid in training the system further.

Aspects of developing a learning system

- Training Data
- Concept Representation
- Function Approximation

To make a system learn, a training data is to be provided. For example if we want to make a system learn to differentiate between animals and birds, then we provide it with some examples of animals and some example of birds (labeled dataset i.e. each example also contains information of whether it is a bird or animal).

From these examples, the system learns some concept, for example, birds have wings, a beak and two legs whereas animals have four legs and have a nose, etc.

After concept representation the system should be able to identify new objects. However, the learning system needs to use function approximation as some of the new example may exactly not match the concepts i.e. an elephant has a trunk (which is neither a nose nor a beak) so the system should be able to approximate whether it is an animal or a bird.

Machine Learning

Definition

Tom M. Mitchell provided a widely quoted, more formal definition: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."

Goals and Applications of ML

The goal of machine learning is to design and develop algorithms that allow computers to learn from historical/training data.

- Stock Market Prediction
- Fraud Detection
- Speech and Handwriting Recognition
- Natural Language Processing
- Computer Vision
- Robotics
- Game Playing
- Recommender Systems
- Sentiment Analysis
- Bioinformatics
- Medical Diagnosis

Types of Machine Learning

- **Supervised learning**: The computer is presented with example inputs and their desired outputs, and the goal is to learn a general rule that maps inputs to outputs. Supervised learning is used in predictive analytics i.e. classification and regression. Classification deals with discrete output, and regression deals with continuous output.
- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find patterns in its input. Unsupervised learning deals with pattern mining and clustering.
- **Reinforcement learning**: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The program is provided feedback in terms of rewards and punishments as it navigates its problem space. Reinforcement learning differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. It finds its main application in control theory, and robotics.

Classification

Classification aims at predicting a discrete value by creating a model from training dataset.

Overview of classification process

- Training : a training (labeled) dataset is required to build the classifier.
- Test : the instances of test dataset are used for prediction (labeled to know if predicted output is correct or wrong).
- Validation dataset : this data set is used to minimize over-fitting (used during training).
- Overfitting : If the accuracy over the training data set increases, but the accuracy over then validation data set stays the same or decreases, then you're over-fitting your neural network and you should stop training.

Classification Families

1. Nearest Neighbors

Nearest Neighbors classifiers is the simplest of all the classifiers. In this classifier, for the new instance to be classified, the class of its nearest neighbors is identified and assigned to it. To find the nearest neighbor a measure like Euclidean distance or cosine similarity may be used. kNN is a popular nearest neighbor algorithm.

2. Decision Trees

In a decision tree, a tree is generated such that the attributes are nodes of the tree and values of the attributes are the branches. The aim is to select the order of nodes from top to bottom in such a manner so as to minimize the height of the tree. Decision Tree is based on the concept of entropy in the dataset.

3. Discriminative models

A (linear) discriminative model aims to find a hyper-plane (e.g. a sub-space of one dimension less than the actual space e.g. a line for 2D space) which divides the space into two halves such that each half contains elements belonging to a single class. Examples of discriminative models are Logistic Regression, Perceptrons and Support Vector Machines.

A non-linear discriminative model aims to handle datasets which can’t be separated by a linear hyper-plane. To handle such datasets a kernel function is used to map the dataset into higher dimensional space where it can be separated by a hyper-plane.

4. Probablistic Models

A probabilistic model gives the probability of each class for the given data instance. The class whose probability comes out to be highest for the instance is assigned. Examples of probabilistic models are logistic regression and Naïve Bayes.

A probabilistic model can be further classified as conditional and generative. A conditional model uses a discriminant function to calculate the probabilities e.g. logistic regression, whereas a generative model models the probability distribution of the given dataset and based on this distribution predicts the probabilities of the instance belonging to each class e.g. Naïve Bayes.

Evaluation of classification algorithms

The performance of a classifier can be measured by keeping some part of training dataset for testing; in k-fold validation data is divided into k-parts, k-1 parts for training, 1 part for testing.

Confusion Matrix -> 2X2 matrix reporting number of True Positives, True Negatives, False Positives, False Negatives.

Area Under (ROC) Curve -> Receiver Operating Characteristic graph has False Positive Rate [FPR=FP/(FP+TN)] and True Positive Rate [TPR=TP/(TP+FN)] as X and Y axes (range 0-1). High value is better value (100% is maximum).

Kappa score -> Measures agreement b/w two ratings (manual "A" and predicted "B"). An N x N histogram matrix O is constructed, such that Oi,j corresponds to the number of search records that received a rating "i" by A and a rating "j" by B. An N-by-N matrix of weights, w, is calculated, where $w_{ij}=(i-j)^2/(N-1)^2$. An N-by-N histogram matrix of expected ratings, E, is calculated, assuming that there is no correlation between rating scores. This is calculated as the outer product between each rater's histogram vector of ratings, normalized such that E and O have the same sum. From these three matrices, the quadratic weighted kappa is calculated as: $K=1-(\Sigma w_{ij}O_{ij})/(\Sigma w_{ij}E_{ij})$.

I. kNN

In kNN, the similarity of the test data point with every point of training data is calculated using a similarity measure like euclidean distance, mahalanobis distance, cosine similarity, pearson correlation, etc., and k most similar points are selected.

- In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

Example:

X_1=(0.8,0.8,1)
X_2=(1.0,1.0,1)
X_3=(1.2,0.8,1)
X_4=(0.8,1.2,1)
X_5=(1.2,1.2,1)
X_6=(4.0,3.0,2)
X_7=(3.8,2.8,2)
X_8=(4.2,2.8,2)
X_9=(3.8,3.2,2)
X_{10}=(4.2,3.2,2)
X_{11}=(4.4,2.8,2)
X_{12}=(4.4,3.2,2)
X_{13}=(3.2,0.4,3)
X_{14}=(3.2,0.7,3)
X_{15}=(3.8,0.5,3)
X_{16}=(3.5,1.0,3)
X_{17}=(4.0,1.0,3)
X_{18}=(4.0,0.7,3)

Predict class for X=(3.0,2.0) by kNN with k=1,k=3 and k=5.

Answer

For k=1, closest neighbor is X_{16} (euclidean distance=1.12) with class 3.

II. Naive Bayes classifier

Naive Bayes classifier is based on Bayes theorem which says that
$P(H|E) = P(E|H) * P(H) / P(E)$
where H is some hypothesis based on some evidence E e.g. evidence=fever, hypothesis=dengue.
P(E), P(H), P(E|H) are priori-probabilities which are used to calculate conditional probability P(H|E).
In Naive Bayes, we have to predict the class (C) of an example(X), so the equation can be re-written as
$P(C|X) = P(X|C) * P(C) / P(X)$

Let's understand Naive Bayes by an example. Suppose, we are given the following dataset for training the classifier, where "Play" is the output with 2 labels "yes" and "no".

Outlook	Temperature	Humidity	Windy	Play
Sunny	hot	high	false	no
Sunny	hot	high	true	no
Overcast	hot	high	false	yes
Rainy	mild	high	false	yes
Rainy	cool	normal	false	yes
Rainy	cool	normal	true	no
Overcast	cool	normal	true	yes
Sunny	mild	high	false	no
Sunny	cool	normal	false	yes
Rainy	mild	normal	false	yes
Sunny	mild	normal	true	yes
Overcast	mild	high	true	yes
Overcast	hot	normal	false	yes
Rainy	mild	high	true	no

So we have to build a classifier using the above training set i.e. we have to calculate priori probabilities P(C), P(X|C) and P(X).

As we have only two classes in out training dataset, therefore P(C) is P(yes) and P(no).
P(C) = number of examples belonging to class C / total examples
P(yes) = 9/14
P(no) = 5/14

P(X) = number of examples having X / total examples
P(sunny) = 5/14
P(overcast) = 4/14
P(rainy) = 5/14
P(hot) = 4/14
P(mild) = 6/14
P(cool) = 4/14

P(high) = 7/14
P(normal) = 7/14
P(false) = 8/14
P(true) = 6/14

P(X|C) = number of times X is associated with C / number of examples belonging to class C
P(sunny|yes) = 2/9, P(sunny|no) = 3/5
P(overcast|yes) = 4/9, P(overcast|no) = 0/5
P(rainy|yes) = 3/9, P(rainy|no) = 2/5
P(hot|yes) = 2/9, P(hot|no) = 2/5
P(mild|yes) = 4/9, P(mild|no) = 2/5
P(cool|yes) = 3/9, P(cool|no) = 1/5
P(high|yes) = 3/9, P(high|no) = 4/5
P(normal|yes) = 6/9, P(normal|no) = 1/5
P(false|yes) = 6/9, P(false|no) = 2/5
P(true|yes) = 3/9, P(true|no) = 3/5

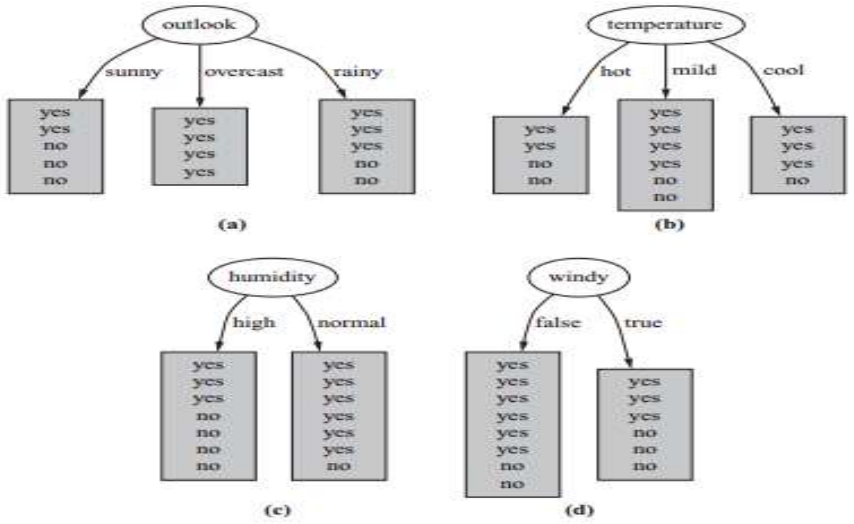
We have obtained all the 3 priori probabilities from the training dataset. Now, we want to classify a new unclassified example.
Let the example be {sunny,cool,high,true} and we have to predict it's class. The class can be predicted using the formula
P(C|X) = {[P(C)*ΠP(X|C)] } / ΠP(X)
Case I: Yes
P(yes|sunny,cool,high,true) = P(yes) * P(sunny|yes) * P(cool|yes) * P(high|yes) * P(true|yes) / P(sunny) * P(cool) * P(high) * P(true) = 9/14 * 2/9 *3/9 * 3/9 * 3/9 / ΠP(X)
Case II : No
P(no|sunny,cool,high,true) = P(no) * P(sunny|no) * P(cool|no) * P(high|no) * P(true|no) / P(sunny) * P(cool) * P(high) * P(true) = 5/14 * 3/5 * 1/5 * 4/5 * 3/5 / ΠP(X)
Result:
As P(X) is same in both equations, we can ignore it giving
P(yes|sunny,cool,high,true) = 0.00529
P(no|sunny,cool,high,true) = 0.02057
As P(no|sunny,cool,high,true) > P(yes|sunny,cool,high,true), therefore we assign label "no" to it.

III. Decision Tree

Decision tree is a technique for data classification in data mining. As an example, we can consider a decision on whether the conditions are suitable to play or not. The various attributes which can be used to make the decision are Outlook, Temperature, Humidity, Windy.

Outlook	Temperature	Humidity	Windy	Play
Sunny	hot	high	false	no
Sunny	hot	high	true	no
Overcast	hot	high	false	yes
Rainy	mild	high	false	yes
Rainy	cool	normal	false	yes
Rainy	cool	normal	true	no
Overcast	cool	normal	true	yes
Sunny	mild	high	false	no
Sunny	cool	normal	false	yes
Rainy	mild	normal	false	yes
Sunny	mild	normal	true	yes
Overcast	mild	high	true	yes
Overcast	hot	normal	false	yes
Rainy	mild	high	true	no

As, we can see in our dataset, 14 rules are given to make the decision. If we create a node for each attribute, they will look like:



As we can see, if we see the attribute “outlook”, it has the value “sunny” in the dataset 5 times, out of which 2 result in ‘yes’ for outcome “Play” and 3 result in ‘no’ for outcome “Play”.
We have to make a decision on which node to be used as root node. Different measures are used to make this decision.

ID3 algorithm

ID3 (Iterative Dichotomiser) uses the concept of information gain to make the decision. Information gain is based on entropy (information value), whose unit is bits.

Entropy

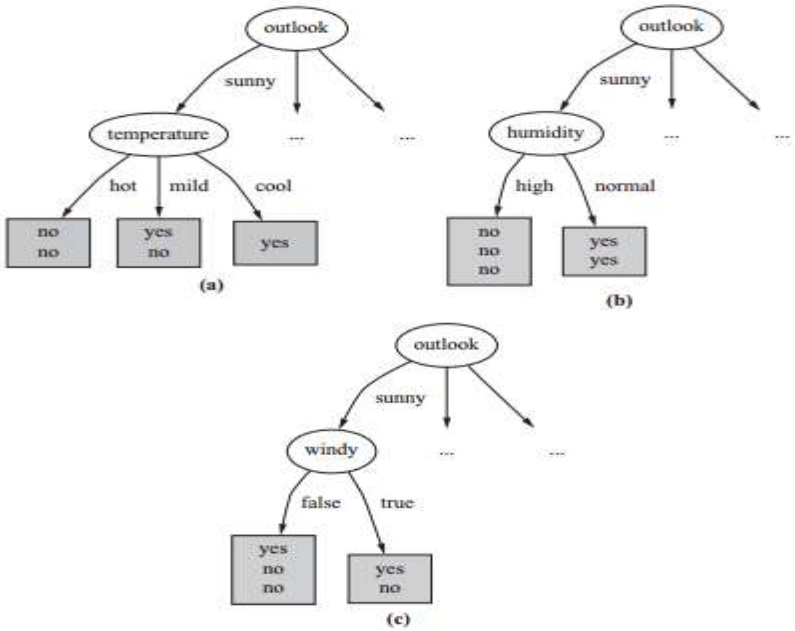
For any element X, entropy can be calculated using the formula:
$ent(P1,P2) = -(P1*\log_2 P1)-(P2*\log_2 P2) = -(P1\log_2 P1) – (1-P1)\log_2(1-P1)$
where P1, P2 denotes the fraction of X belonging to first class and second class.
e.g.
Entropy of outlook = ent(9/14,5/14) = -(9/14)log 2(9/14)-(5/14)log 2(5/14) = 0.940
Entropy of sunny = ent(2/5,3/5) = -(2/5)log 2(2/5)-(3/5)log 2(3/5) = 0.971 bits.

Information Gain

Information Gain = Entropy of parent node – Entropy of child nodes
e.g.
Info-gain of outlook = entropy of outlook – [(5/14) * entropy of sunny)] + [(4/14) * entropy of overcast] + [(5/14) * entropy of rainy] = 0.940 – [(5/14)*0.971 + (4/14)*0 + (5/14)*0.940] = 0.940 – 0.693 = 0.247

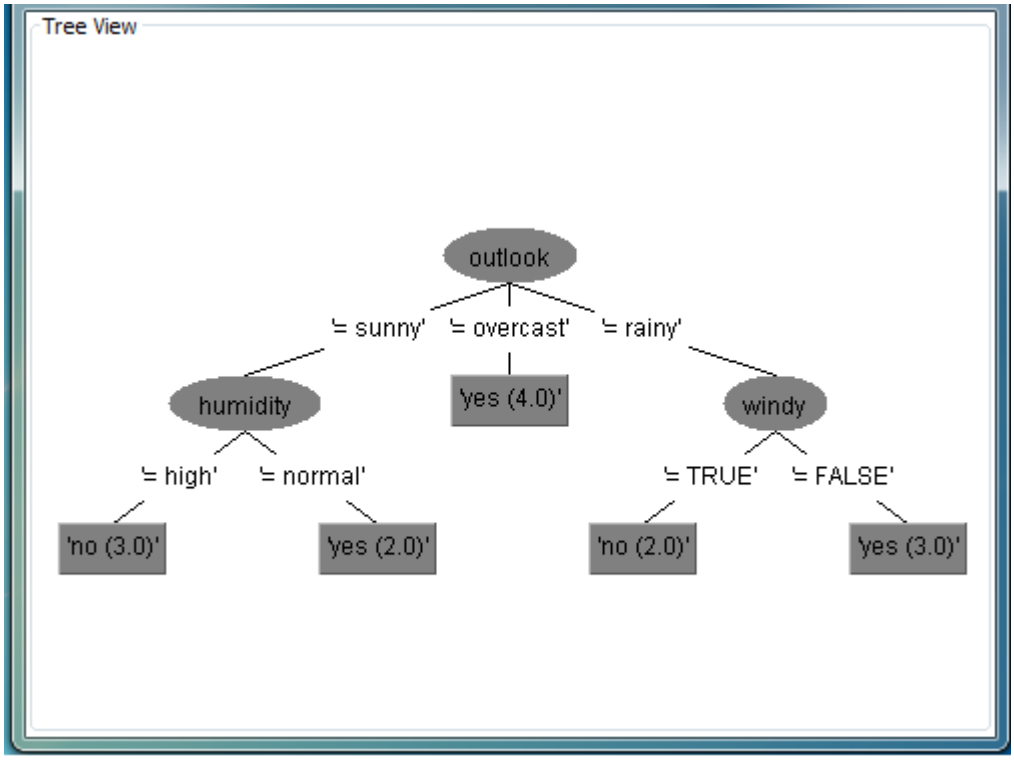
For the given dataset: gain(outlook) = 0.247 bits, gain(temperature) = 0.029 bits, gain(humidity) = 0.152 bits, gain(windy) = 0.048 bits.

Therefore, we select “outlook” as the root of the tree.



Now we have 3 branches for this root (sunny, overcast, rainy). We have to select next node for each of these branches. Each branch can point to nodes temperature/humidity/windy.
Calculating gain for each of these sub-trees:
gain(temperature) = 0.571 bits
gain(humidity) = 0.971 bits
gain(windy) = 0.020 bits
Hence, next node for branch sunny is “humidity”.

The process terminates when all leaf nodes are pure i.e. contain either all yes or all no.



IV. Linear/Gaussian Discriminant Analysis

LDA is a generative learner as it makes assumption about the data distribution.
LDA makes some simplifying assumptions about your data:

- That your data is Gaussian, that each variable is shaped like a bell curve when plotted.
- That each attribute has the same variance, that values of each variable vary around the mean by the same amount on average.

With these assumptions, the LDA model estimates the mean and variance from your data for each class.

In our example, $\mathbf{x} = \begin{bmatrix} 2.95 & 6.63 \\ 2.53 & 7.79 \\ 3.57 & 5.65 \\ 3.16 & 5.47 \\ 2.58 & 4.46 \\ 2.16 & 6.22 \\ 3.27 & 3.52 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}$

$\mathbf{x}_k$ = data of row k . For example, $\mathbf{x}_3 = [3.57 \quad 5.65]$, $\mathbf{x}_7 = [3.27 \quad 3.52]$

g = number of groups in $\mathbf{y}$. In our example, $g = 2$

$\mathbf{x}_i$ = features data for group i . Each row represents one object; each column stands for one feature. We separate $\mathbf{x}$ into several groups based on the number of category in $\mathbf{y}$.

$\mathbf{x}_1 = \begin{bmatrix} 2.95 & 6.63 \\ 2.53 & 7.79 \\ 3.57 & 5.65 \\ 3.16 & 5.47 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 2.58 & 4.46 \\ 2.16 & 6.22 \\ 3.27 & 3.52 \end{bmatrix}$

$\boldsymbol{\mu}_i$ = mean of features in group i , which is average of $\mathbf{x}_i$

$\boldsymbol{\mu}_1 = [3.05 \quad 6.38]$, $\boldsymbol{\mu}_2 = [2.67 \quad 4.73]$

$\boldsymbol{\mu}$ = global mean vector, that is mean of the whole data set.

In this example, $\boldsymbol{\mu} = [2.88 \quad 5.676]$

$\mathbf{x}_i^o$ = mean corrected data, that is the features data for group i , $\mathbf{x}_i$, minus the global mean vector $\boldsymbol{\mu}$

$\mathbf{x}_1^o = \begin{bmatrix} 0.060 & 0.951 \\ -0.357 & 2.109 \\ 0.679 & -0.025 \\ 0.269 & -0.209 \end{bmatrix}$, $\mathbf{x}_2^o = \begin{bmatrix} -0.305 & -1.218 \\ -0.732 & 0.547 \\ 0.386 & -2.155 \end{bmatrix}$

$\mathbf{c}_i = \frac{\left(\mathbf{x}_i^o\right)^T \mathbf{x}_i^o}{n_i}$ = covariance matrix of group i

$\mathbf{c}_1 = \begin{bmatrix} 0.166 & -0.192 \\ -0.192 & 1.349 \end{bmatrix}$, $\mathbf{c}_2 = \begin{bmatrix} 0.259 & -0.286 \\ -0.286 & 2.142 \end{bmatrix}$

$C(r,s) = \frac{1}{n}\sum_{i=1}^{g} n_i \cdot c_i(r,s)$ = pooled within group covariance matrix. It is calculated for each entry (r,s) in the matrix. In our example, $\frac{4}{7}\cdot 0.166 + \frac{3}{7}\cdot 0.259 = 0.206$, $\frac{4}{7}(-0.192) + \frac{3}{7}(-0.286) = -0.233$ and $\frac{4}{7}\cdot 1.349 + \frac{3}{7}\cdot 2.142 = 1.689$, therefore

$\mathbf{C} = \begin{bmatrix} 0.206 & -0.233 \\ -0.233 & 1.689 \end{bmatrix}$

The inverse of the pooled covariance matrix is $\mathbf{C}^{-1} = \begin{bmatrix} 5.745 & 0.791 \\ 0.791 & 0.701 \end{bmatrix}$

$\mathbf{p}$ = prior probability vector (each row represent prior probability of group i). If we do not know the prior probability, we just assume it is equal to total sample of each group divided by the total samples, that is $p_i = \frac{n_i}{N}$

$\mathbf{p} = \begin{bmatrix} 0.571 \\ 0.429 \end{bmatrix} = \begin{bmatrix} 4/7 \\ 3/7 \end{bmatrix}$

Discriminant function

$f_i = \boldsymbol{\mu}_i \mathbf{C}^{-1} \mathbf{x}_k^T - \frac{1}{2}\boldsymbol{\mu}_i \mathbf{C}^{-1} \boldsymbol{\mu}_i^T + \ln(p_i)$

We should assign object k to group i that has maximum f_i

Factory "ABC" produces very expensive and high quality chip rings that their qualities are measured in term of curvature and diameter. Result of quality control by experts is given in the table below.

Curvature	Diameter	Quality Control Result
2.95	6.63	Passed
2.53	7.79	Passed
3.57	5.65	Passed
3.16	5.47	Passed
2.58	4.46	Not Passed
2.16	6.22	Not Passed
3.27	3.52	Not Passed

The results of our computation are given in MS Excel as shown in the figure below.

	A	B	C	D	E	F	G	H	I	J	K
1	**Linear Discriminant Analysis: 2 Category 2 Dimensional Data**										
2											
3	**Training Data, D**				**Mean Corrected Data**			Discriminant function			**Results**
4	class	X1	X2		X1o	X2o		f1	f2		**Classificati**
5	1	2.95	6.63		0.060	0.951		55.220	53.071		1
6	1	2.53	7.79		-0.357	2.109		53.774	51.394		1
7	1	3.57	5.65		0.679	-0.025		62.476	59.589		1
8	1	3.16	5.47		0.269	-0.209		51.953	50.764		1
9	2	2.58	4.46		-0.305	-1.218		32.028	34.313		2
10	2	2.16	6.22		-0.732	0.547		34.554	35.757		2
11	2	3.27	3.52		0.386	-2.155		41.174	42.414		2
12	**prediction**	2.81	5.46		-0.078	-0.219		44.049	44.085		2
13											
14	average	2.888	5.676								
15	std dev	0.454	1.300								

Appendix-1: Generative Learners

A generative learner tries to learn the model that generates the data. It makes assumptions about the distributions.

Generative models contrast with discriminative models, in that a generative model is a full probabilistic model of all variables, whereas a discriminative model provides a model only for the target variable(s) conditional on the observed variables. Thus a generative model can be used, for example, to simulate (i.e. *generate*) values of any variable in the model, whereas a discriminative model allows only sampling of the target variables conditional on the observed quantities.

Examples of generative models are Gaussian/Linear Discriminant Analysis, Naïve Bayes and Hidden Markov Model.

Exponential Family of Distributions

A probability density function gives the likelihood (probability) of a random variable to have a certain value. A probability curve plotted for a specific range of values of random variable is called a distribution e.g. normal distribution has a (inverted) bell-shaped curve. An important family of distributions is the exponential family.

2.1 Density of an exponential family distribution

$$P(x|\eta) = h(x)\exp\{\eta(\theta)^T T(x) - A(\eta)\}. \quad (1)$$

Here, $\eta(\theta)$ represents the *natural parameter* (for most of this discussion we will refer to this parameter simply as η), $T(x)$ is the *sufficient statistic*, $h(x)$ is a *normalizing constant* (which can be thought of as a regularizer), and $A(\eta)$ is the *log partition function*.

The distributions belonging to exponential family are Bernoulli, normal/Gaussian, negative exponential, poisson, gamma, categorical, etc.

2.2 Representing the Bernoulli distribution in the exponential family form

For a Bernoulli distribution, with $x \in \{0, 1\}$ representing either success (1) or failure (0) of a trial and μ representing the probability of a success, $0 \leq \mu \leq 1$, we have,

$$\begin{aligned} P(x|\mu) &= \mu^x(1-\mu)^{(1-x)} \\ &= \exp\{\log(\mu^x(1-\mu)^{(1-x)})\} \\ &= \exp\{x\log\mu) + (1-x)\log(1-\mu)\} \\ &= \exp\left\{(\log\frac{\mu}{1-\mu})x + \log(1-\mu)\right\}. \end{aligned}$$

Comparing the final expression with equation 1, we have,

$$\begin{aligned} \eta &= \log\frac{\mu}{1-\mu} \\ T(x) &= x \\ h(x) &= 1 \\ A(\eta) &= -\log(1-\mu) = \log(1+e^{\eta}), \end{aligned}$$

where the last expression for $A(\eta)$ can be obtained by using the expression for μ (the logistic function) derived below.

Gaussian	$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\|\, x-\mu\,\|^2/(2\sigma^2)}$	$x \in \mathbb{R}$
Bernoulli	$p(x) = \alpha^x\,(1-\alpha)^{1-x}$	$x \in \{0,1\}$
Binomial	$p(x) = \binom{n}{x}\,\alpha^x\,(1-\alpha)^{n-x}$	$x \in \{0,1,2,\ldots,n\}$
Multinomial	$p(x) = \frac{n!}{x_1!x_2!\ldots x_n!}\,\prod_{i=1}^{n}\alpha_i^{x_i}$	$x_i \in \{0,1,2,\ldots,n\}\,,\ \sum_i x_i = n$
Exponential	$p(x) = \lambda\, e^{-\lambda x}$	$x \in \mathbb{R}^+$
Poisson	$p(x) = \frac{e^{-\lambda}}{x!}\,\lambda^x$	$x \in \{0,1,2,\ldots\}$
Dirichlet	$p(x) = \frac{\Gamma\left(\sum_i \alpha_i\right)}{\prod_i \Gamma(\alpha_i)}\,\prod_i x_i^{\alpha_i - 1}$	$x_i \in [0,1]\,,\ \sum_i x_i = 1$

The most important distribution of exponential family is the normal/Gaussian distribution. The probability density function of a normal distribution is:

$$P(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where, μ is the mean of distribution, σ is standard deviation, σ^2 is the variance.

Appendix-2: Ensemble Learning (Combining Classifiers)

Bagging

Bagging (bootstrap aggregating) is a technique where N new datasets are created from the original dataset. These datasets are the same size as the original dataset. Each dataset is built by randomly selecting an example from the original "with replacement" i.e. an example can be selected more than once. Then on each of these datasets we generate decision trees (in parallel). For a given test observation, we record the class predicted by each of these N trees, and take a majority vote.

Boosting

Whereas in bagging individual models are built separately, in boosting each new model is influenced by the performance of those built previously.

A popular boosting algorithm is AdaBoost which begins by assigning equal weights to all the instances in the training data. It then calls the learning algorithm to form a classifier for this data and re-weights each instance according to the classifiers output. The weight of correctly classified instance is decreased and that of misclassified is increased. In next iteration, the classifier focuses on correct classification of high weight instances. The amount of weight increase or decrease in AdaBoost is –log(e/1-e).

Appendix-3: Model Selection and Feature Selection

Model selection

Given a set of models, choose the model that is expected to give the best results.

- Chosing among different learning algorithms e.g. choosing kNN over other classifiction algorithms.
- Choosing parameters in same learning model e.g. choosing value of k in kNN.

Feature Selection

Selecting a useful subset from all the features.

Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size

Note: Feature Selection is different from Feature Extraction. The latter transforms original features to get a small set of new features

How?
Remove a binary feature if nearly all of it values are same.
Use some criteria to rank features and keep top ranked features.
Wrapper Methods: requires repeated runs of the learning algorithm with different set of features.