ACDS Lecture Series

Lecture - 14

CSIR

Pattern Recognition and Feature Selection

G. N. Sastry and Team

ADVANCED COMPUTATION AND DATA SCIENCES (ACDS) DIVISION

CSIR-North East Institute of Science and Technology, Jorhat, Assam, India

14. Pattern Recognition and Feature Selection

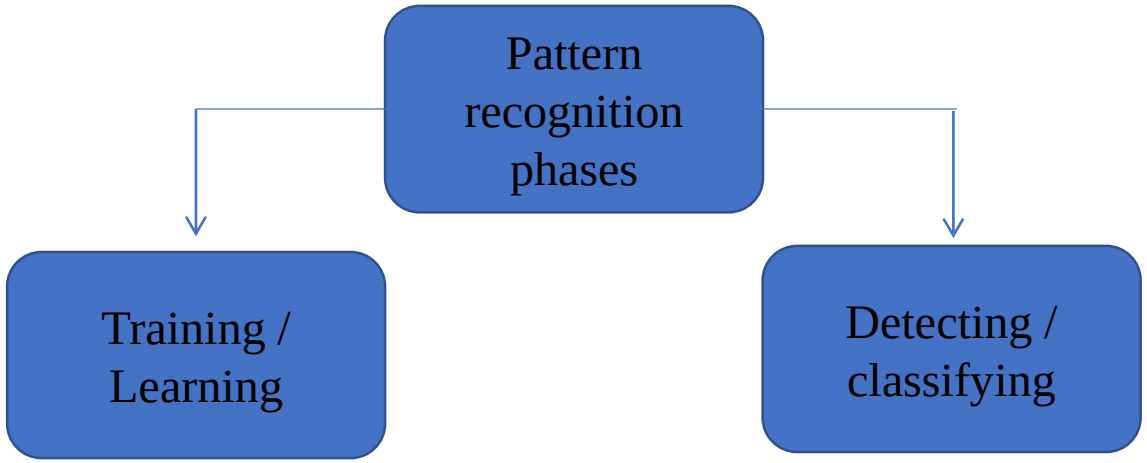
14.1 Introduction

Pattern is an abstraction represented by a set of measurements describing the physical object

Pattern recognition is the study of how machines can -

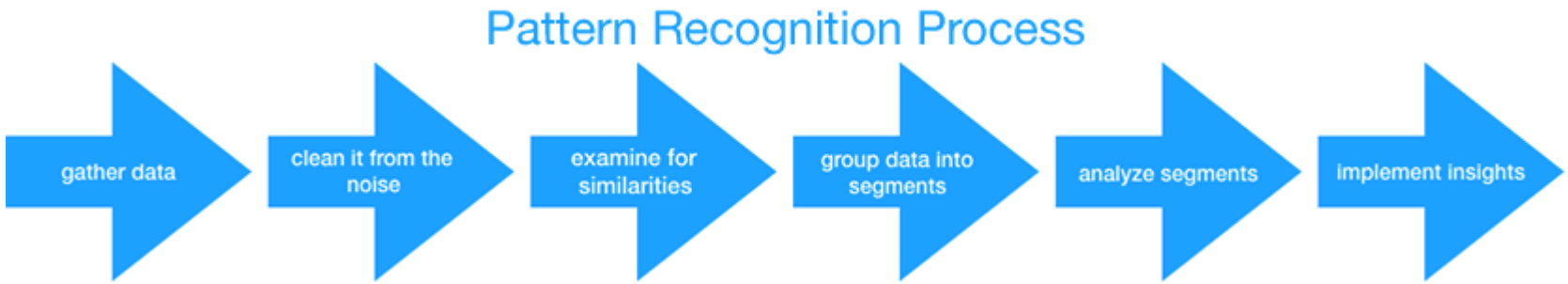
- ❑ Observe the environment
- ❑ Distinguish pattern of interest
- ❑ Make reasonable decisions over categories of patterns

Pattern class is set of patterns sharing similar properties but not identical objects

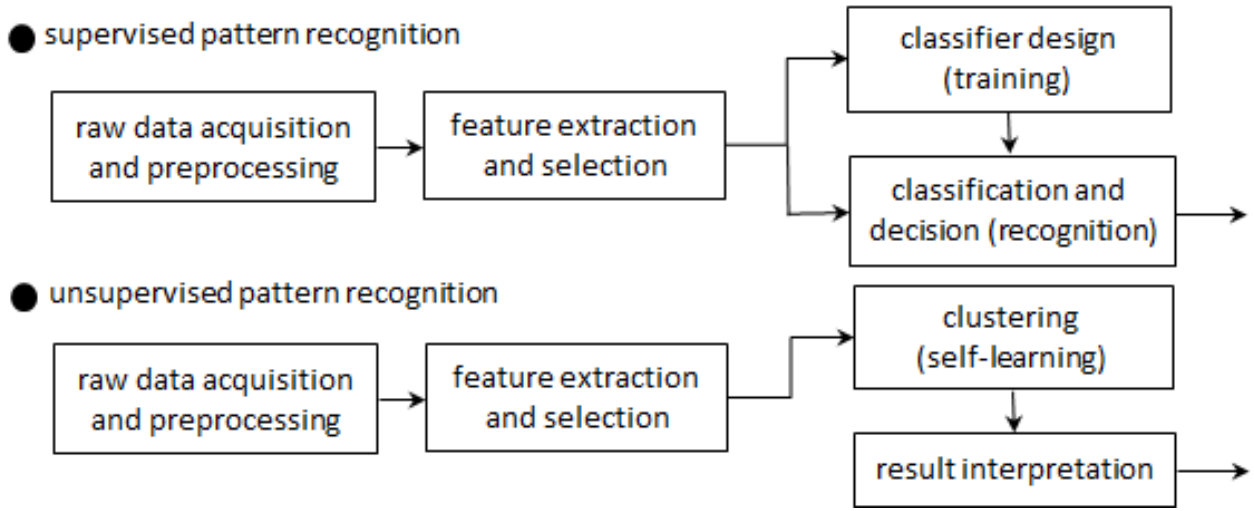


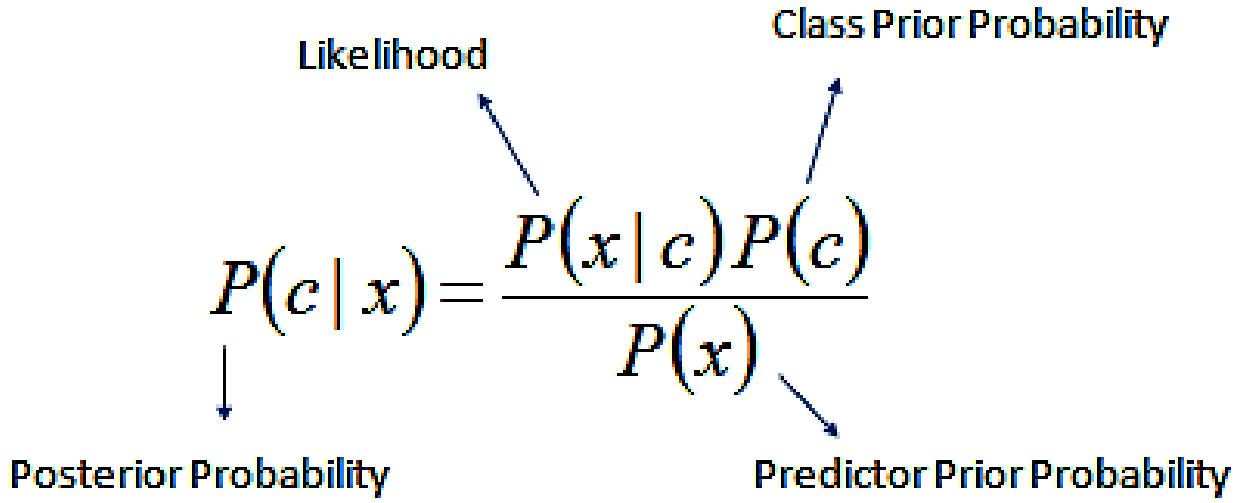
14.2 Supervised Pattern Detection

A set of labelled data is used to train the detection model for classification of the patterns.



Pattern recognition approaches





$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Naïve Bayes' Classifier

- Probabilistic Classifier based on Baye's theorem with strong (Naïve) independence assumptions among the features.

Why Naïve?

- It assumes the features that go into the model is independent of each other.
- That is changing the value of one feature, does not directly influence any other features.
- Because of independence nature it makes assumptions that may or may not turn out to be correct.

Multivariant Bayesian Classifier

- Compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, An|C)\, P(C)}{P(A_1, A_2, \ldots, An)}$$

- Choose value of C that maximizes $P(C \mid A_1, A_2, \ldots, A_n)$
- Equivalent to choosing value of C that maximizes $P(A_1, A_2, \ldots, A_n|C)\, P(C)$

Note: Other names "Simple Baye's" and "Independence Baye's"

Variants

Based on the probability distribution adopted

Bernoulli: Assumes that all features are binary such that they take only two values.

Multinomial: Assumes that features element represents frequency i.e. the number of times it appears.

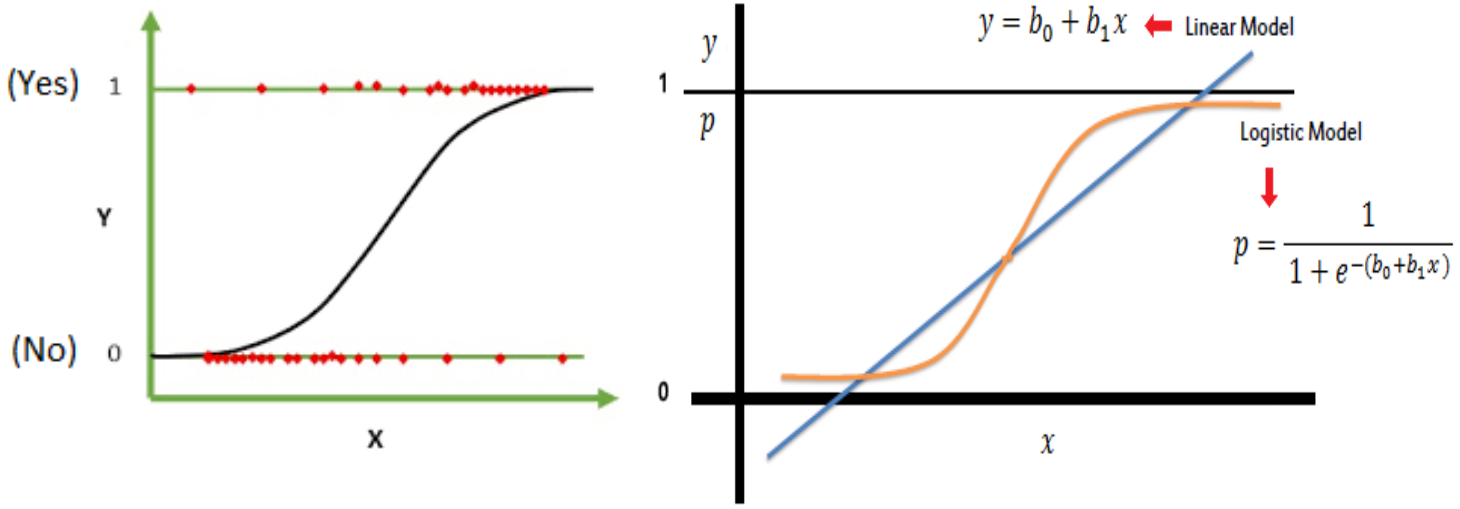
Gaussian: Based on a continuous distribution and it's suitable for more generic classification tasks.

14.2.2 Linear Classifiers

Logistic Regression

- Used in Classification (binary or multiclass) problem where y is a discrete value.
- Set threshold based on which classifier classifies.

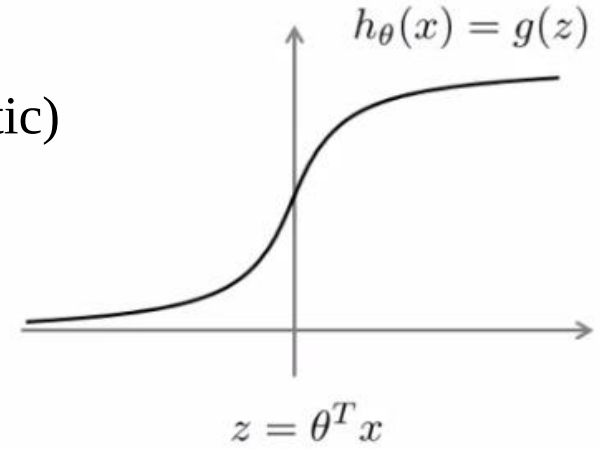
Size of Tumor	Malignant
2	N
1	N
1	N
8	Y
…	…
…	…



14.2.2 Linear Classifiers

Hypothesis

- Regression hypothesis can give values large than 1 or less than 0
- Logistic regression generates a value always between 0 or 1 (probabilistic)



- In linear regression we did $h_\theta(x) = (\theta^T x)$
- For classification hypothesis representation we do $h_\theta(x) = g((\theta^T x))$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

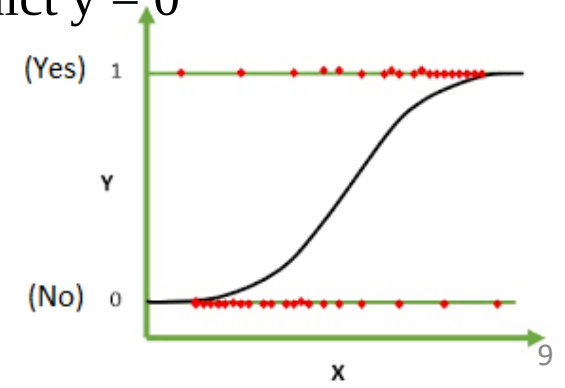
- If we combine these equations we can write out the hypothesis as $g(z) = (z) = 1/(1 + e^{-z})$

Where the **sigmoid** or **logistic function**

- When the probability of $y > 0.5$ then we can predict $y = 1$ Else we predict $y = 0$

$g(z) >= 0.5$ when $z (=\theta^T x) >= 0$

$g(z) < 0.5$ when $z (=\theta^T x) < 0$



- Multivariate logistic Regression $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
- Polynomial logistic Regression $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2 + \theta_4 x_2^2)$

Cost Function

- Linear regression uses the following function to determine θ

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

- Instead of writing the squared error term, we can write
  - If we define "cost()" as $\text{cost}(h_\theta(x^{(i)}), y) = ½ (h_\theta(x^{(i)}) - y^{(i)})^2$
  - We can **redefine J(θ) as** $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$

Optimization

Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

Minimize cost function $J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$

Gradient Descent

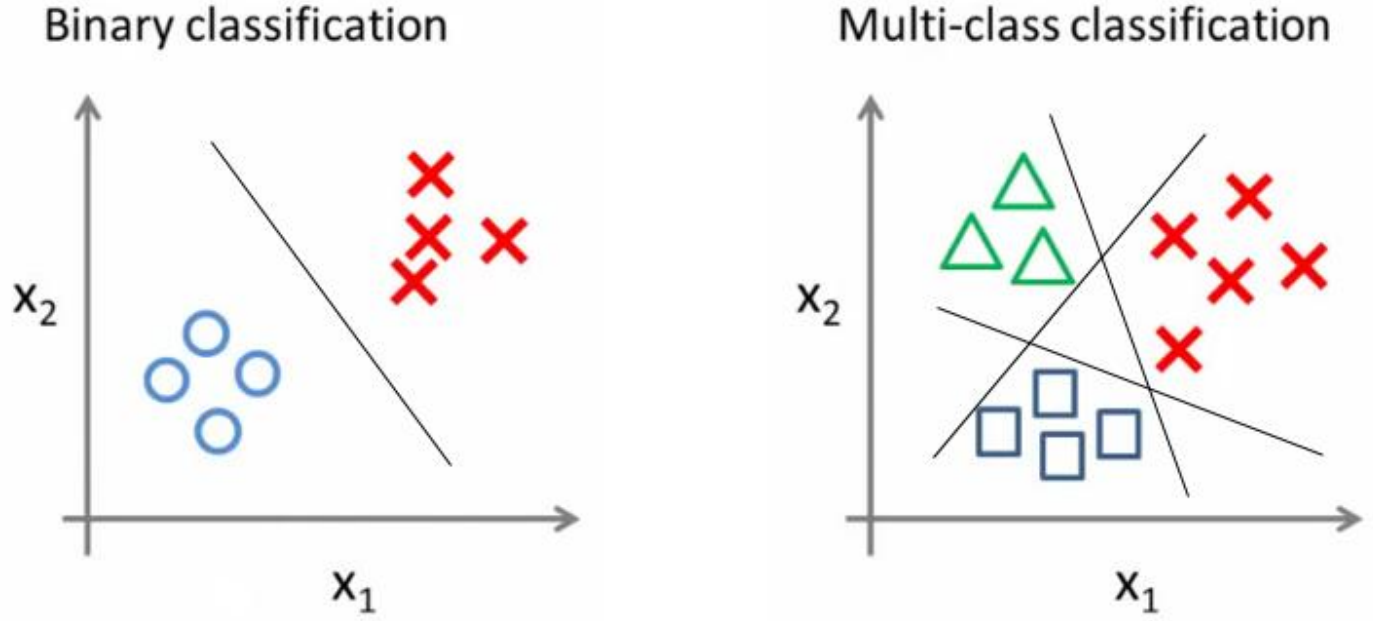
Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

6.3.5 Multiclass Classification



- Split the training set into three separate binary classification problems.
  - Triangle (1) vs crosses and squares (0) $h_\theta^1(x)$ $p(y=1 \mid x_1; \theta)$
  - Crosses (1) vs triangle and square (0) $h_\theta^2(x)$ $p(y=1 \mid x_2; \theta)$
  - Square (1) vs crosses and square (0) $h_\theta^3(x)$ $p(y=1 \mid x_3; \theta)$

14.2.3 Non Linear Classifiers

Both kNN and kernel methods use the concept of distance/similarity to training instances
Nonlinear classifiers that make predictions based on features instead of distance/similarity:

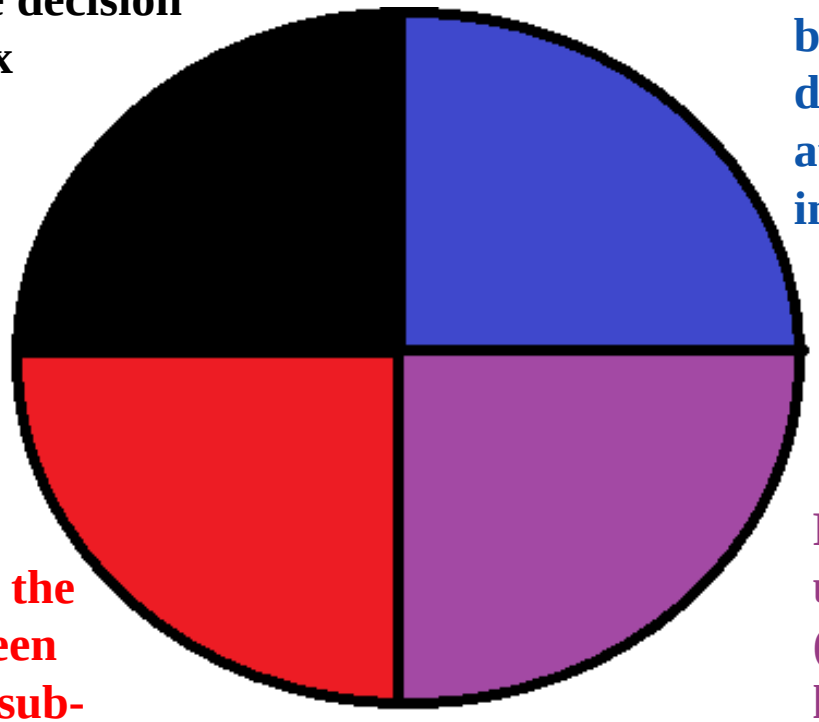
- Decision tree
- Multilayer perceptron
  - A basic type of neural network

What is Decision Tree ???

- A decision tree is a graphical representation of all the possible solutions to a decision based on certain condition.
- A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

14.2.3 Non Linear Classifiers

The measurement of impurity (or purity used in building the decision tree in CART is Gini Index



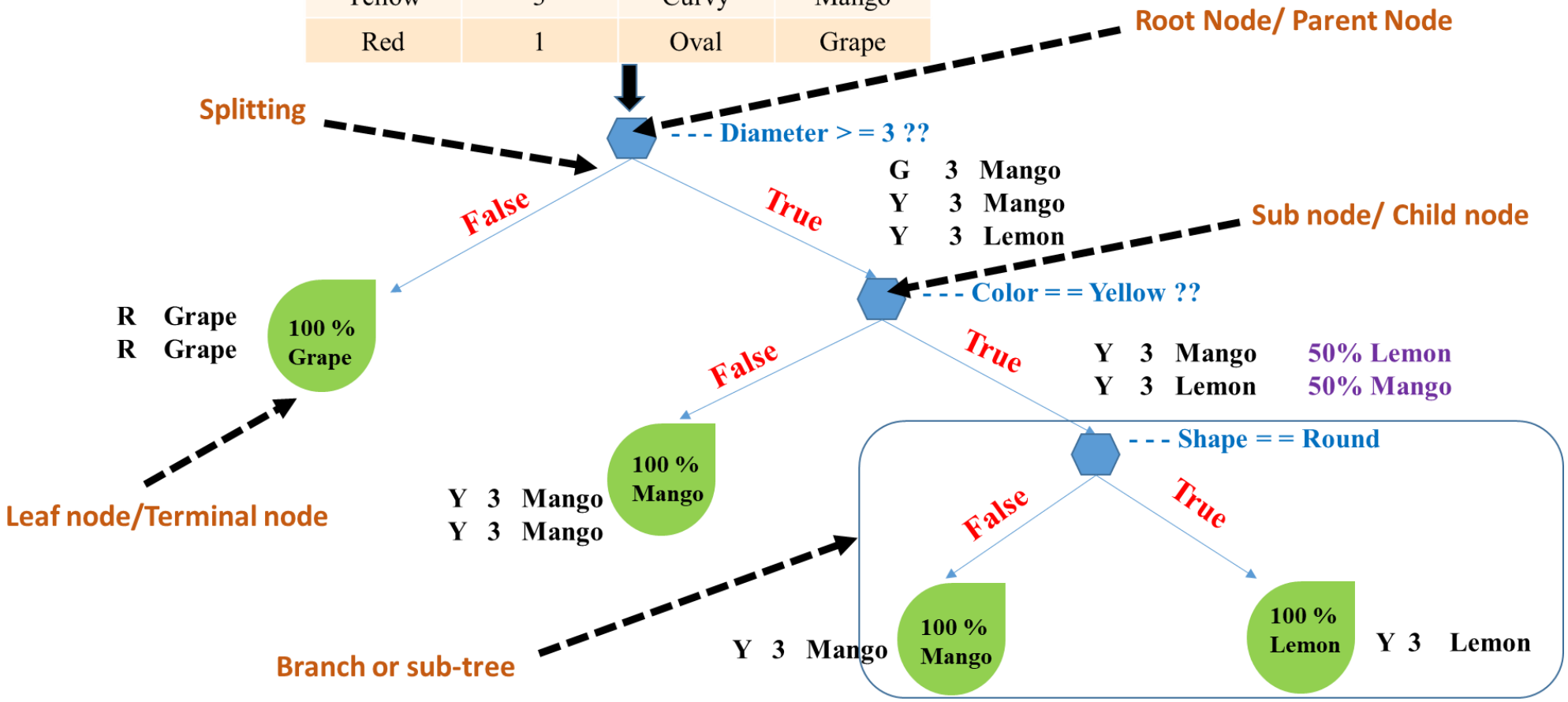
The information gain is the decrease in entropy after a dataset is split on the basis of an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain

It is an algorithm to find out the statistical significance between the differences between sub-nodes and parents node

Reduction in Variance is an algorithm used for continuous target variables (regression problems). The split with lower variance is selected as the criteria to split the population

14.2.3 Non Linear Classifiers

Color	Diameter	Shape	Label
Green	3	Curvy	Mango
Yellow	3	Round	Lemon
Red	1	Oval	Grape
Yellow	3	Curvy	Mango
Red	1	Oval	Grape



How does a tree decide where to split?

The four most commonly used algorithms in decision tree are:

1. Information Gain

Entropy(S) = -P (Yes) log2 P(Yes) – P (No) log2 P (No)

Information Gain = Entropy(S) – [(Weighted Avg) x Entropy (each feature)

2. Gini Index

Gini (D) = 1 - (probability of YES)^2 - (Probability of NO)^2

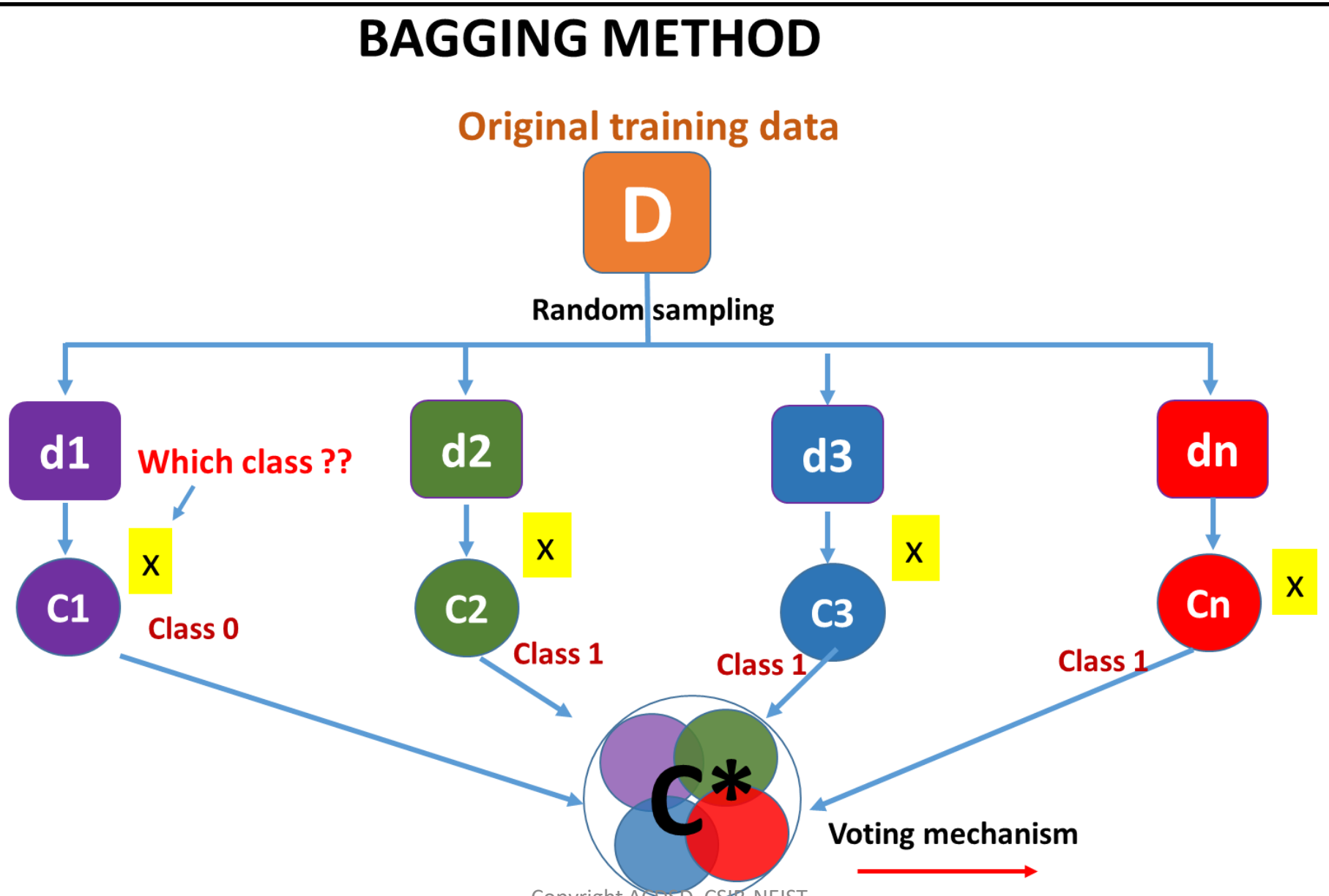
3. Reduction in variance

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

4. Chi-Square

14.2.3 Non Linear Classifiers

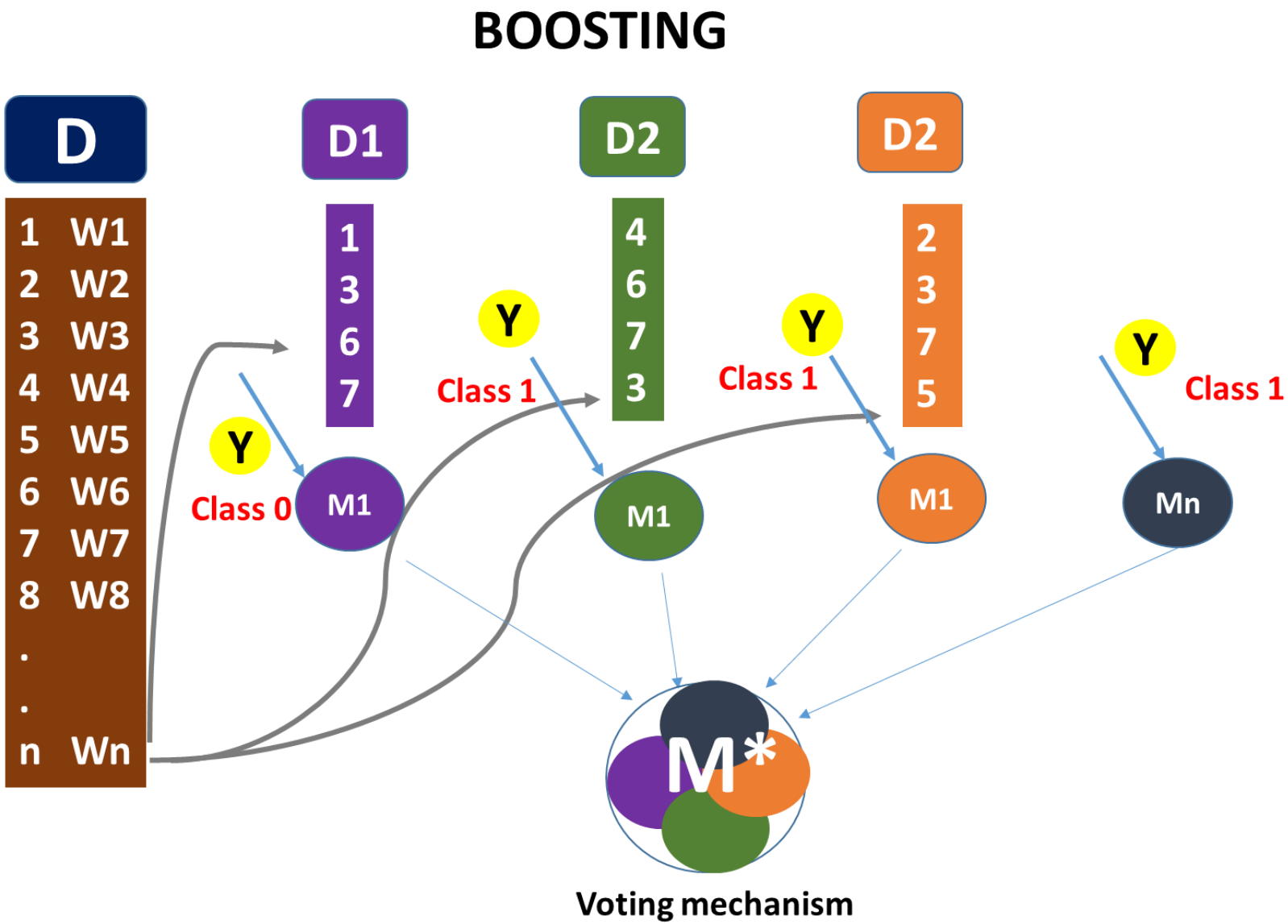
- **Bagging** is used when the goal is to reduce the variance of a decision tree classifier. Here the objective is to create several subsets of data from training sample chosen randomly with replacement. Each collection of subset data is used to train their decision trees. As a result, we get an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree classifier.
- **Bagging Steps:**
- Suppose there are N observations and M features in training data set. A sample from training data set is taken randomly with replacement.
- A subset of M features are selected randomly and whichever feature gives the best split is used to split the node iteratively.
- The tree is grown to the largest.
- Above steps are repeated n times and prediction is given based on the aggregation of predictions from n number of trees.
- **Advantages:**
- Reduces over-fitting of the model.
- Handles higher dimensionality data very well.
- Maintains accuracy for missing data.



14.2.3 Non Linear Classifiers

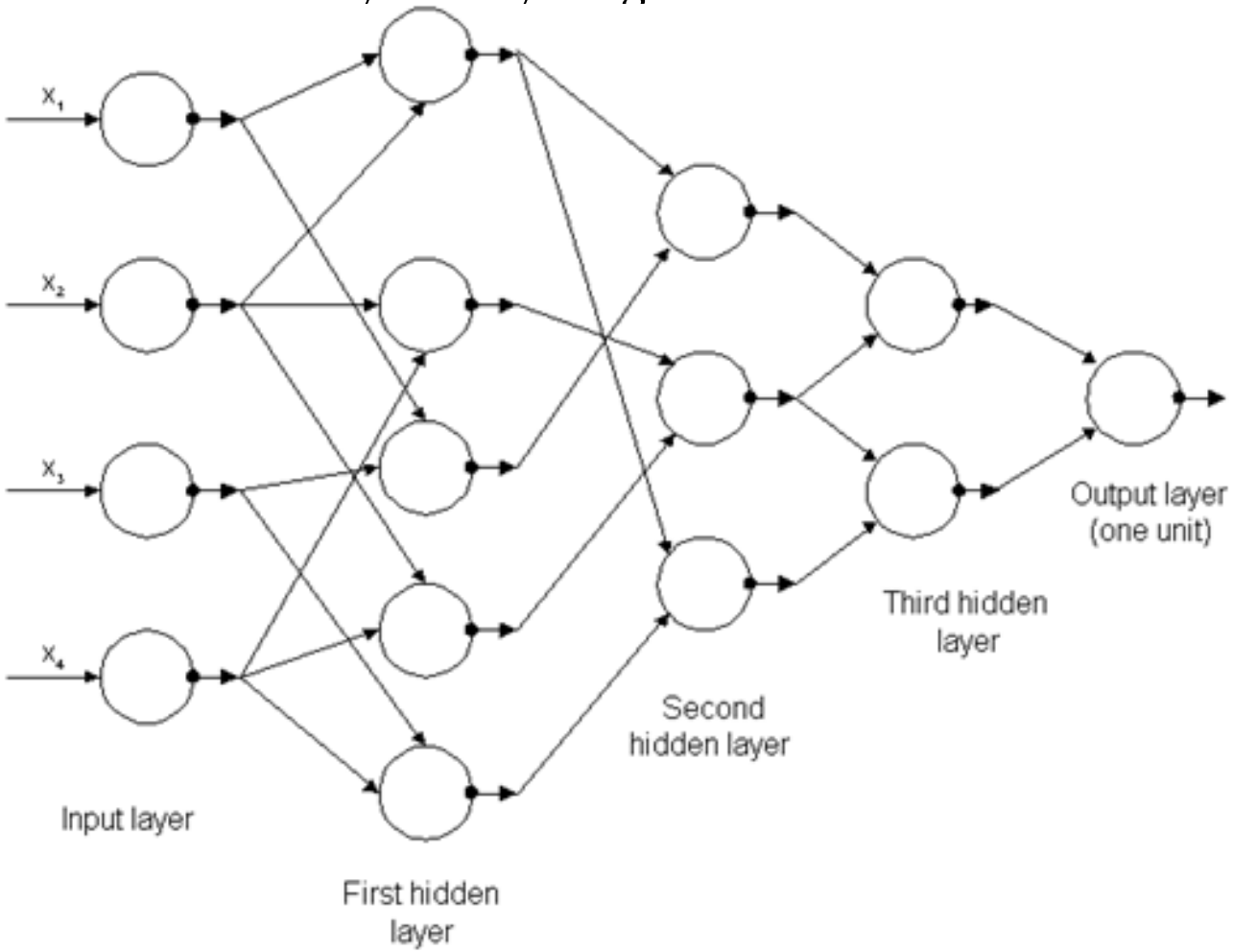
- **Boosting** is used to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analysing data for errors. Consecutive trees (random sample) are fit and at every step, the goal is to improve the accuracy from the prior tree. When an input is misclassified by a hypothesis, its weight is increased so that next hypothesis is more likely to classify it correctly. This process converts weak learners into better performing model.
- **Boosting Steps:**
- Draw a random subset of training samples d1 without replacement from the training set D to train a weak learner C1
- Draw second random training subset d2 without replacement from the training set and add 50 percent of the samples that were previously falsely classified/misclassified to train a weak learner C2
- Find the training samples d3 in the training set D on which C1 and C2 disagree to train a third weak learner C3
- Combine all the weak learners via majority voting.
- **Advantages:**
- Supports different loss function (we have used ‘binary:logistic’ for this example).
- Works well with interactions.

14.2.3 Non Linear Classifiers



Multi Layer Perceptron

- First Generation MLP introduced by Alexey Grigorevich Ivakhnenko in 1965.



14.2.3 Non Linear Classifiers

Working:

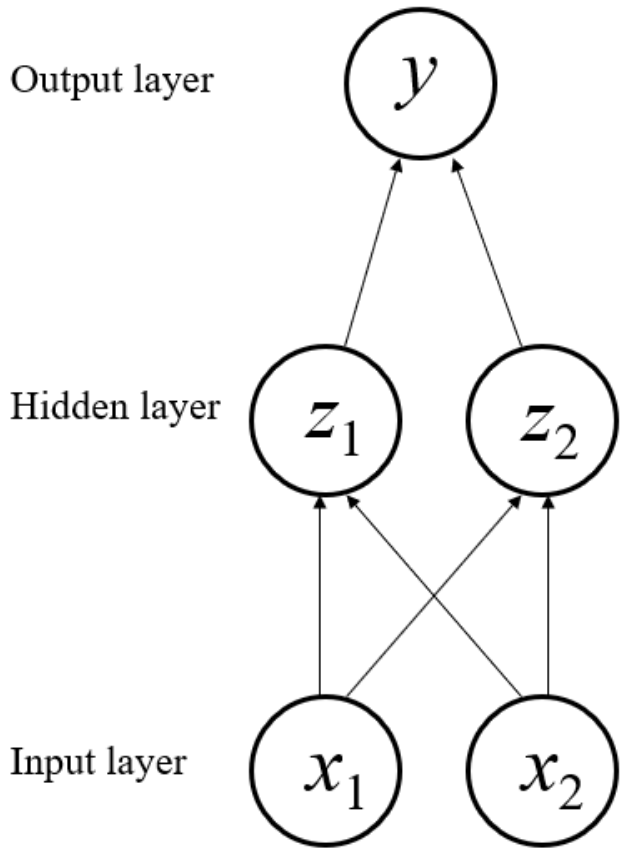
- Same as perceptron but with multiple layers.

Advantages:

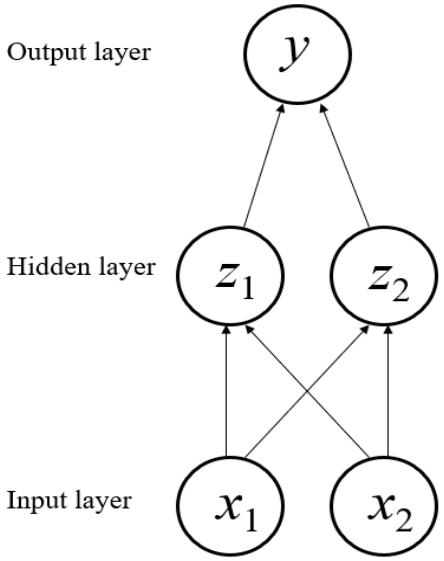
- MLP can represent non-linear Boolean functions.

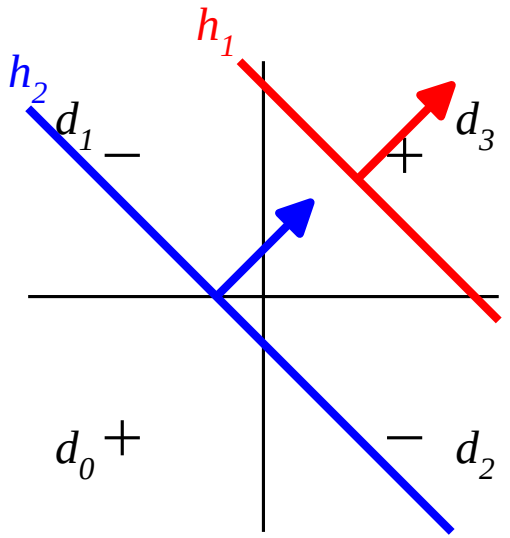
Disadvantages:

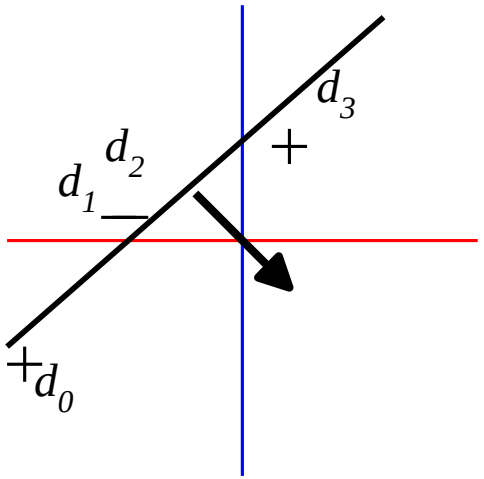
- Can not realize any continuous function.



Capabilities:

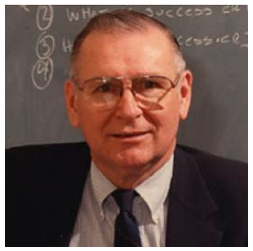






Single layer	Multi layer
Can not represent non linear function	Can represent any linear and non linear functions
Insufficiently expressive; they are non-linear i.e. can not represent non-linear and continuous functions	Sufficiently expressive; can represent any Boolean and continuous functions using appropriate activation functions
Can not deal with generalized data sets	Can be trained to deal with generalized data sets, i.e. via error back-propagation

LefTeri H. Tsoukalas & Robert E. Uhrig [1997]



Data processing system consisting of a large number of simple, highly interconnected processing elements (neurons) in an architecture inspired by the structure of the cerebral cortex of the brain.

Simon S. Haykin [1994]

A neural network is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:

- Knowledge is acquired by the network through a learning process.
- Interneuron connection strengths known as synaptic weights are used to store the knowledge.

Applications

When To Use

- Problems where we can't formulate an algorithmic solution.
- Problems where we can get lots of examples of the behavior we require.
- Input is high-dimensional discrete or real-valued (e.g., raw sensor input)
- Noise in training examples.
- Problems where we need to pick out the structure from existing data.
- If long training time is acceptable.

Problems where applied

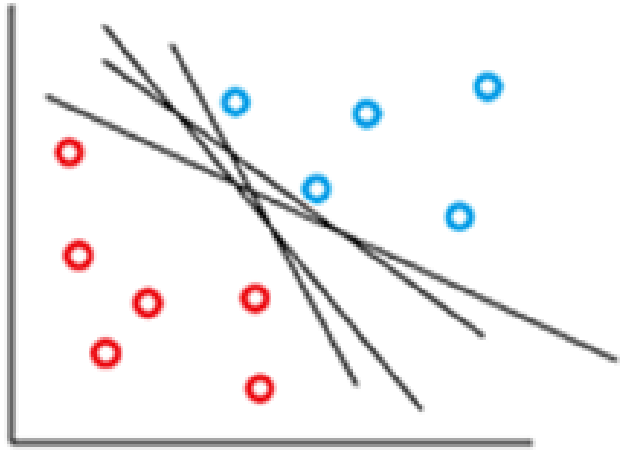
- Applied to following types of problems from real world –
  - Regression
  - Classification
  - Clustering
  - Association and Pattern Recognition

Applications

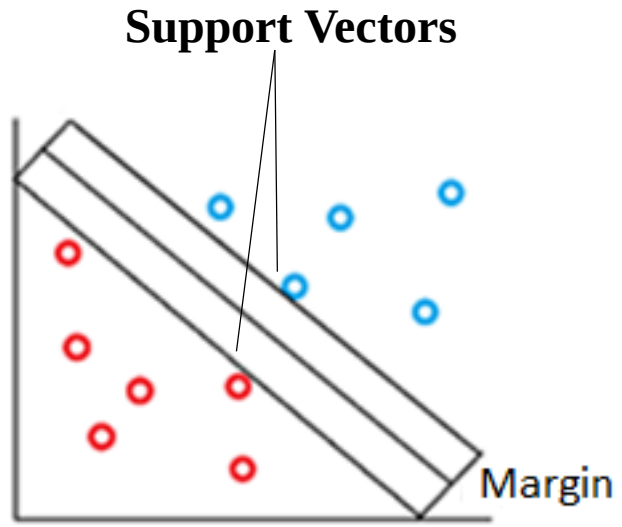
- Regression:
  - Forecasting (prediction on base of past history)
  - Forecasting e.g., predicting behavior of stock market
- Classification:
  - Image recognition
  - Speech recognition
  - Diagnostic
  - Fraud detection
  - Face recognition
- Clustering:
  - Clients profiles
  - Disease subtypes
- Associations and Pattern recognition
  - Network attacks
  - Breast cancer
  - Handwriting recognition
  - Retrieve an image from corrupted one

14.2.5 Support Vector Machine

Supervised data



Decision Boundary



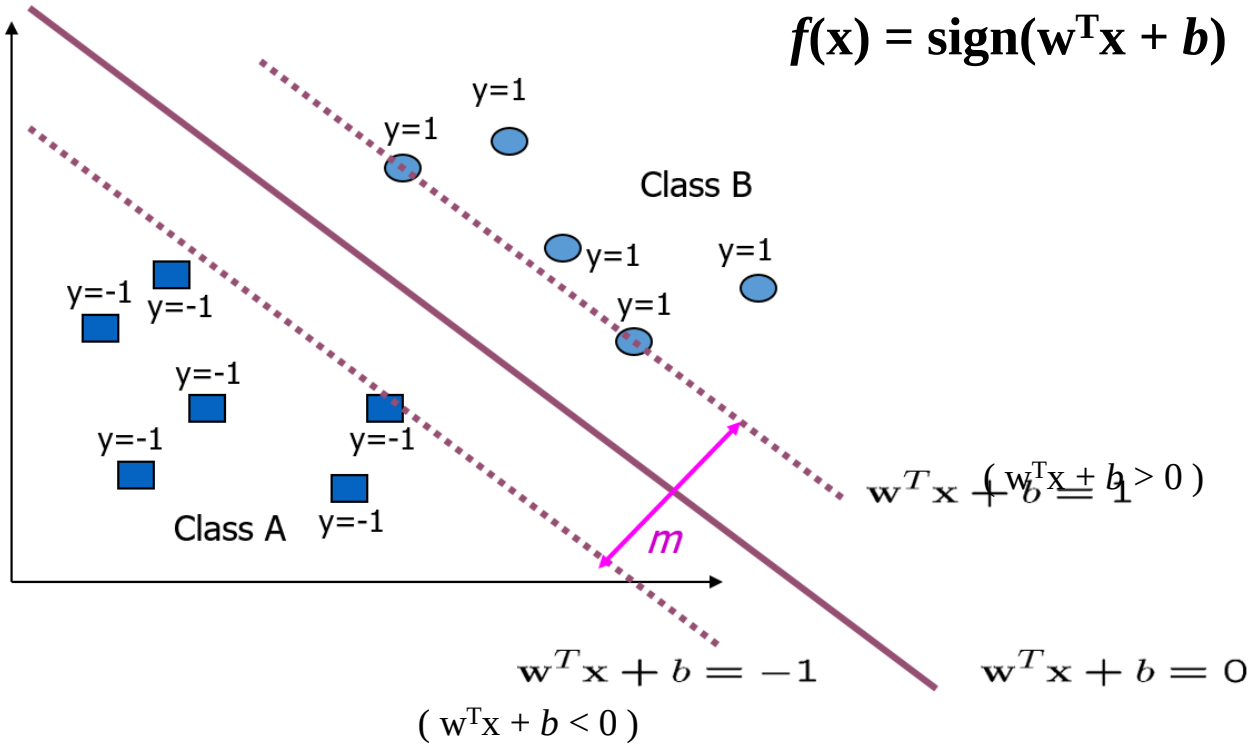
Optimal Decision Boundary (Large Margin)

Decision Boundary

- Let $\{x_1, ..., x_n\}$ be our data set and let y_i $\{1,-1\}$ be the class label of x_i

- For y_i=1 $w^T x_i + b \geq 1$
- For y_i=-1 $w^T x_i + b \leq -1$

$$y_i \cdot (w^T x_i + b) \geq 1, \forall (x_i, y_i)$$



Margin

- The decision boundary should be as far away from the data of both classes as possible.
- Goal is to maximize the margin (*m*) $m = \frac{2}{||\mathbf{w}||}$

Optimization

- The decision boundary should classify all points correctly for y_i=1

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- The decision boundary can be found by solving the following constrained optimization problem

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

- This is a constrained optimization problem.
- Solving it requires to use Lagrange multipliers.

Optimization

- **The optimization problem**

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \leq 0 \qquad \text{for } i = 1,\ldots,n$$

- **The Lagrangian is**

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

Note: $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

Note: Lagrange multipliers are used in multivariable calculus to find maxima and minima of a function subject to constraints (like "find the highest elevation along the given path" or "minimize the cost of materials for a box enclosing a given volume").

Optimization

Gradient with respect to *w* and *b*

- Setting the gradient of L w.r.t. **w** and b to zero, we have

$$L = \frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(w^T x_i + b\right)\right)$$

$$= \frac{1}{2} \sum_{k=1}^{m} w^k w^k + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\sum_{k=1}^{m} w^k x_i^k + b\right)\right)$$

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i (-y_i) \mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\begin{cases} \frac{\partial L}{\partial w^k} = 0, \forall k \\ \frac{\partial L}{\partial b} = 0 \end{cases}$$

- If we substitute $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ to Lagrange Multipliers, we have

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + b\right)\right)$$

$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b \sum_{i=1}^{n} \alpha_i y_i$$

$$= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i$$

Therefore, this is a function of α_i only Since $\sum_{i=1}^{n} \alpha_i y_i = 0$

Optimization

Dual Problem

- The new objective function is in terms of α_i only.
- It is known as the dual problem: if we know w, we know all α_i if we know all α_i we know w.
- The original problem is known as the primal problem.

- The objective function of the dual problem needs to be maximized.

- The dual problem is therefore:

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- This is a **quadratic programming (QP) problem**
  - A global maximum of α_i can always be found
- **w** can be recovered by $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$

Optimization

Quadratic Programming Problem

- Many approaches have been proposed.
- Most are "interior-point" methods.
  - Start with an initial solution that can violate the constraints
  - Improve this solution by optimizing the objective function

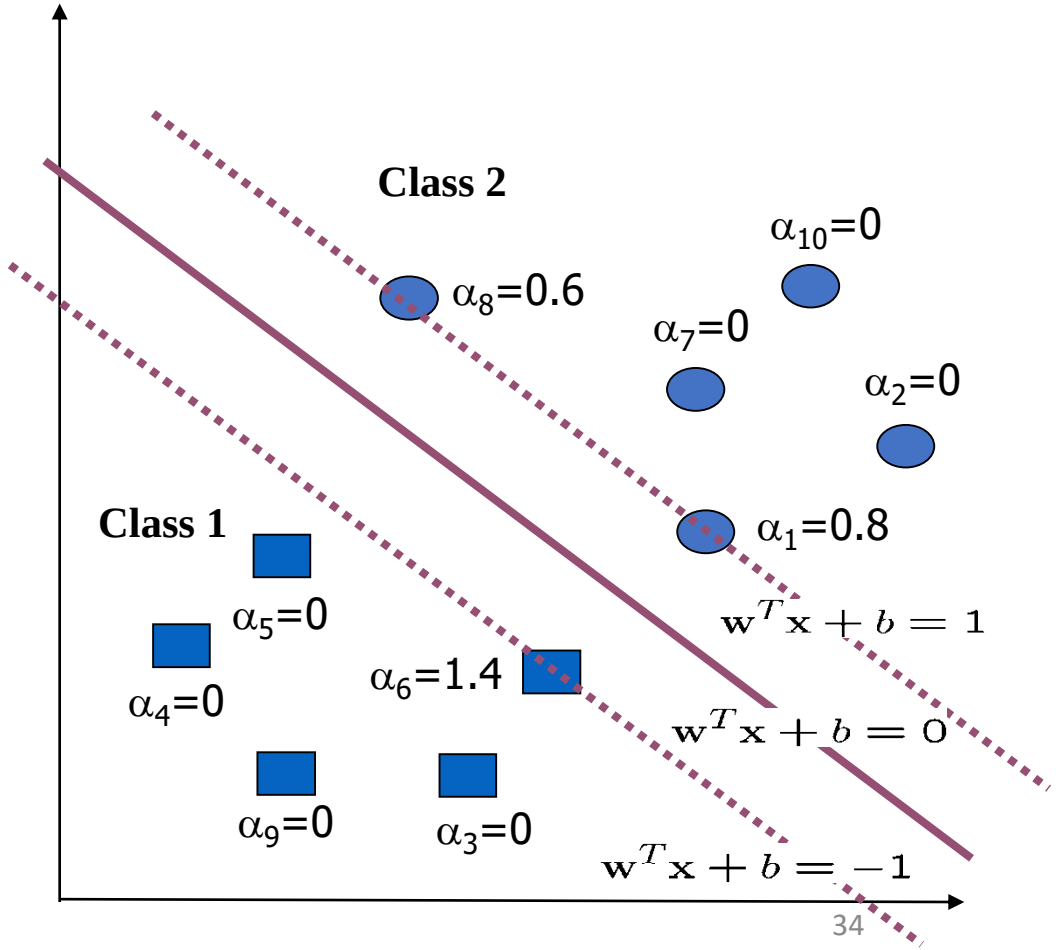
- For SVM, sequential minimal optimization (SMO) seems to be the most popular.
  - A QP with two variables is trivial to solve
  - Each iteration of SMO picks a pair of (α_i, α_j) and solve the QP with these two variables; repeat until convergence

- In practice, we can just regard the QP solver as a "black-box" without bothering how it works.

Characteristics of the Solution

- Many of the α_i are zero
- So **w** is a linear combination of a small number of data points

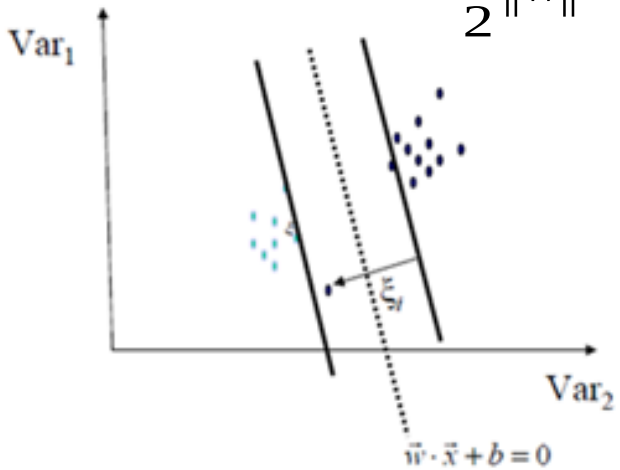
- $\mathbf{x}_i$ with non-zero α_i are called support vectors (SV)
  - The decision boundary is determined only by the SV
  - Let t_j (j=1, ..., s) be the indices of the s support vectors
  - We can write $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- For testing with a new data **z**
  - Compute $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$
  - Classify **z** as class 1 if the sum is positive else class 2



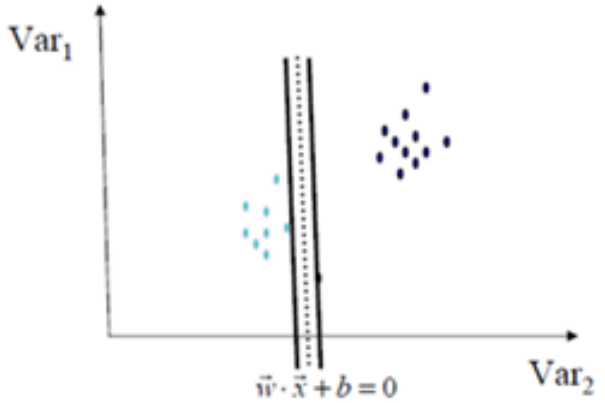
Non Linearly Separable Problems

Soft vs. Hard Margin SVM

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$



Soft Margin SVM



Hard Margin SVM

- The algorithm try to keep ξ null, maximizing the margin.
- The algorithm does not minimize the number of error.
- Instead, it minimizes the sum of distances from the hyperplane.
- When C increases the number of errors tend to lower.
- At limit C→infinite, the solution tend to that given by hard margin formulation, with 0 errors.

Non Linearly Separable Problems

Soft Margin Hyperplane

- The new conditions become

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

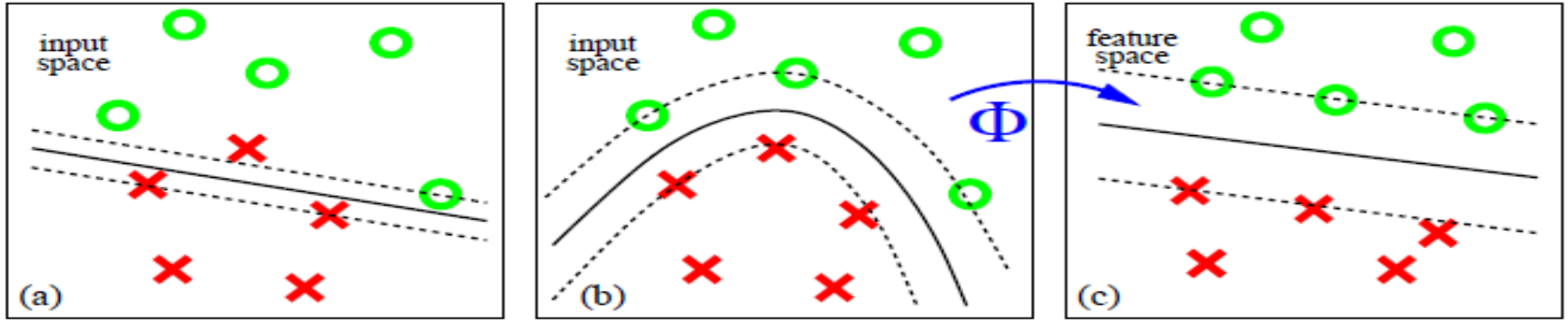
  - x_i are "slack variables" in optimization
  - Note that x_i=0 if there is no error for $\mathbf{x}_i$
  - x_i is an upper bound of the number of errors

- We want to minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$

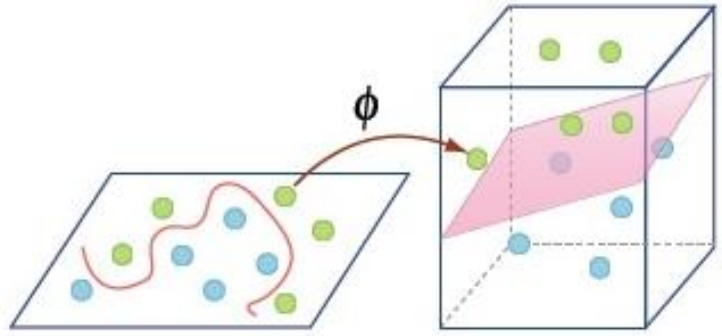
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

- C : tradeoff parameter between error and margin

Transformation



- Feature space is of higher dimension than the input space in practice
- Computation in the feature space can be costly because it is high dimensional.
- The kernel trick comes to rescue.



Input space (Hyperplane in R^2 is a line)

Feature space (Hyperplane in R^3 is a plane)

SVM Parameter Tuning

- **Kernel:** It specifies the kernel type to be used in the algorithm.
  - It can be ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’
  - The default value is ‘rbf’
- **Degree:** The degree of the polynomial kernel function (‘poly’)

- **C:** It is the regularization parameter of the error term.
- **Gamma:** It is the kernel coefficient for ‘rbf’, ‘poly’, and ‘sigmoid’.

https://cs.stanford.edu/people/karpathy/svmjs/demo/

14.2.6 Rule Based Classifiers

Classify records by using a collection of "*if .. then*" rules

Rule: (Condition) -> y where, *Condition:* conjugation of attributes and *Y*: a class label

Example:

- IF age = youth AND student = yes THEN buys_computer = yes
- Rule antecedent/precondition vs. rule consequent

Assessment of a rule: coverage and accuracy

- N_{covers} =no. of tuples covered by Rule
- $N_{correct}$ = no. of tuples correctly classified by the rule
- Coverage (rule) = N_{covers} / |D| # D is training data set
- Accuracy (rule) = $N_{correct}$ / N_{covers}

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Coverage = 2/14 = 14.28%

Accuracy = 2/2 = 100%

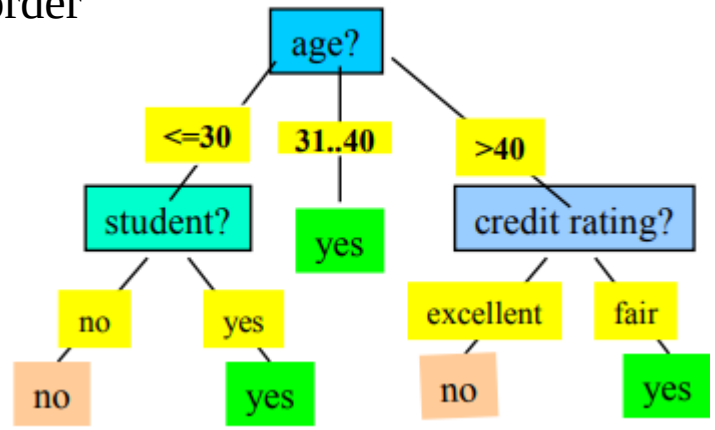
14.2.6 Rule Based Classifiers

If – Then Rules

- Rule Triggering
  - Input X satisfies a rule
- Several rules are triggered – Conflict Resolution
  - Size Ordering
    - Highest priority to toughest (rule antecedent size) rule
  - Rule Ordering
    - Rules are prioritized before-hand
    - Class based ordering
      - Rules for most prevalent class comes first or based on mis-classification cost / class
    - Rule-based ordering
      - Rule Quality based measures
      - Ordered list – Decision list – Must be processed strictly in order
- No rule is triggered – Default rule

Example: Rule extraction from the buys_computer decision-tree

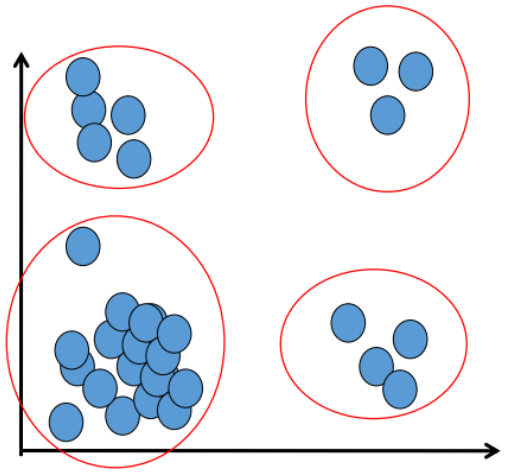
IF age = young AND student = no THEN buys_computer = no
IF age = young AND student = yes THEN buys_computer = yes
IF age = mid-age THEN buys_computer = yes
IF age = old AND credit_rating = excellent THEN buys_computer = yes
IF age = young AND credit_rating = fair THEN buys_computer = no



14.3 Unsupervised Pattern Detection

The goal of clustering is to

- group data points that are close (or similar) to each other
- identify such groupings (or clusters) in an unsupervised manner



- A cluster is represented by a single point, known as centroid (or cluster center) of the cluster.
- Centroid is computed as the mean of all data points in a cluster $C_j = \sum x_i$
- Cluster boundary is decided by the farthest data point in the cluster.

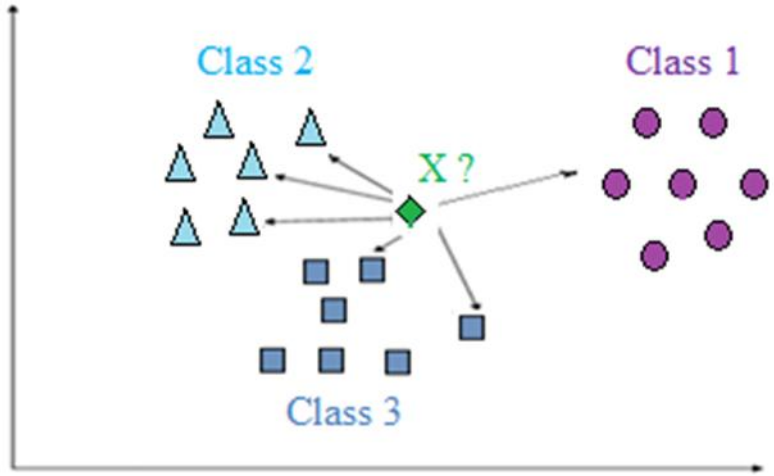
Types of Clustering

1. Exclusive Clustering: K-means
2. Overlapping Clustering: Fuzzy C-means
3. Hierarchical Clustering: Agglomerative clustering, divisive clustering
4. Probabilistic Clustering: Mixture of Gaussian models

- Clustering involves grouping data into categories based on some measure of similarity or distance.
- Thus clustering requires some methods for computing (dis)**similarity** or **distance** between each pair of observations.
- The result of this computation is known as a dissimilarity or **distance metrics**.

Similarity among points is defined using –

- Some inter-observation distance measures
- Correlation-based distance measures



What is a Self Organizing Map?

- Neural Net for unsupervised learning.
- Assuming that class membership is broadly defined by the input patterns sharing common features and that the network will be able to identify those features across the range of input patterns.
- It performs dimensionality reduction and clustering both.

Learning

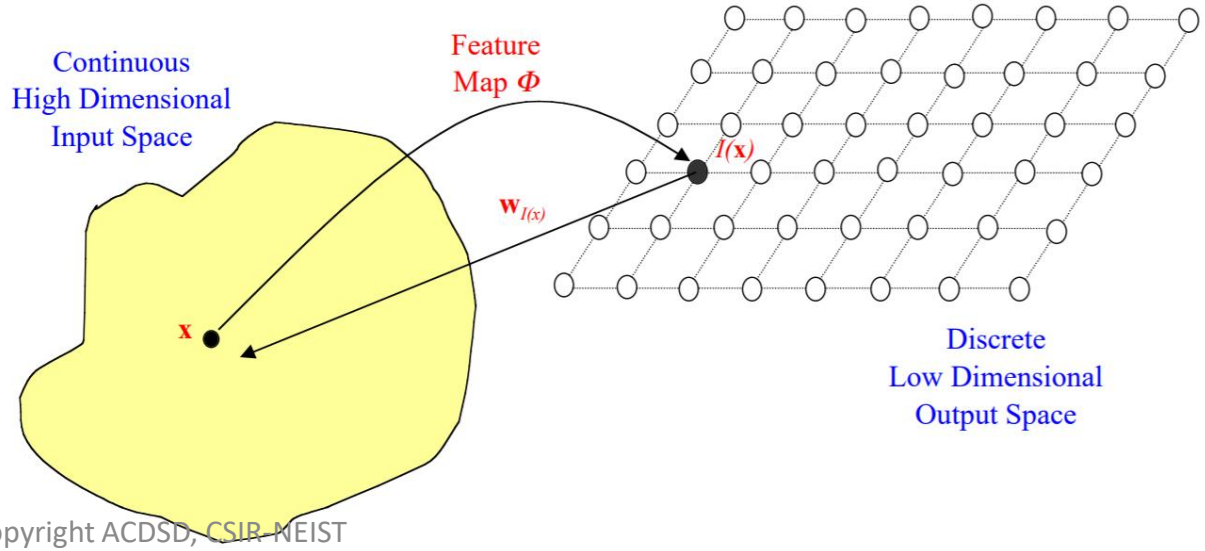
- Based on competitive learning, in which the output neurons compete amongst themselves to be activated with the result that only one is activated at any one time.
- This activated neuron is called a winner-takesall neuron or simply the winning neuron.

Why Self Organization?

- Competition implemented by having lateral inhibition connections (negative feedback paths) between neurons resulting the neurons are forced to organize themselves.

Architecture

- The goal is to transform an input pattern of arbitrary dimension into a 1 or 2 dimensional map.
- Therefore set up SOM by placing neurons at the nodes of a 1 or 2 or higher dimensional lattice.
- The neurons become selectively tuned to various input pattern class during competitive learning.
- The locations of the neurons so tuned (i.e. the winning neurons) become ordered and a meaningful coordinate system for the input features is created on the lattice.
- We can view this as a non-linear generalization of principal component analysis (PCA).



Components

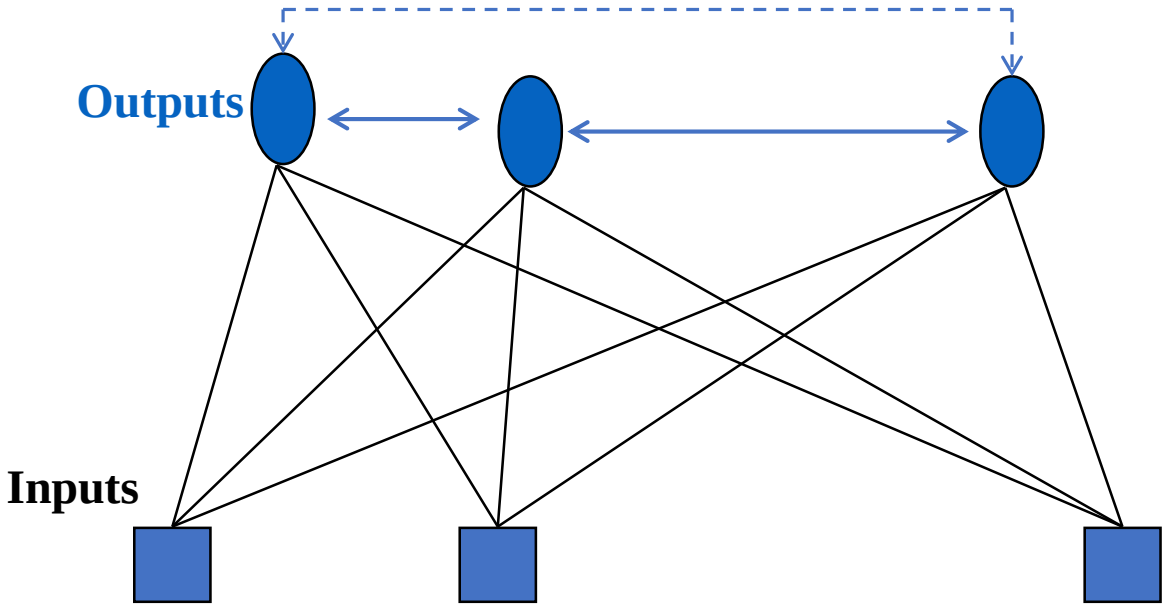
The self-organization process involves four major components:

- **Initialization:** All the connection weights are initialized with small random values.
- **Competition:** For each input pattern, the neurons compute their respective values of a discriminant function which provides the basis for competition. The particular neuron with the smallest value of the discriminant function is declared the winner.
- **Cooperation:** The winning neuron determines the spatial location of a topological neighbourhood of excited neurons, thereby providing the basis for cooperation among neighbouring neurons.
- **Adaptation:** The excited neurons decrease their individual values of the discriminant function in relation to the input pattern through suitable adjustment of the associated connection weights, such that the response of the winning neuron to the subsequent similar input pattern is enhanced.

14.3.3 Competitive Learning

- Finite resources: outputs ‘compete’ to see which will win via inhibitory connections between them
- Aim is to automatically discover statistically salient features of pattern vectors in training data set: feature detectors
- Can find clusters in training data pattern space which can be used to classify new patterns

Competitive network is similar to a single layer feed-forward network having feedback connection between the outputs. The connections between the outputs are inhibitory type, which is shown by dotted lines, which means the competitors never support themselves.



Concept of Competitive Learning

- Condition to be a winner for y_k

$$y_k = \begin{cases} 1 & if\ v_k > v_j\ for\ all\ j,\ j \neq k \\ 0 & otherwise \end{cases}$$

- Condition of the sum total of weight

$$\sum_k w_{kj} = 1 \quad for\ all\ k$$

- Change of weight for the winner

$$\Delta w_{kj} = \begin{cases} -\alpha(x_j - w_{kj}), & if\ neuron\ k\ wins \\ 0 & if\ neuron\ k\ losses \end{cases}$$

Here α is the learning rate.

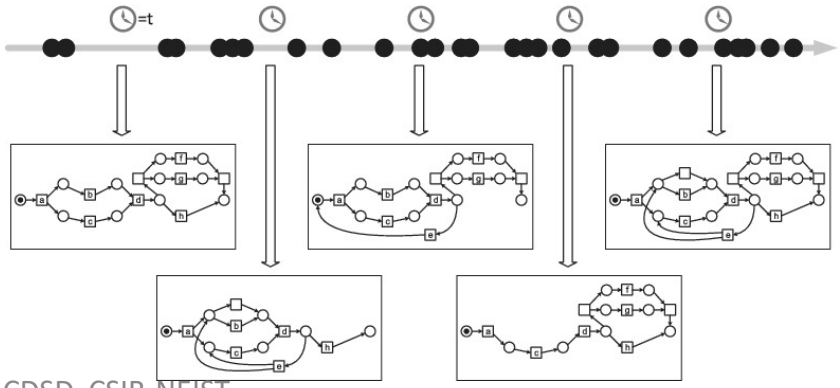
14.4 Sequential Pattern Recognition

- Sequence database – consists of sequences of ordered elements or events (with or without time)
- Sequential Pattern Mining is the mining of frequently occurring ordered events or subsequences as patterns
- Example:
  - Customer shopping sequences:
    - First buy computer, then CD-ROM, and then digital camera, within 3 months.
  - Web access patterns, Weather prediction
- Usually categorical or symbolic data
  - Numeric data analysis – Time Series Analysis

- A sequence database, S, is a set of tuples, , where SID is a sequence ID and s is a sequence
- A tuple is said to contain a sequence a, if a is a subsequence of s
- The support of a sequence α in a sequence database S is the number of tuples in the database containing α
- Given the minimum support threshold, a sequence a is frequent in sequence database S if support $_S$(a) >= min sup
- A frequent sequence is called a *sequential pattern*
- A sequential pattern with length l is called an *I-pattern*

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

- Full Set vs Closed Set
  - A sequential patterns s is closed if there exists no s' where s' is a proper super-sequence of s and s' has same support as s
    - Subsequences of a frequent sequence are also ferquent
    - Mining closed sequential patterns avoids generation of unnecessary sub-sequences
- **GSP (Generalized Sequential Patterns)**
  - Candidate generate and test approach on horizontal data format
- **SPADE (Sequential Pattern Mining on Vertical data format)**
  - Candidate generate and test approach on vertical data format
- **PrefixSpan (Prefix-Projected Sequential Pattern Growth)**
  - does not require candidate generation
- All approaches exploit Apriori property – every non empty subsequence of a sequential pattern is a sequential pattern



Markov Models

- Set of states: $\{s_1, s_2, \ldots, s_N\}$
- Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

- To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(s_i \mid s_j)$ and initial probabilities $\pi_i = P(s_i)$

Markov Models

• Set of states: $\{s_1, s_2, \ldots, s_N\}$

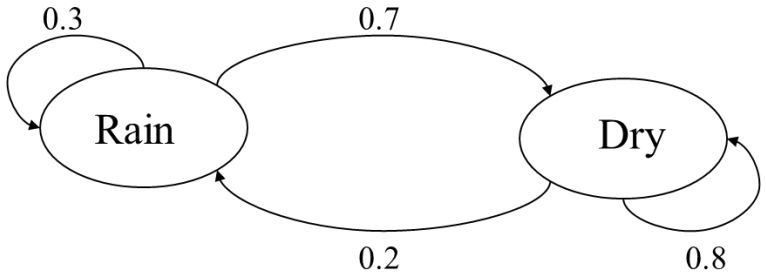
• Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$

• Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

• To define Markov model, the following probabilities have to be specified: transition probabilities $a_{ij} = P(s_i \mid s_j)$ and initial probabilities $\pi_i = P(s_i)$

Example:



• Two states : 'Rain' and 'Dry'.

• Transition probabilities: P('Rain'|'Rain')=0.3 ,

P('Dry'|'Rain')=0.7 , P('Rain'|'Dry')=0.2, P('Dry'|'Dry')=0.8

• Initial probabilities: say P('Rain')=0.4 , P('Dry')=0.6 .

Calculation of observation sequence probability

• By Markov chain property, probability of state sequence can be found by the formula:

$$P(s_{i1}, s_{i2}, \ldots, s_{ik}) = P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1})$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{i1}, s_{i2}, \ldots, s_{ik-1}) = \ldots$$
$$= P(s_{ik} \mid s_{ik-1}) P(s_{ik-1} \mid s_{ik-2}) \ldots P(s_{i2} \mid s_{i1}) P(s_{i1})$$

•Suppose we want to calculate a probability of a sequence of states in our example, {'Dry','Dry','Rain',Rain'}.

P({'Dry','Dry','Rain',Rain'}) = P('Rain'|'Rain') P('Rain'|'Dry') P('Dry'|'Dry') P('Dry')
= 0.3*0.2*0.8*0.6

14.4.1 Hidden Markov Models (HMMs)

• Set of states: $\{s_1, s_2, \ldots, s_N\}$

•Process moves from one state to another generating a sequence of states : $s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots$

• Markov chain property: probability of each subsequent state depends only on what was the previous state:

$$P(s_{ik} \mid s_{i1}, s_{i2}, \ldots, s_{ik-1}) = P(s_{ik} \mid s_{ik-1})$$

• States are not visible, but each state randomly generates one of M observations (or visible states)

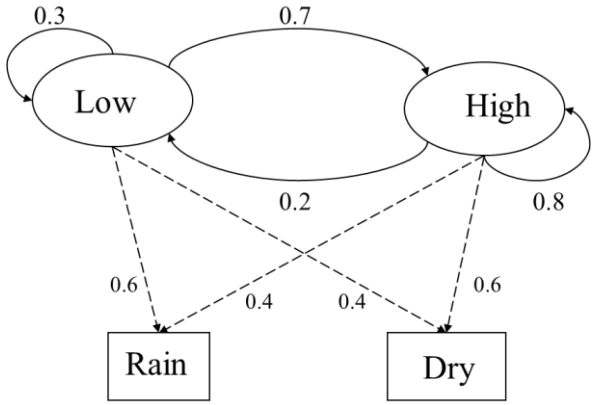
$$\{v_1, v_2, \ldots, v_M\}$$

• To define hidden Markov model, the following probabilities have to be specified:

- Matrix of transition probabilities A=(a_{ij}), a_{ij}= P(s_i | s_j)
- Matrix of observation probabilities B=(b_i (v_m)), b_i(v_m) = P(v_m | s_i)
- A vector of initial probabilities π=(π_i), π_i = P(s_i) .

- Model is represented by M=(A, B, π).

Example



- Two states : 'Low' and 'High' atmospheric pressure.
- Two observations : 'Rain' and 'Dry'.
- Transition probabilities: P('Low'|'Low')=0.3 , P('High'|'Low')=0.7 , P('Low'|'High')=0.2, P('High'|'High')=0.8
- Observation probabilities : P('Rain'|'Low')=0.6 , P('Dry'|'Low')=0.4 , P('Rain'|'High')=0.4 , P('Dry'|'High')=0.3 .
- Initial probabilities: say P('Low')=0.4 , P('High')=0.6 .

•Suppose we want to calculate a probability of a sequence of observations in our example, {'Dry','Rain'}.

•Consider all possible hidden state sequences:

P({'Dry','Rain'}) = P({'Dry','Rain'} , {'Low','Low'}) + P({'Dry','Rain'} , {'Low','High'}) + P({'Dry','Rain'} , {'High','Low'}) + P({'Dry','Rain'} , {'High','High'})

where first term is :

P({'Dry','Rain'} , {'Low','Low'})=

P({'Dry','Rain'} | {'Low','Low'}) P({'Low','Low'}) =

P('Dry'|'Low')P('Rain'|'Low') P('Low')P('Low'|'Low)

= 0.4*0.4*0.6*0.4*0.3

Elements of Discrete Hidden Markov Model

- N: number of states in the model
  - states, $\boldsymbol{s} = \{s_1, s_2, \ldots, s_N\}$
  - state at time t, $q_t \in \boldsymbol{s}$
- M: number of observation symbols (i.e., discrete observations)
  - observation symbols, $\boldsymbol{v} = \{v_1, v_2, \ldots, v_M\}$
  - observation at time t, $o_t \in \boldsymbol{v}$
- $\boldsymbol{A} = \{a_{ij}\}$: state transition probability distribution
  - $a_{ij} = P(q_{t+1} = s_j | q_t = s_i),\ 1 \leq i, j \leq N$
- $\boldsymbol{B} = \{b_j(k)\}$: observation symbol probability distribution in state j
  - $b_j(k) = P(v_k \text{ at } t | q_t = s_j),\ 1 \leq j \leq N, 1 \leq k \leq M$
- $\pi = \{\pi_i\}$: initial state distribution
  - $\pi_i = P(q_1 = s_i),\ 1 \leq i \leq N$

Notationally, an HMM is typically written as: $\lambda = \{\boldsymbol{A}, \boldsymbol{B}, \pi\}$

Continuous or Continuous Density Hidden Markov Model

- A *continuous density* HMM replaces the discrete observation probabilities, $b_j(k)$, by a continuous PDF $b_j(\boldsymbol{x})$
- A common practice is to represent $b_j(\boldsymbol{x})$ as a mixture of Gaussians:

$$b_j(\boldsymbol{x}) = \sum_{k=1}^{M} c_{jk} N[\boldsymbol{x}, \mu_{jk}, \boldsymbol{\Sigma}_{jk}] \qquad 1 \le j \le N$$

where c_{jk} is the mixture weight

$$c_{jk} \ge 0 \quad (1 \le j \le N, 1 \le k \le M, \text{ and } \sum_{k=1}^{M} c_{jk} = 1, 1 \le j \le N),$$

N is the normal density, and
μ_{jk} and $\boldsymbol{\Sigma}_{jk}$ are the mean vector and covariance matrix associated with state j and mixture k.

14.5 Feature Selection and Extraction

What? *Selecting the most "relevant" subset of attributes according to some selection criteria.*

Why?

High-dimensional data often contain irrelevant or redundant features
– reduce the accuracy of data mining algorithms
– slow down the mining process
– be a problem in storage and retrieval
– hard to interpret

How?

May improve performance of learning algorithm
– Learning algorithm may not scale up to the size of the full feature set either in sample or time
– Allows us to better understand the domain
– Cheaper to collect a reduced set of features

What

Generally, features are characterized as:
– Relevant: These are features which have an influence on the output and their role can not be assumed by the rest
– Irrelevant: Irrelevant features are defined as those features not having any influence on the output, and whose values are generated at random for each example.
– Redundant: A redundancy exists whenever a feature can take the role of another (perhaps the simplest way to model redundancy).

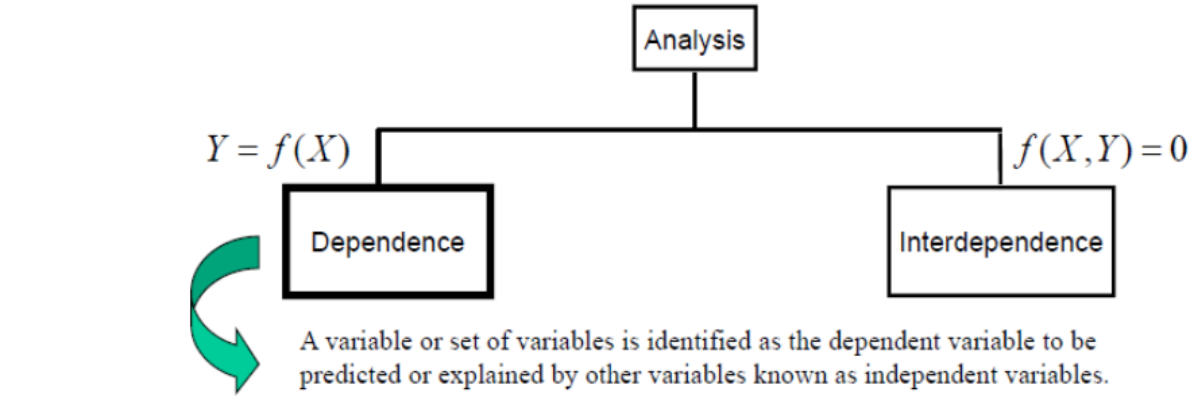
14.5.1 Multivariate Data

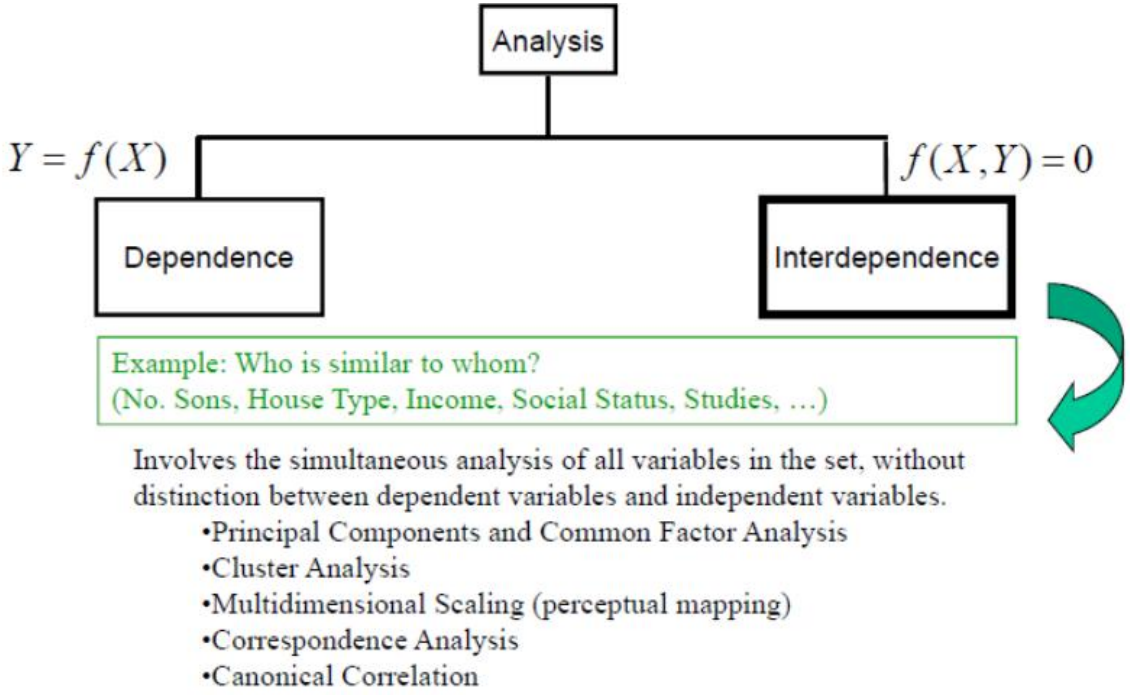
These are those data that contain more than one variable

Multivariate Analysis (MVA): Statistical analysis of data collected on more than one (response) variable

Multivariate Analysis Methods:

- Analysis of dependence: Dependent variables are predicted by other variables (Regression)
- Analysis of interdependence: Looks at the relationship among variables, objects and classes (Clustering)





Objectives of Multivariate Data

1. Data reduction or structural simplification. To simplify without loosing any valuable information and make interpretation easier.
2. Sorting and grouping. Similar objects or variables are grouped, based upon the characteristics. Define rules for classifying objects into well-defined groups.
3. Investigation of the dependence among variables. The nature of the relationships among variables is of interest. Are all the variables mutually dependent/ independent?
4. Prediction. Relationships between variables must be determined for the purpose of predicting the values of one or more variables on the basis of observations on the other variables.
5. Hypothesis construction and testing. Specific statistical hypotheses, formulated are tested.

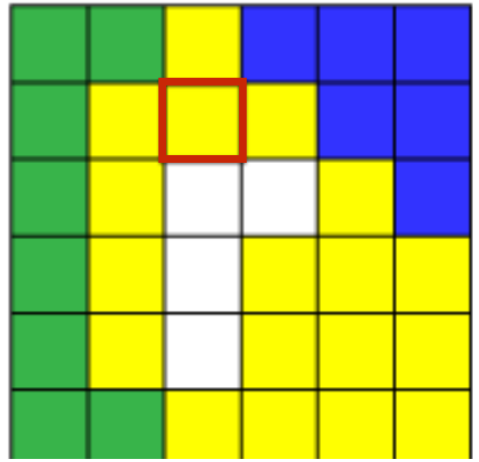
Types of Multivariate Analysis

- **Exploratory Data Analysis (EDA):** Sometimes called data mining this area is useful for gaining deeper insights into large, complex data sets.
- **Regression analysis:** Develops models to predict new and future events. Is useful for predictive analytics applications.
- **Classification for identifying new or existing classes:** This area is useful in research, development, market analysis, etc.

14.5.2 Image Data

It is a photographic or trace objects that represent the underlying pixel data of an area of an image element, which is created, collected and stored using image constructor devices. Images may contain any data that can be represented in a two-dimensional matrix.

A picture that we see is actually comprised of tiny squares of color. These "squares" represent one pixel. A pixel holds just one color of information. An image is broken down into the smallest single piece of information: a single color. This one color is represented as a number. An image is broken down into pixels. Each pixel holds the information about its color. Therefore, a pixel's role is to represent a single color. Combine these pixels together to paint the picture using the grid.

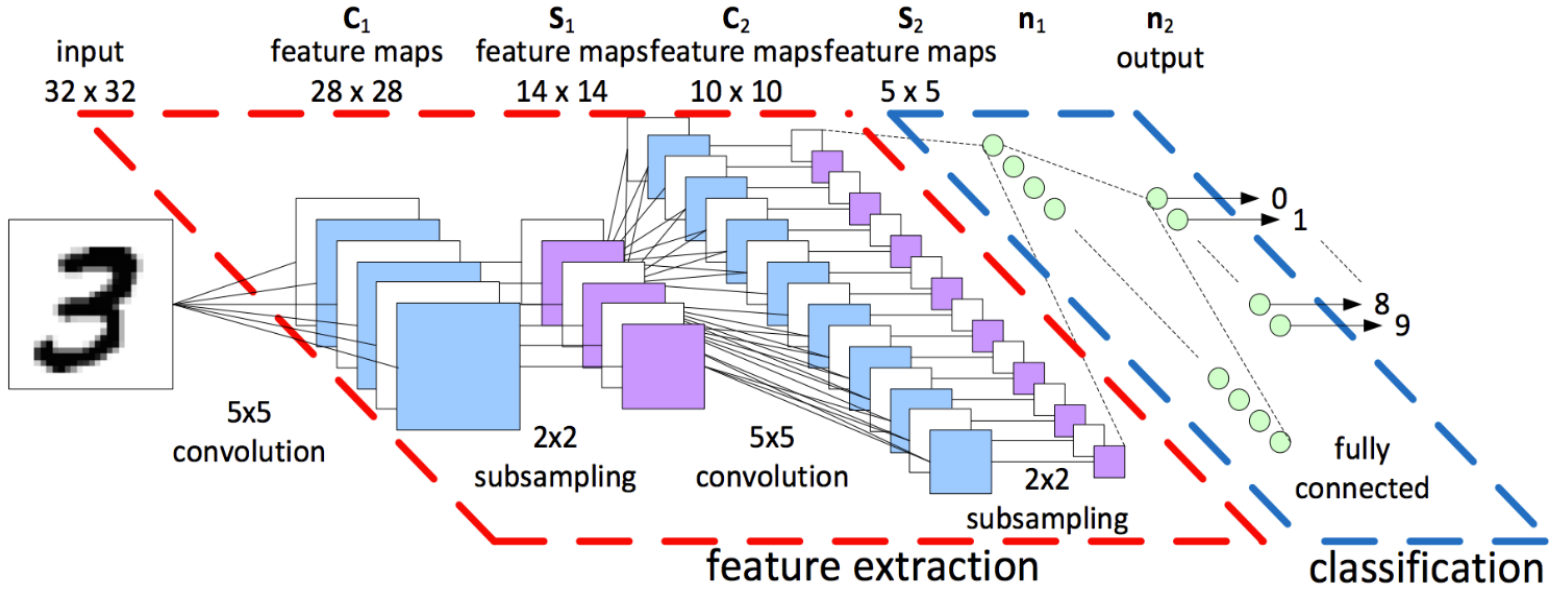


Applications of Image Data

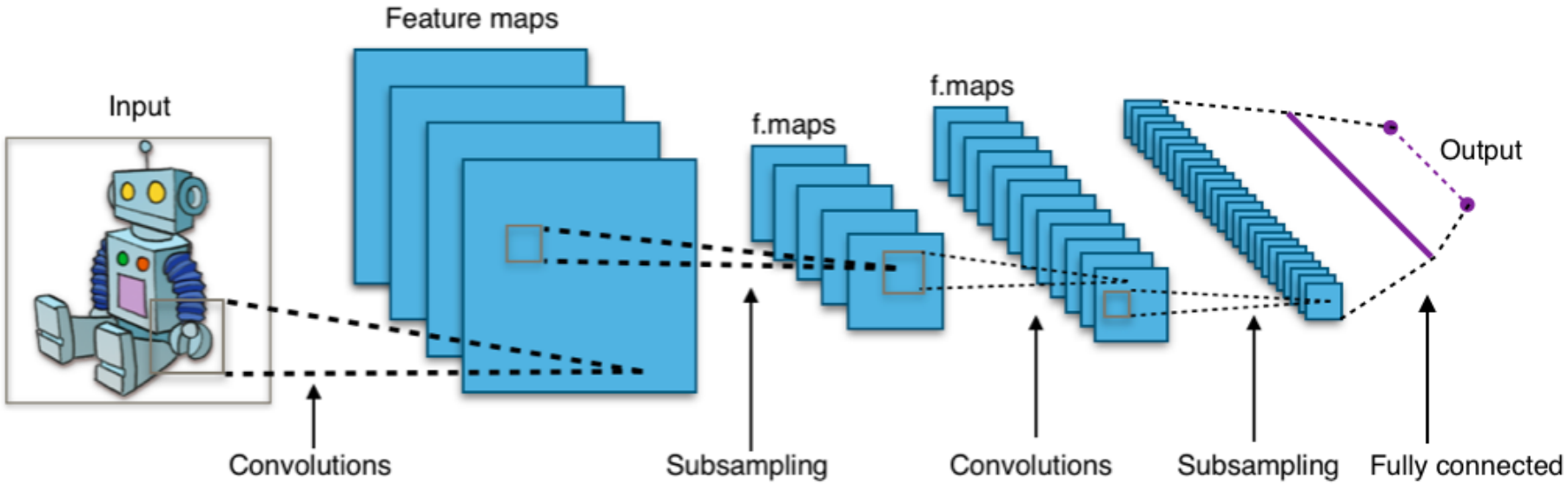
Image Classification: Label an image with a predefined set of categories based on objects present.
Segmentation: Break up an image into cluster regions according to object occupancy and property.
Object Detection: Detecting and categorizing the object.
Object Tracking: Define the location and keeps track of objects across a sequence of images.
Image Reconstruction: Extracting the information about geometry in the image.
Image Geometry: Computing the depth of objects from the camera.
Image Generation: Merging different image styles or generating completely new ones.

14.6 Deep Learning, Convolution Neural Nets

- The better the features are, the more accurate the results are going to be.
- In recent periods, the features have become more precise and better accuracy has been achieved.
- This is due to a new kind of feature extractor called **Convolutional Neural Networks** (**CNNs**).
- CNN shown remarkable accuracy in complex tasks such as object detection and classifying images with high accuracy.
- Applications ranging from smartphone photo enhancements to satellite image analysis.



- Convolution or Feature Extraction Layer (+ ReLU)
- Pooling or Subsampling Layer (max or average Pooling)
- Fully Connected + ReLU
- Softmax

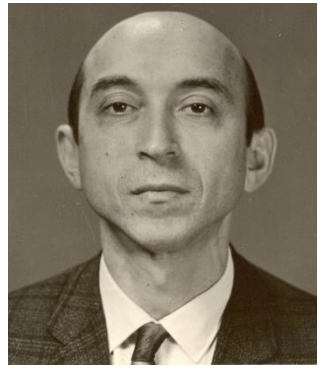


Note: CNN hidden layers are basically alternation of convolution and pooling layers.

Note: Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a *sliding dot product* or cross-correlation.

14.7 Fuzzy Logics, Genetic Algorithm

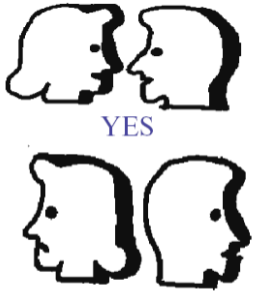
Fuzzy Logic was initiated in 1965, by Dr. Lotfi A. Zadeh, professor for computer science at the university of California in Berkley.



- Fuzzy logic is a mathematical tool for dealing with uncertainty.
- It provides a technique to deal with imprecision and information granularity.
- The fuzzy theory provides a mechanism for representing linguistic constructs such as "many," "low," "medium," "often," "few."

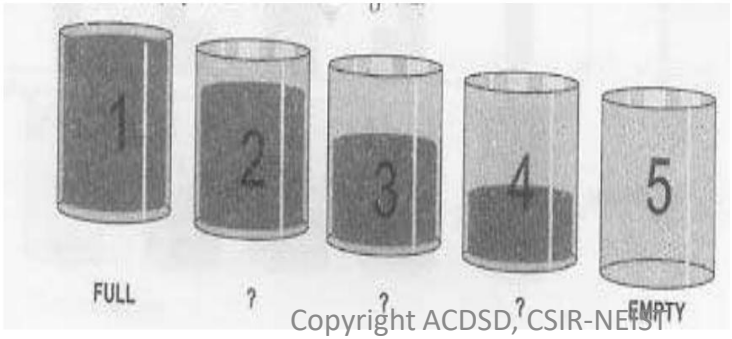
The term "fuzzy logic" refers to a logic of approximation.

- Boolean logic assumes that every fact is either entirely true or false.
- Fuzzy logic allows for varying degrees of truth.
- Computers can apply this logic to represent vague and imprecise ideas, such as "hot", "tall" or "balding".



Binary logic

Fuzzy logic



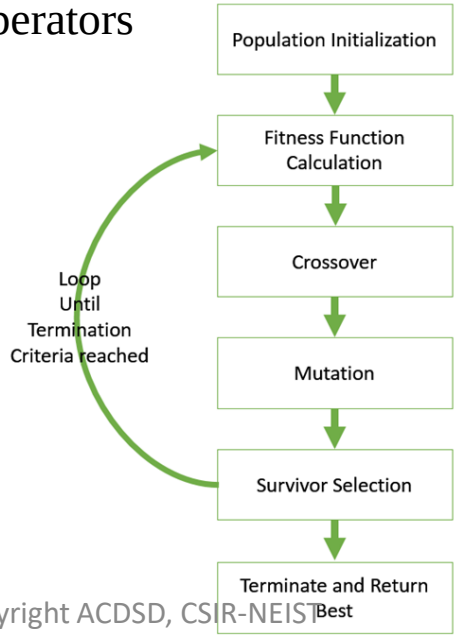
A **genetic algorithm** is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

- Typically applied to:
  - discrete optimization
- Attributed features:
  - not too fast
  - good heuristic for combinatorial problems
- Special Features:
  - Traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators

Representation	Binary strings
Recombination	N-point or uniform
Mutation	Bitwise bit-flipping with fixed probability
Parent selection	Fitness-Proportionate
Survivor selection	All children replace parents
Speciality	Emphasis on crossover

Phases of Genetic Algorithm

- Initial population
- Fitness function
- Selection
- Crossover
- Mutation

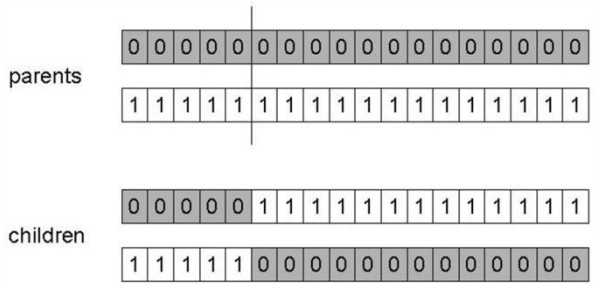


Initial population

1. Select parents for the mating pool
   (size of mating pool = population size)
2. Shuffle the mating pool
3. For each consecutive pair apply crossover with probability p_c , otherwise copy parents
4. For each offspring apply mutation (bit-flip with probability p_m independently for each bit)
5. Replace the whole population with the resulting offspring

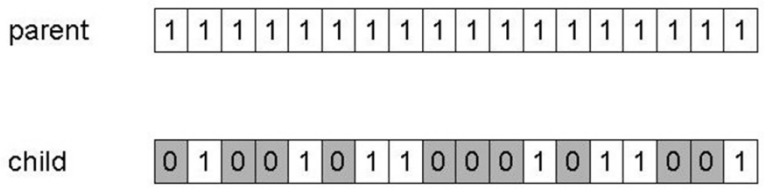
Crossover

- Choose a random point on the two parents
- Split parents at this crossover point
- Create children by exchanging tails
- P_c typically in range (0.6, 0.9)



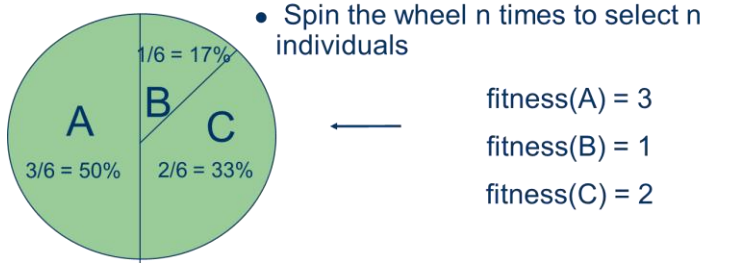
Mutation

- Alter each gene independently with a probability p_m
- p_m is called the mutation rate
  - Typically between 1/population _ size and 1/ chromosome_length



Selection

- Main idea: better individuals get higher chance
  - Chances proportional to fitness
  - Implementation: roulette wheel technique
    - Assign to each individual a part of the roulette wheel
    - Spin the wheel n times to select n individuals



14.8 Sensors and Data Fusion

- Problem solving based on the idea of integrating many answers into a single; the best answer
- Combined data and information from various sensors to provide robust and complete description of the process
- Deals with automatic – detection, association, correlation, estimation and combination

- Advantages:
  - Improves accuracy
  - Improves precision
  - Improves availability
  - Reduces uncertainty
  - Support effective decision making

Data fusion combines two expertise areas

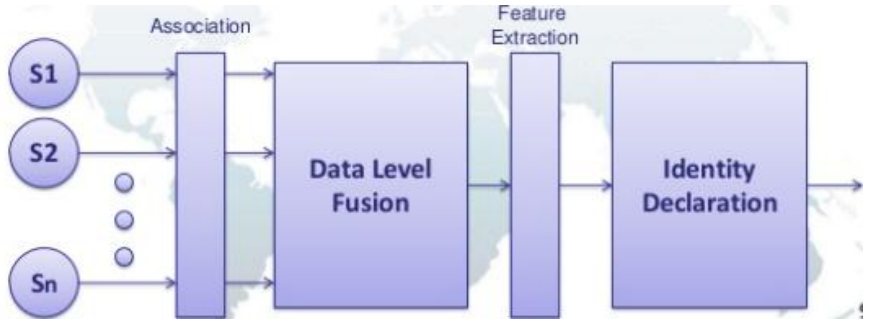
- Sensors
- Information integration

Data fusion is categorized into three types:

- Measurement fusion (Sensor data fusion)
- Feature-level fusion
- Decision-level fusion (High level data fusion)

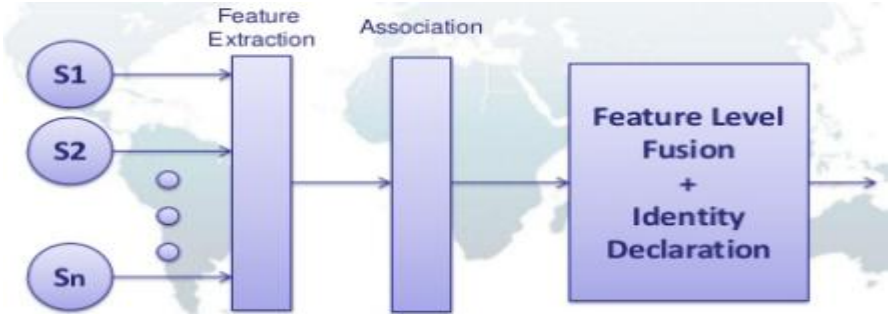
Measurement Fusion (Sensor data fusion)

- ❑ Direct fusion of data sensors
- ❑ Sensors measuring homogenous physical phenomena



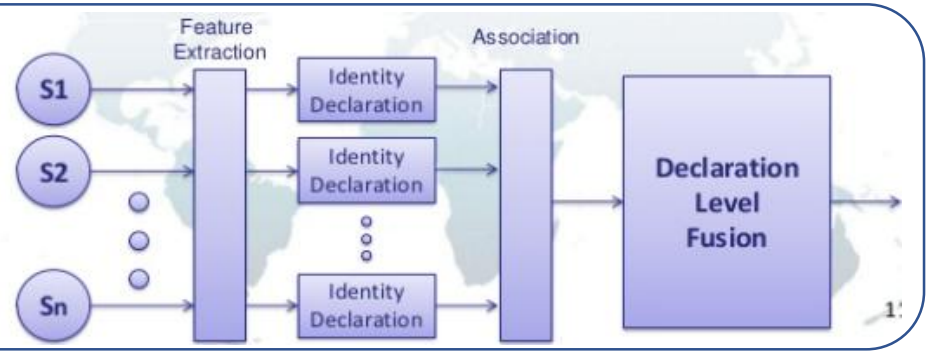
Feature-level Fusion

- ❑ Extraction of representative features from sensors
- ❑ Features are combined into a single concatenated feature vector



Decision-level Fusion (High-level data fusion)

- ❑ Each sensor makes preliminary determination of an entity's location
- ❑ Algorithms like weighted decision, Bayesian inference etc. are used



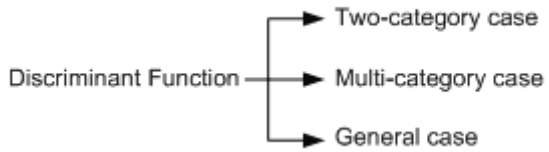
14.9 Linear Discriminant Functions

- Linear discriminant functions can be optimal if the underlying distributions are cooperative, such as Gaussians having equal covariance, as might be obtained through an intelligent choice of feature detectors.
- Linear discriminant functions are relatively easy to compute and in the absence of information suggesting otherwise, linear classifiers are attractive candidates for initial, trial classifiers.

A discriminant function that is a linear combination of the components of x can be written as

$$g(x) = \mathbf{W}^T x + w_0$$

where **w** is the weight vector and w_0 the bias or threshold weight



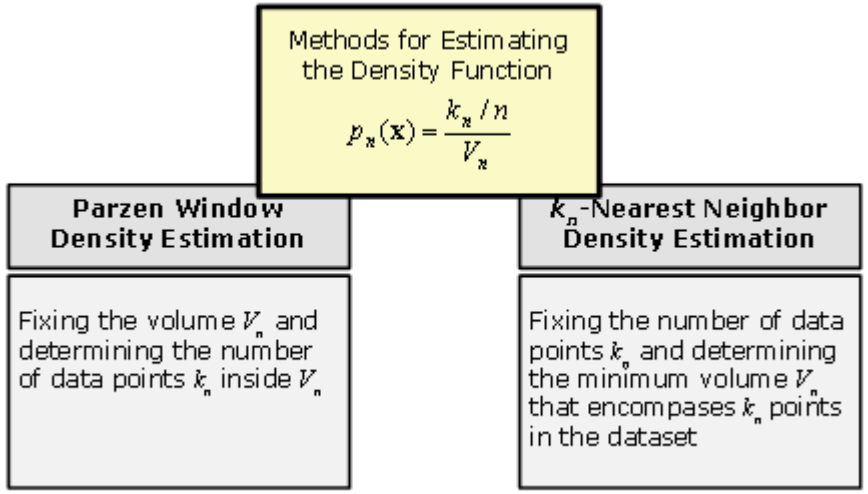
14.10 Nonparametric Pattern Recognition

Non parametric methods can be used with arbitrary distribution and without assumption that the forms of densities underlying are known.

Two important Nonparametric methods are :

- ❑ Estimating the density function
- ❑ Bypass estimating the density and so directly to estimate a posteriori probability

Density estimation methods: a) Prazen window and b) K-Nearest Neighbour



14.11 Algorithm-independent Learning

Among all the Machine Learning models:

- Is some classifier better than all others?
- How to compare classifiers?
- Is comparison possible at all?
- Is at least some classifier always better than random?
- Do techniques exist which boost all classifiers?

No Free Lunch Theorem

For any two learning algorithms $P_1(h|D)$ and $P_2(h|D)$, the following are true, independent of the sampling distribution P(x) and the number n of training points:

1. Uniformly averaged over all target functions F,
   $\varepsilon_1(E|F,n) - \varepsilon_2(E|F,n) = 0$.
2. For any fixed training set D, uniformly averaged over F,
   $\varepsilon_1(E|F,D) - \varepsilon_2(E|F,D) = 0$
3. Uniformly averaged over all priors P(F),
   $\varepsilon_1(E|n) - \varepsilon_2(E|n) = 0$
4. For any fixed training set D, uniformly averaged over P(F),
   $\varepsilon_1(E|D) - \varepsilon_2(E|D) = 0$

No Free Lunch Theorem

1. Uniformly averaged over all target functions F,

   $\mathcal{E}_1(E|F,n) - \mathcal{E}_2(E|F,n) = 0.$

Average over all possible target functions, the error will be the same for all classifiers.

Possible target functions: 2^5

2. For any fixed training set D, uniformly averaged over F,

   $\mathcal{E}_1(E|F,D) - \mathcal{E}_2(E|F,D) = 0$

Even if we know the training set D, the off-training errors will be the same.

	x	F	h_1	h_2
Training set D	000	1	1	1
	001	-1	-1	-1
	010	1	1	1
Off-Training set	011	-1	1	-1
	100	1	1	-1
	101	-1	1	-1
	110	1	1	-1
	111	1	1	-1

Consequence of No Free Lunch Theorem

If no information about the target function F(x) is provided:

- No classifier is better than some other in the general case
- No classifier is better than random in the general case

14.11 Algorithm-independent Learning

Among all the Machine Learning models:

- Is some classifier better than all others?
- How to compare classifiers?
- Is comparison possible at all?
- Is at least some classifier always better than random?
- Do techniques exist which boost all classifiers?

No Free Lunch Theorem

For any two learning algorithms $P_1(h|D)$ and $P_2(h|D)$, the following are true, independent of the sampling distribution P(x) and the number n of training points:

1. Uniformly averaged over all target functions F,
   $\varepsilon_1(E|F,n) - \varepsilon_2(E|F,n) = 0$.
2. For any fixed training set D, uniformly averaged over F,
   $\varepsilon_1(E|F,D) - \varepsilon_2(E|F,D) = 0$
3. Uniformly averaged over all priors P(F),
   $\varepsilon_1(E|n) - \varepsilon_2(E|n) = 0$
4. For any fixed training set D, uniformly averaged over P(F),
   $\varepsilon_1(E|D) - \varepsilon_2(E|D) = 0$

Statistical pattern recognition is straightforward, but may not be ideal for many realistic problems. Patterns that include structural or relational information are difficult to quantify as feature vectors.

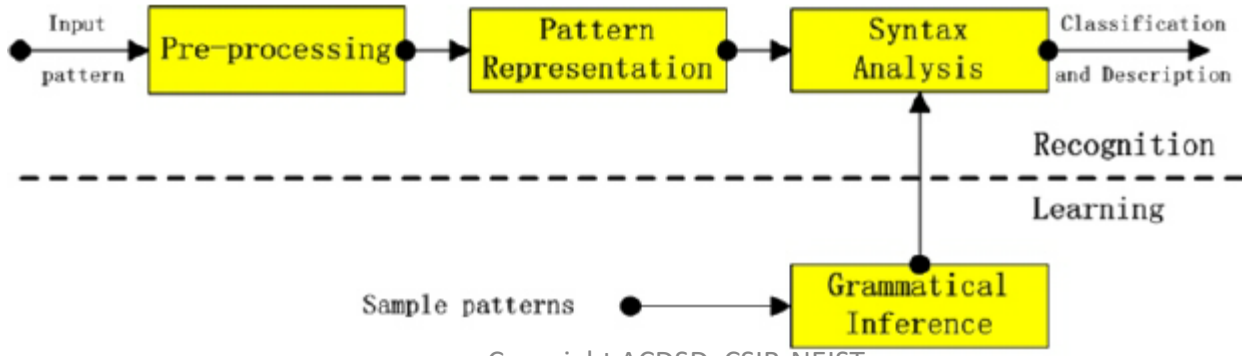
- Syntactic pattern recognition uses this structural information for classification and description.
- Grammars can be used to create a definition of the structure of each pattern class.

Syntactic Pattern Recognition

Consists of two main parts:

- Analysis - primitive selection and grammatical or structural inference
- Recognition - preprocessing, segmentation or decomposition, primitive and relation recognition, and syntax analysis

Preprocessing includes the tasks of pattern encoding and approximation, filtering, restoration, and enhancement.



14.13 Examples

14.14 Exercises

1. What are the problems arise by activities in design of pattern recognition System?
2. Consider training and HMM by the forward and backward algorithm for a single sequence of length T where each symbol could be one of c values. What is the computational complexity of a single revision of all values aij and bij
3. When a test pattern is classified by a decision tree, that pattern is subjected to a sequence of queries, corresponding to the nodes along a path from root to leaf ? Prove that for any decision tree.
4. Write algorithm for K-means clustering with the help of diagram. Explain how the K-means clustering produces a form of stochastic hill climbing in the log likelihood function.
5. How is image acquisition done with sensor arrays?

14.15 References

- Friedman, M. and Kandel, A., 1999. *Introduction to pattern recognition: statistical, structural, neural and fuzzy logic approaches* (Vol. 32). World Scientific Publishing Company. ★ ★★ ★
  - ➢ An introduction to pattern recognition has been covered. In addition to statistical and structural approaches, novel topics such as fuzzy pattern recognition and pattern recognition via neural networks are also reviewed.

- Bishop, C.M., 2006. *Pattern recognition and machine learning*. springer.
  - ➢ This is the first textbook on pattern recognition to present the Bayesian viewpoint. it uses graphical models to describe probability distributions. ★ ★★ ★

- Duda, R.O., Hart, P.E., and Stork D.G., (2001). *Pattern Classification*. (2nd ed.). New York: Wiley-Interscience Publication.
  - ➢ Provides information on key new topics such as neural networks and statistical pattern recognition, the theory of machine learning, and the theory of invariances. ★ ★★ ★

- Schalkoff R. J. (1997). *Artificial Neural Networks*. The McGraw-Hill Press, USA. ★ ★★
  - ➢ This book provides a detailed knowledge on artificial neural networks and its perspectives in different areas.

Thank You