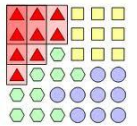
# Data Mining and Data Warehousing

## Chapter 4

## Classification and Prediction

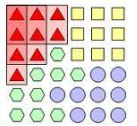
Instructor: Suresh Pokharel
ME in ICT (Asian Institute of Technology, Thailand)
BE in Computer ( NCIT, Pokhara University)

# What is classification?



- is a data mining technique used to predict the category of categorical data by building a model based on some predictor variables (to classify data).
- Predictor variable/attribute is called class label attribute (predefined class)

# What is classification?



It is a two-step process

1. Model Construction (learning step or training phase)

   - build a model to explain the target concept

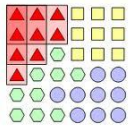
   - model is represented as classification rules, decision trees, or mathematical formulae.

2. Model Usage

   - is used for classifying future or unknown cases

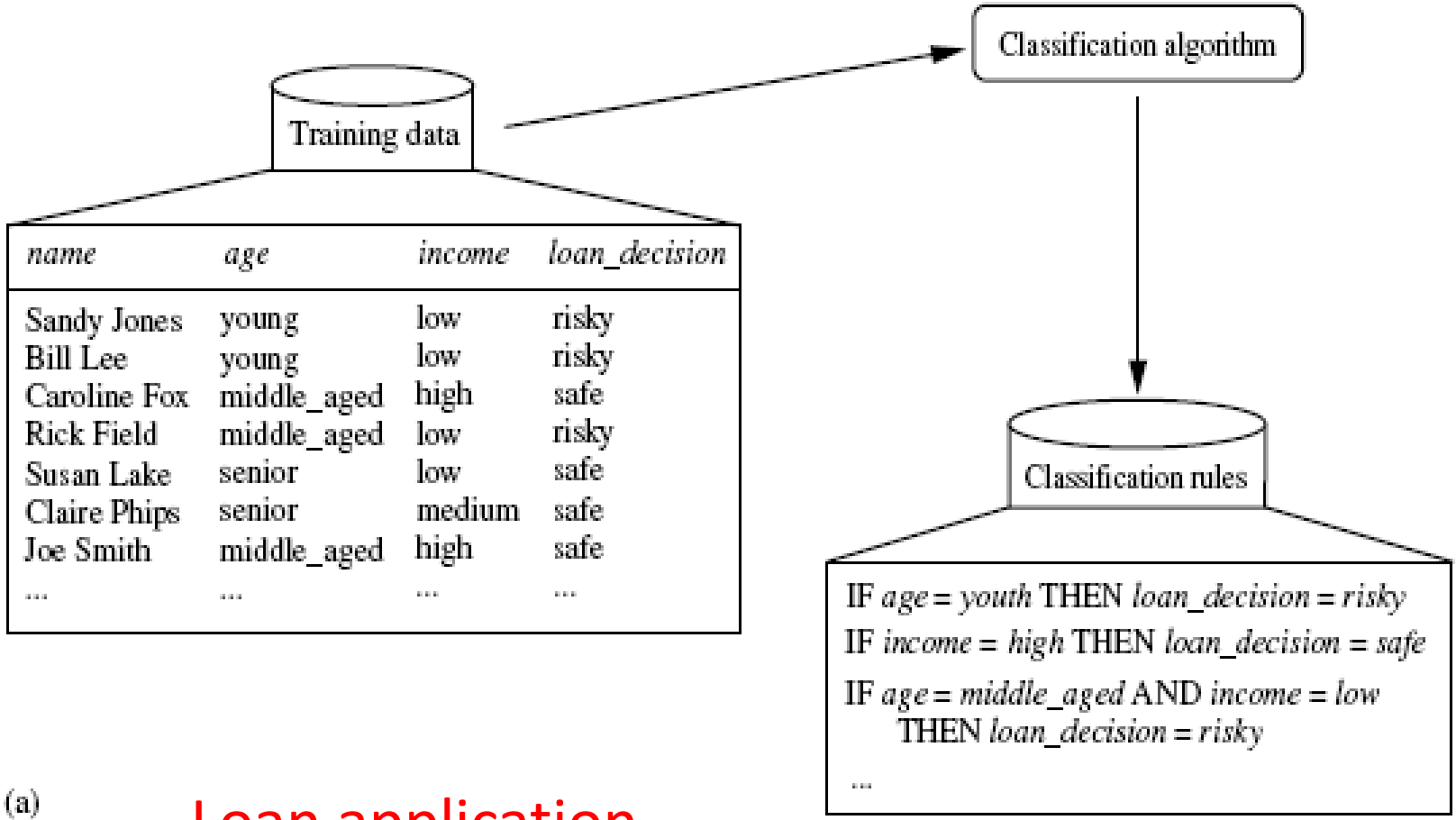
   - estimate the accuracy of the model

# Example



| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Step 1

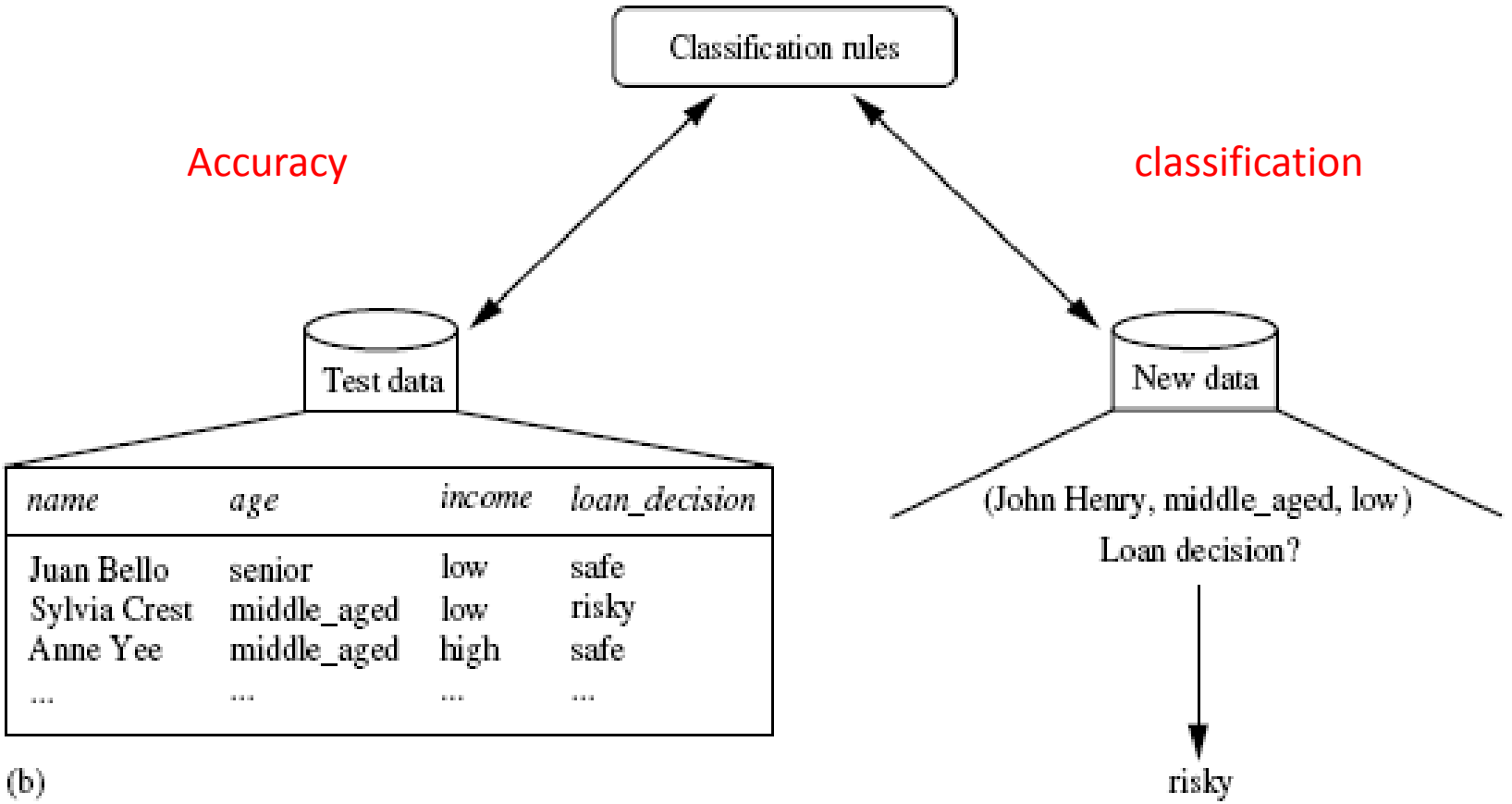



| *name* | *age* | *income* | *loan_decision* |
|---|---|---|---|
| Sandy Jones | young | low | risky |
| Bill Lee | young | low | risky |
| Caroline Fox | middle_aged | high | safe |
| Rick Field | middle_aged | low | risky |
| Susan Lake | senior | low | safe |
| Claire Phips | senior | medium | safe |
| Joe Smith | middle_aged | high | safe |
| ... | ... | ... | ... |

(a)

Loan application

# Step 2

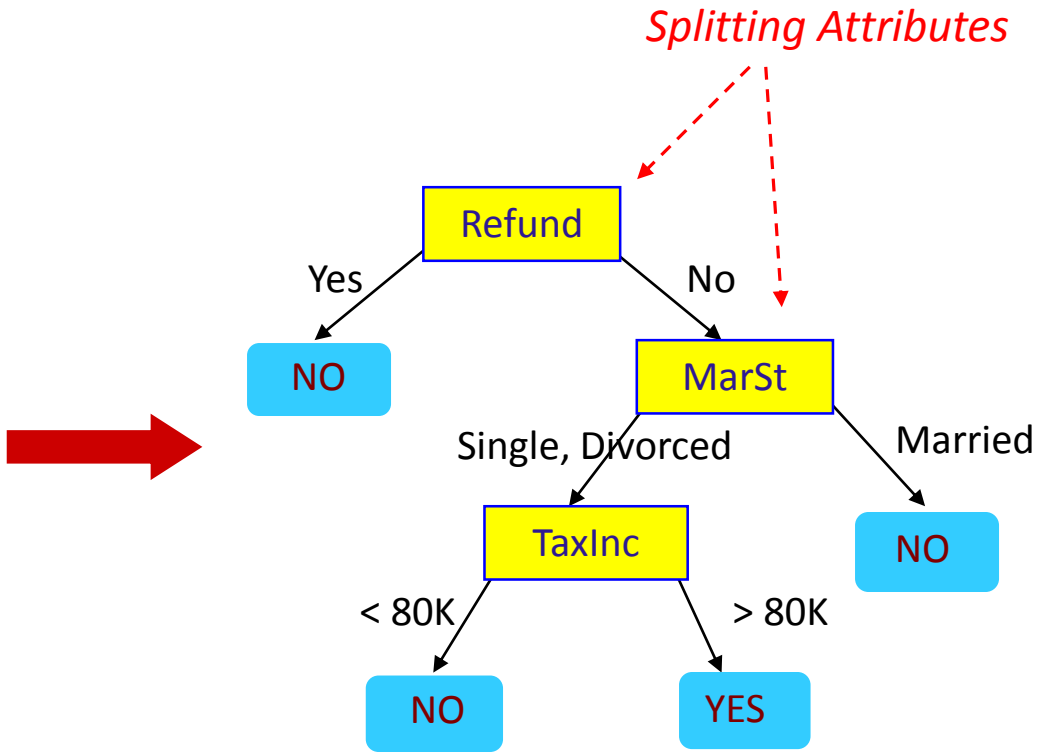

Classification rules
Accuracy
classification
Test data
New data
name age income loan_decision
Juan Bello senior low safe
Sylvia Crest middle_aged low risky
Anne Yee middle_aged high safe
... ... ... ...
(John Henry, middle_aged, low)
Loan decision?
risky
(b)

# Example of a Decision Tree

categorical categorical continuous class

| Tid | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 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 |

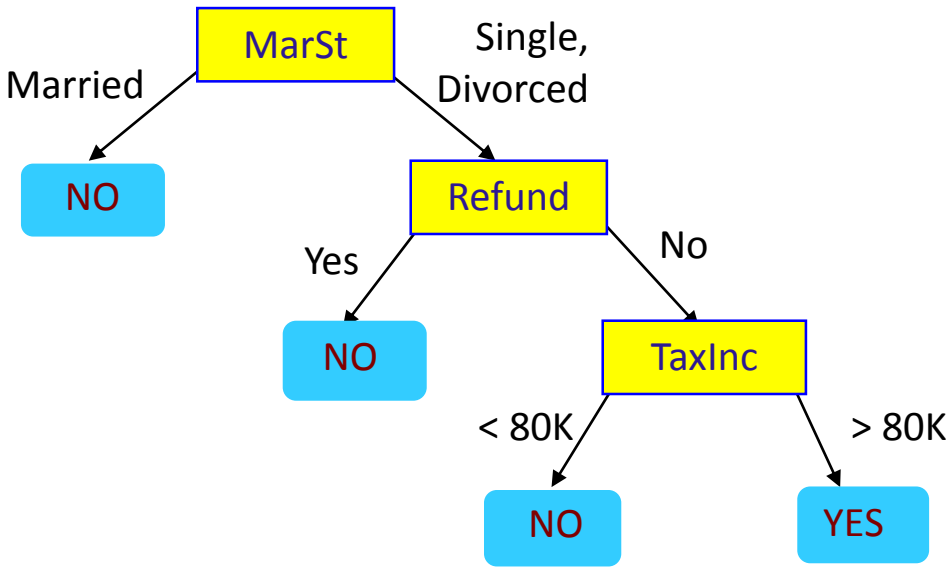
Training Data




Model: Decision Tree

# Another Example of Decision Tree

| | categorical | categorical | continuous | class |
|---|---|---|---|---|
| *Tid* | **Refund** | **Marital Status** | **Taxable Income** | **Cheat** |
| 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** |




There could be more than one tree that fits the same data!

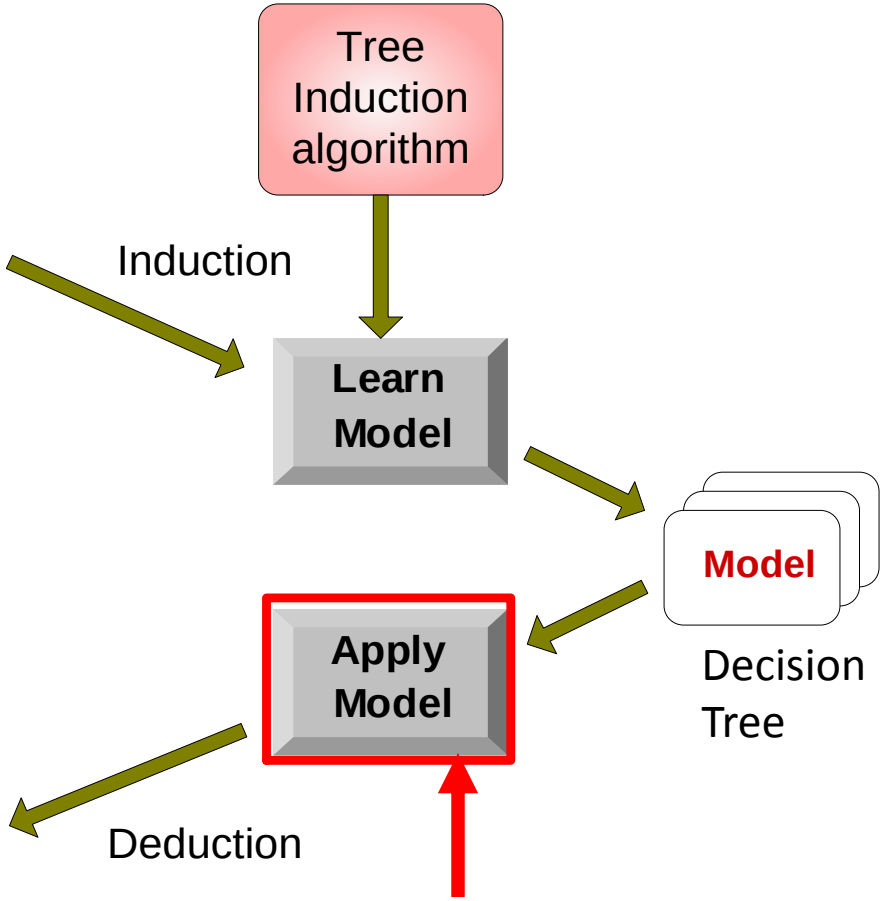
# Decision Tree Classification Task

| 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

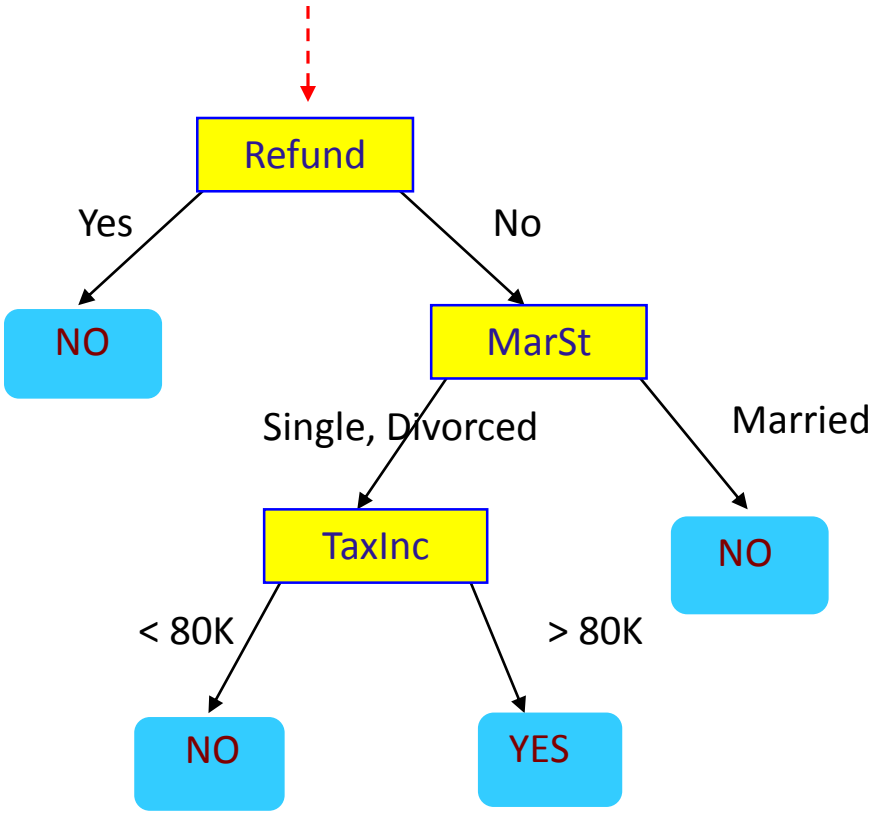
| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



# Apply Model to Test Data

Start from the root of tree.



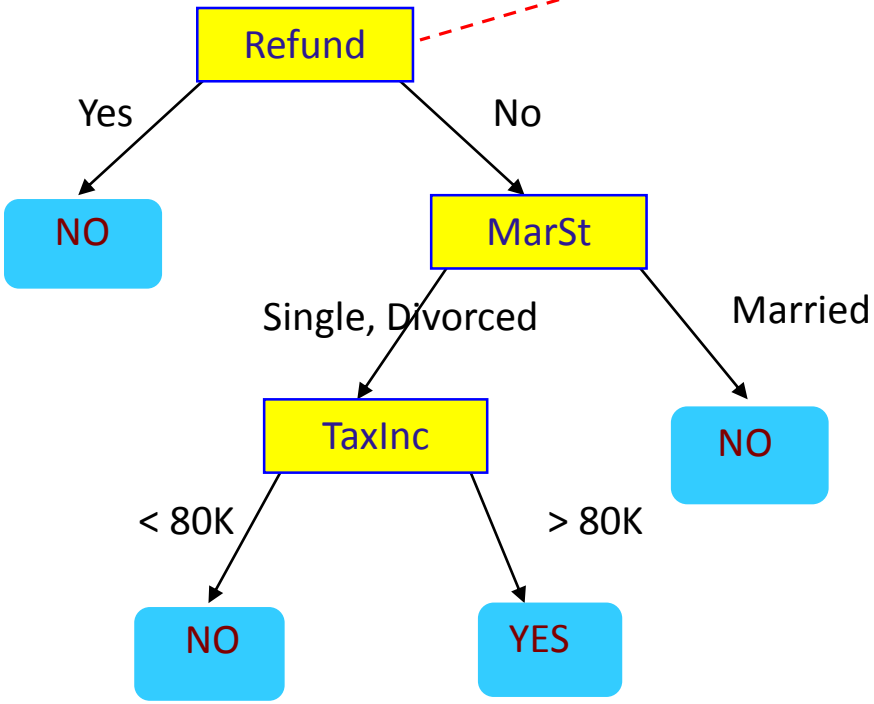

Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |

# Apply Model to Test Data

Test Data

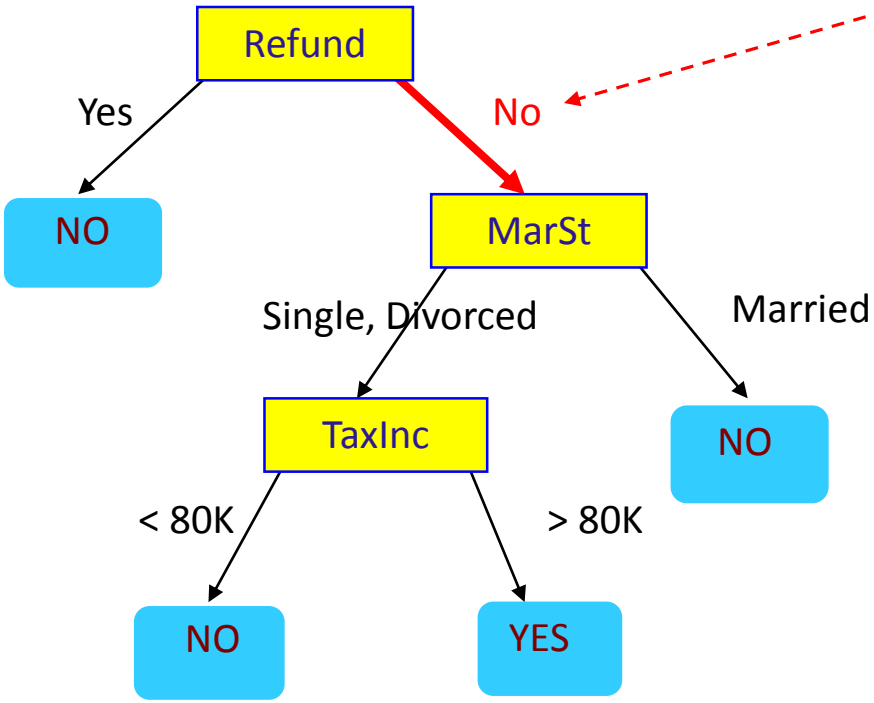
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



# Apply Model to Test Data

Test Data

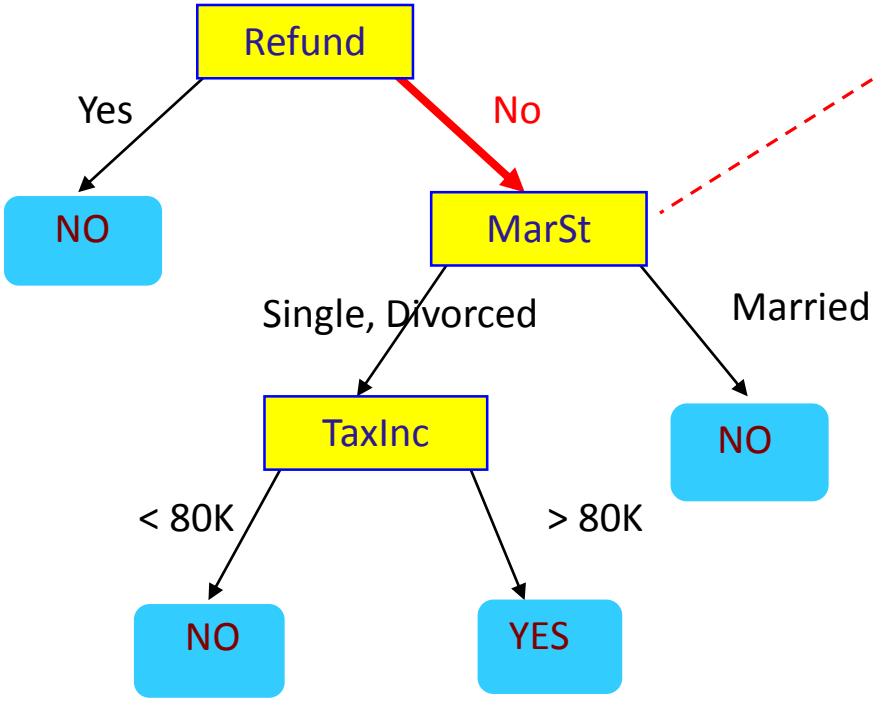
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



# Apply Model to Test Data

Test Data

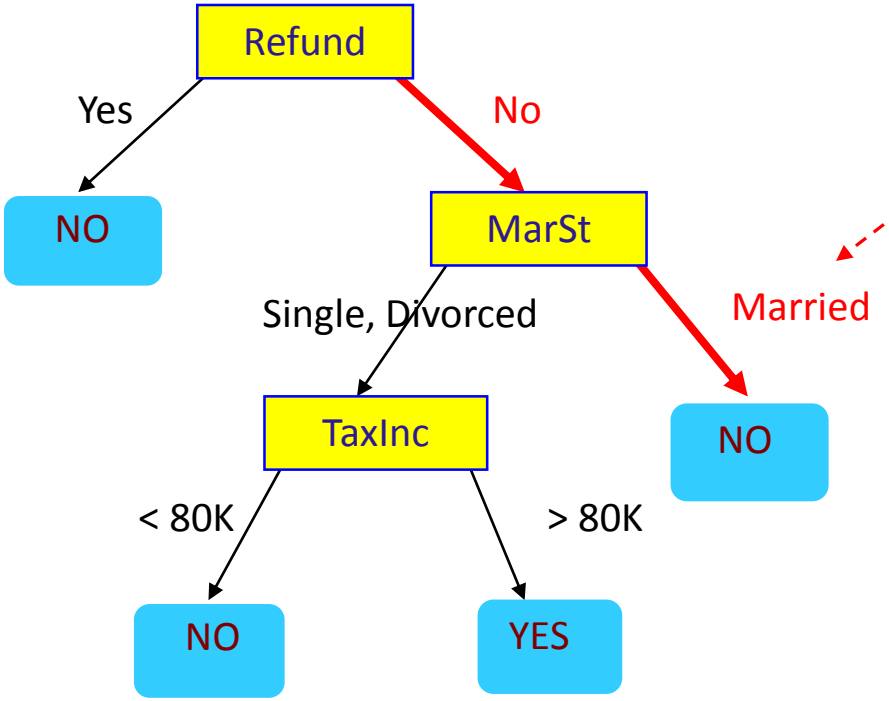
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



# Apply Model to Test Data

Test Data

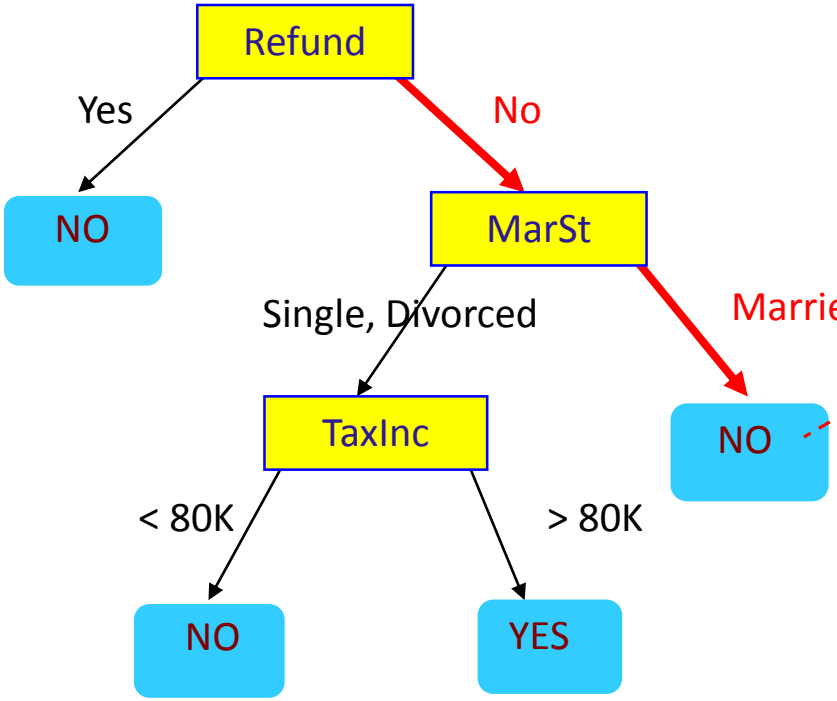
| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |



# Apply Model to Test Data

Test Data

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Married | 80K | ? |




Assign Cheat to “No”

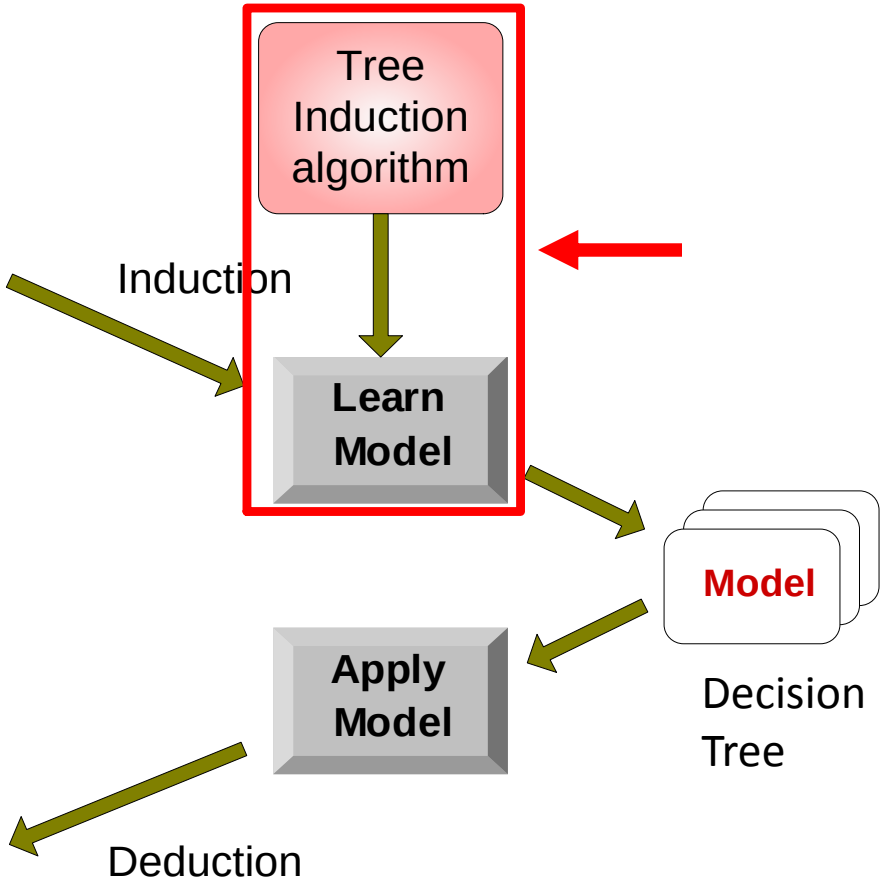
# Decision Tree Classification Task

| 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 |

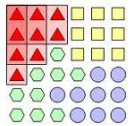
Training Set

| 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

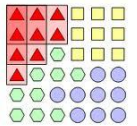


# What is Prediction?



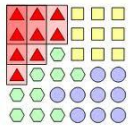
- Models continuous-valued functions, i.e. predicts unknown or missing values (numeric)
- Lost terminology of "class label attribute", instead we use "predicted attribute"
- Viewed as a mapping or function $y = f(X)$
- Example: predict the amount (in dollars) that would be safe for the bank to loan an application

# Supervised & Unsupervised Learning



- Supervised Learning (Classification)
  - Supervision: The training data are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised Learning (Clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

# Issue regarding classification & Prediction



- Data Preparation

  data cleaning, relevant analysis, data transformation and reduction

- Comparing classification & prediction method

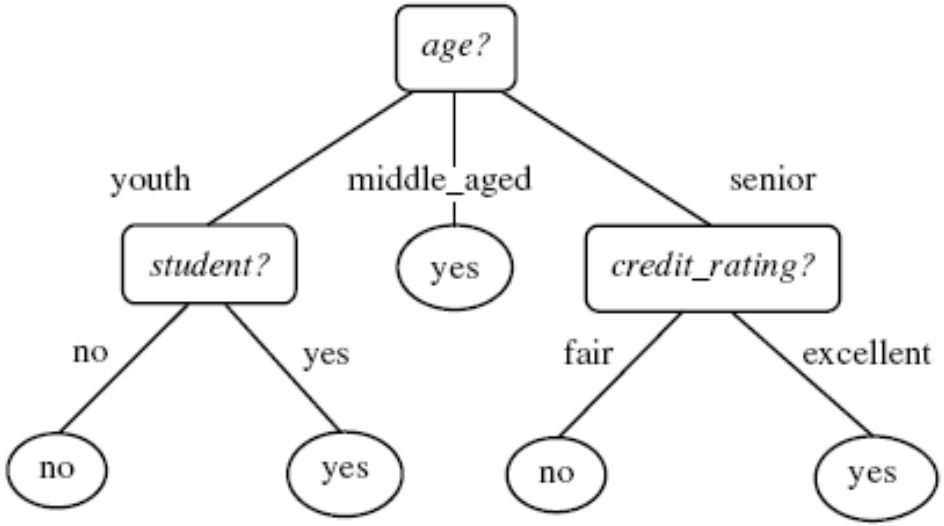
  Accuracy, speed, Robustness, scalability, interpretability

# Classification by decision tree induction

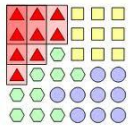
What is decision tree?

- flow chart like tree structure
- internal node denotes a test on an attribute
- each branch represents an outcome of the test
- each leaf node holds a class label

**Decision tree for the concept buys computer**

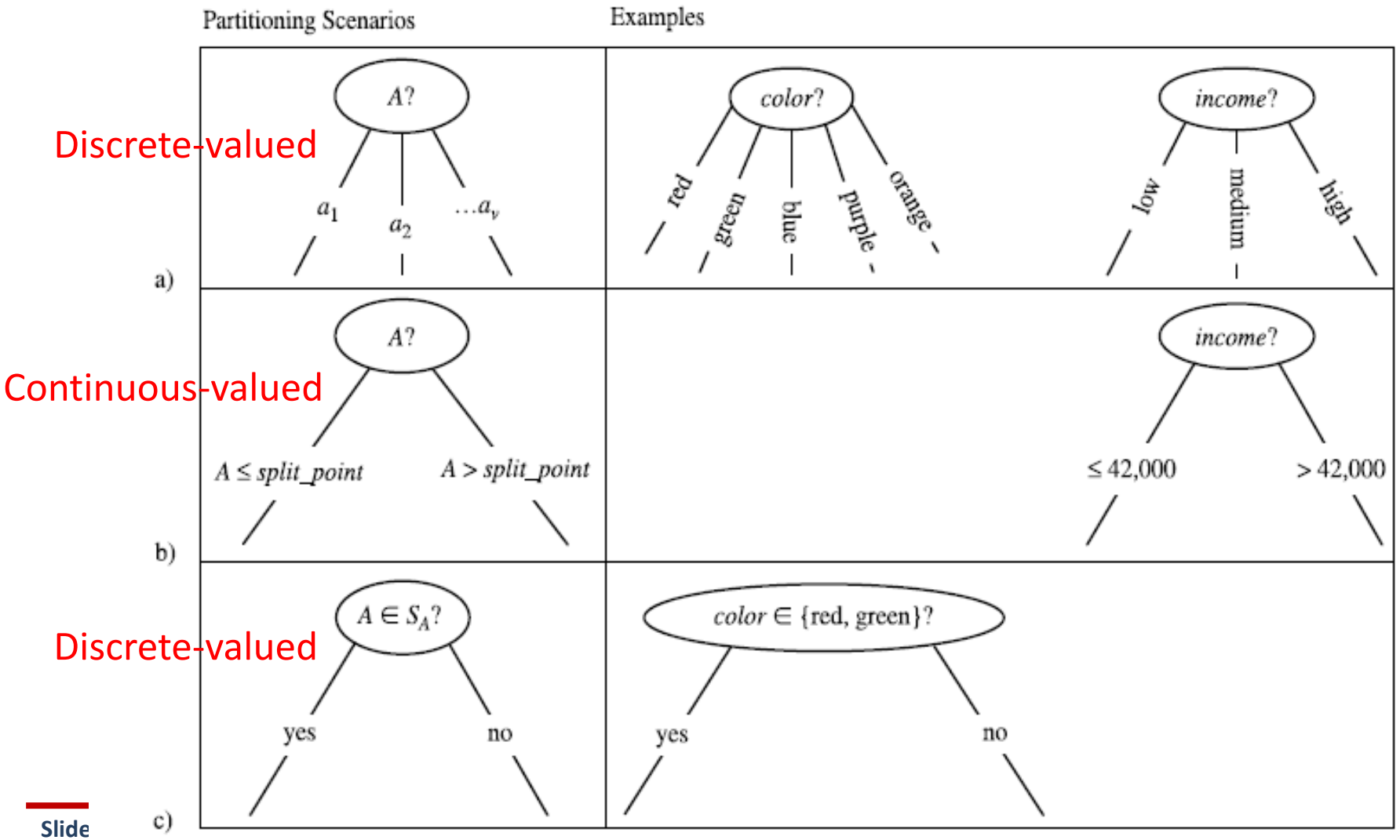


# Why decision tree?



- Construction does not require any domain knowledge
- Can handle high dimensional data
- Learning step is simple and fast
- In general have good accuracy
- People are able to understand decision tree models after a brief explanation

# Example


Partitioning Scenarios
Examples
Discrete-valued
Continuous-valued
Discrete-valued
a)
b)
c)
A?
$a_1$
$a_2$
$\ldots a_v$
color?
red
green
blue
purple
orange
income?
low
medium
high
A?
$A \leq split\_point$
$A > split\_point$
income?
≤ 42,000
> 42,000
$A \in S_A$?
yes
no
color ∈ {red, green}?
yes
no

# Attribute selection measures

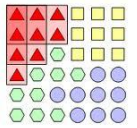


**Table 6.1** Class-labeled training tuples from the *AllElectronics* customer database.

| *RID* | *age* | *income* | *student* | *credit_rating* | *Class: buys_computer* |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

Information Gain, gain ratio and gini index

# Attribute selection measures

- Information Gain

the expected information needed to classify a tuple in *D :* (entropy of *D)*

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

- *Pi = |C*
- *A* having *v* distinct values, $\{a_1, a_2, \ldots, a_v\}$
- $D_1, D_2, \ldots, D_v$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

# Attribute selection measures

$$Gain(A) = Info(D) - Info_A(D).$$ (Choose maximum value)

**Table 6.1** Class-labeled training tuples from the *AllElectronics* customer database.

| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Example A is discrete-valued

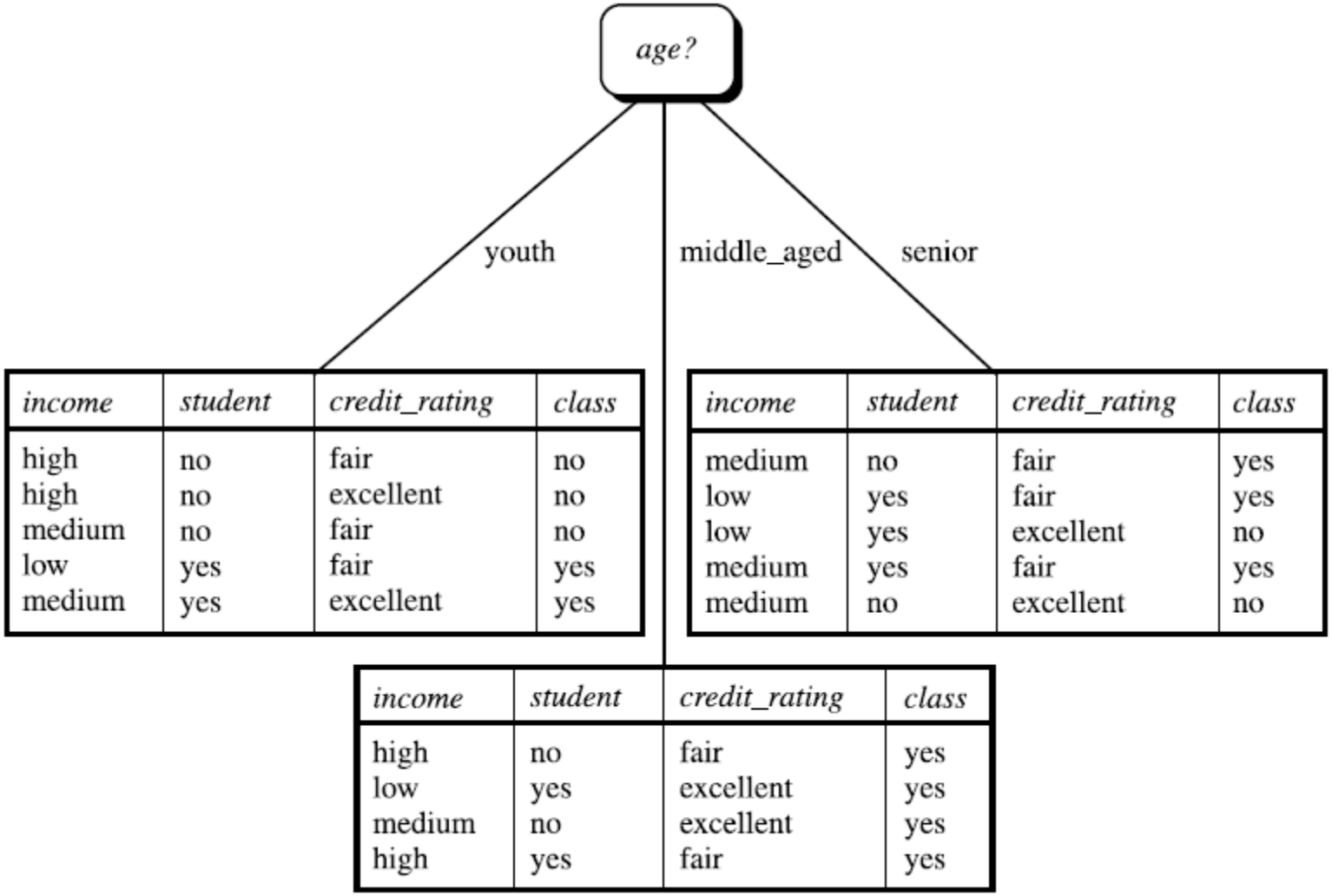
$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

$$\begin{aligned} Info_{age}(D) = & \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ & + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) \\ & + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ = & 0.694 \text{ bits.} \end{aligned}$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Gain(income)=0.029, Gain(student)=0.151,
Gain(credit_rating)=0.048

➔Age




| *income* | *student* | *credit_rating* | *class* |
|---|---|---|---|
| high | no | fair | no |
| high | no | excellent | no |
| medium | no | fair | no |
| low | yes | fair | yes |
| medium | yes | excellent | yes |

| *income* | *student* | *credit_rating* | *class* |
|---|---|---|---|
| medium | no | fair | yes |
| low | yes | fair | yes |
| low | yes | excellent | no |
| medium | yes | fair | yes |
| medium | no | excellent | no |

| *income* | *student* | *credit_rating* | *class* |
|---|---|---|---|
| high | no | fair | yes |
| low | yes | excellent | yes |
| medium | no | excellent | yes |
| high | yes | fair | yes |

# Gain ratio

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}.$$

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right).$$

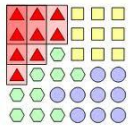
choose maximum gain ratio

- Note, however, that as the split information approaches 0, the ratio becomes unstable.

**So a constraint is added to avoid this**:

information gain of the test selected must be large—at least as great as the average gain over all tests examined.

# Example



A = income (low, medium, high)

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right).$$

$$= 0.926.$$

➔ GainRatio(income)= 0.029/0.926 = 0.032

# Gini Index

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$P_i$ is the probability that a tuple in $D$ belongs to class $Ci$

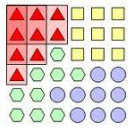
➔ $P_i = |C_{i,D}| / |D|$

- A is discrete-valued
  - $A : \{a1, a2, : : : , av\}$
  - and $2^v - 2$ possible

Ex: income {low, medium, high}

➔ {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}

# Gini Index

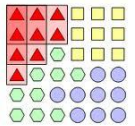


Ex: if a binary split on *A* partitions *D* into *D*1 and *D*2, the gini index of *D* given that partitioning is

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2).$$

➔ **Choose the subset that gives the minimum gini index as splitting subset.**

# Gini Index



- A is continuous-valued
  - Each possible split point must be considered
  - *D*1 is the set of tuples in *D* satisfying

  *A* <= *split point*
  - *D*2 is the set of tuples in *D* satisfying

  *A* > *split point*
- The reduction in impurity

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

➔ Attribute that maximizes the reduction in impurity

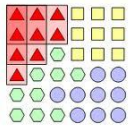


**Table 6.1** Class-labeled training tuples from the *AllElectronics* customer database.

| *RID* | *age* | *income* | *student* | *credit_rating* | *Class: buys_computer* |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Example

Yes No

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

$$Gini_{income \in \{low,medium\}}(D)$$

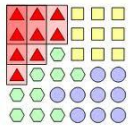
$$= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right)$$

$$= 0.450$$

$$= Gini_{income \in \{high\}}(D).$$

# Biased

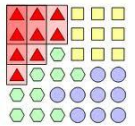


- Information gain is biased toward multivalued attributes.

- Gain ratio tends to prefer unbalanced splits in which one partition is much smaller than the others.

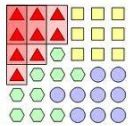
- Gini index is biased toward multivalued and has difficulty when the number of classes is large.

# Tree Pruning



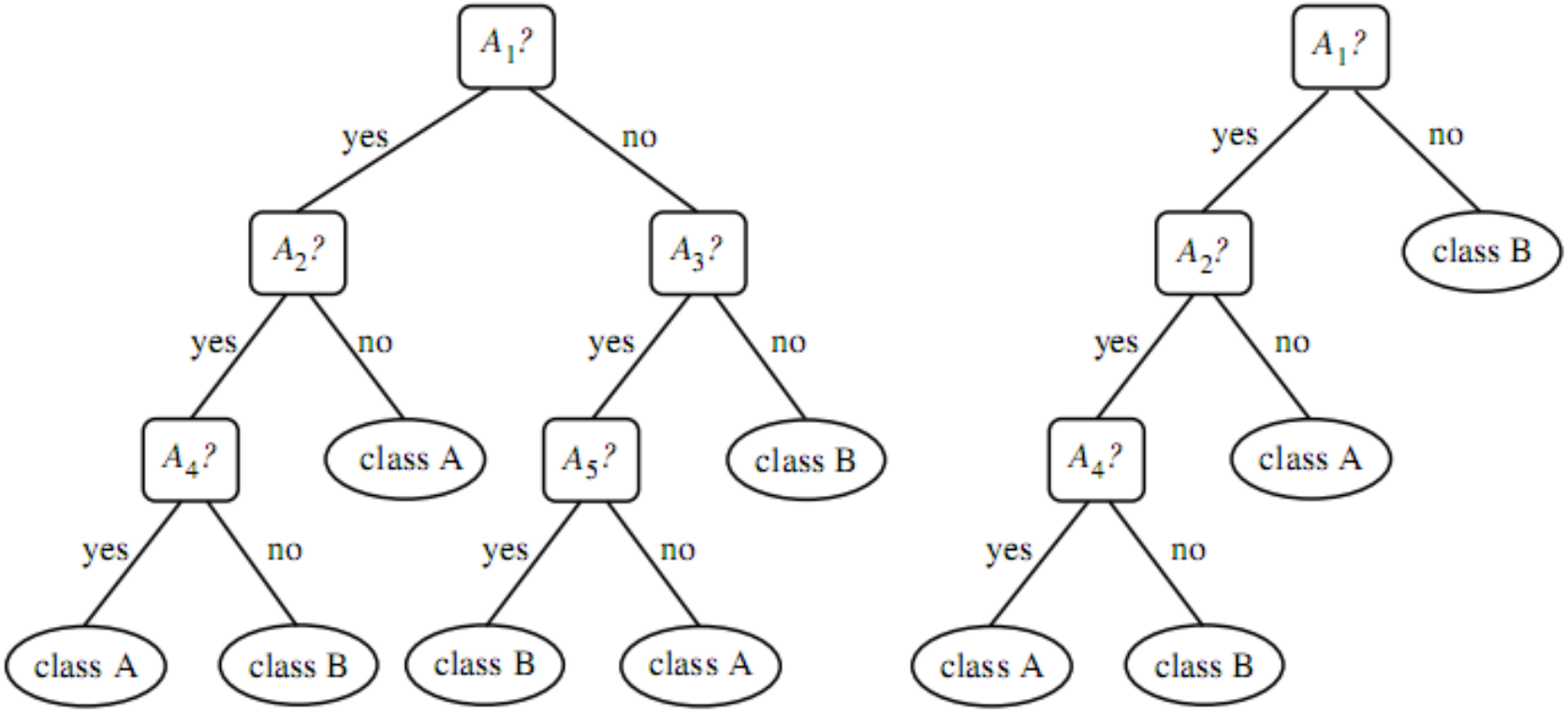
- When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers. Tree pruning methods address this problem of *overfitting* the data.
- There are two common approaches for tree pruning
  - Pre-pruning (early stopping rule):
    - The tree is "pruned" by halting its construction early
    - Or stop algorithm before it becomes a fully grown tree
    - Most popular test: chi-squared test

# Tree Pruning



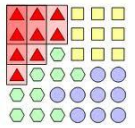
- Post-pruning
  - removes sub-trees from a "fully grown" tree: get a sequence of progressively pruned trees
  - Possible strategies: error estimation, significance testing, MDL pruning
  - preferred in practice

# Example..




An unpruned decision tree and a pruned version of it.

# Rule Based Classification



- has learned model as a set of IF-THEN rules
- We need to
  - How to generate the model
  - Examine how to use model to classify data
- IF-THEN rule is an expression of the form

  IF condition THEN conclusion

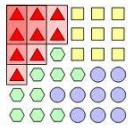
Ex:

R1:IF age=youth and student=yes THEN buys_computer=yes

Or

R1: (age=youth)Λ(student=yes)=>(buys_computer=yes)

# Using IF-THEN rules for classification (1/7)



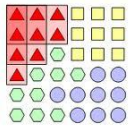
- IF part is called rule antecedent or precondition, it can consist of one or more attributes test
- THEN part is called rule consequent, it consist a class prediction
- A rule R can be assessed by its coverage and accuracy
  - Given a tuple X from a data D
  - Let $n_{cover}$: # of tuples covered by R
  - $n_{correct}$: # of tuples correctly classify by R
  - |D|: # of tuples in D

$$coverage(R) = \frac{n_{covers}}{|D|}$$

➔

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

# Using IF-THEN rules for classification (2/7)

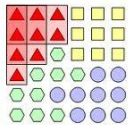


- Ex: of assessing R

**Table 6.1** Class-labeled training tuples from the *AllElectronics* customer database.

| *RID* | *age* | *income* | *student* | *credit_rating* | *Class: buys_computer* |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Using IF-THEN rules for classification (3/7)



R: IF age=youth AND student=yes THEN buys_computer=yes

➔ |D| = 14

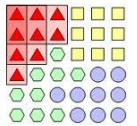
$n_{cover}$ = 2

$n_{correct}$ = 2

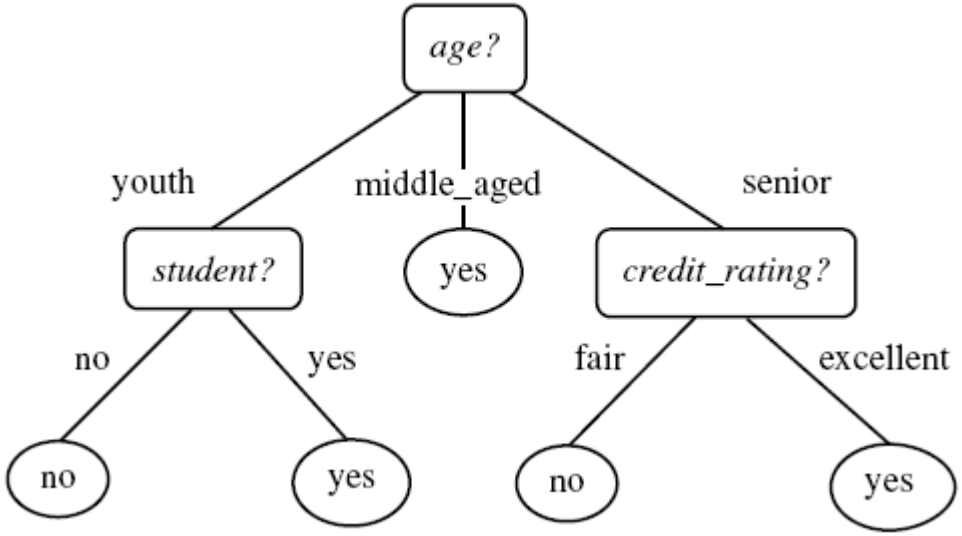
$$\text{coverage}(R) = \frac{2}{14} = 14.28\%$$

$$accuracy(R) = \frac{2}{2} = 100\%$$

# Rule Extraction from a Decision Tree

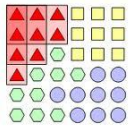


- One rule is created for each path from the root to a leaf node
- Each splitting criterion is logically AND to form the rule antecedent (IF part)
- Leaf node holds the class prediction for rule consequent (THEN part)
- Logical OR is implied between each of the extracted rules



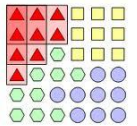

R1: IF *age* = *youth* AND *student* = *no* THEN *buys_computer* = *no*
R2: IF *age* = *youth* AND *student* = *yes* THEN *buys_computer* = *yes*
R3: IF *age* = *middle_aged* THEN *buys_computer* = *yes*
R4: IF *age* = *senior* AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*
R5: IF *age* = *senior* AND *credit_rating* = *fair* THEN *buys_computer* = *no*

# Bayesian classification



- are statistical classifiers
- based on Baye's theorem
- Simple Bayesian classifier called naïve Bayesian classifier
- the effect of an attribute value on a given class is independent of the values of the other attributes: this assumption is called class conditional independence

# Baye's theorem



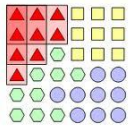
- Is name after Thomas Bayes, nonconformist English clergyman who did early work in probability and decision theory during the 18$^{th}$ century
- Let X : a data tuple, evidence, measure on n attributes.

  H: some hypothesis such as that X $\in$ class C

$\Rightarrow$We want to calculate

P(H|X): prob that hypothesis H holds given evidence X or prob that X $\in$ C

# Baye's theorem



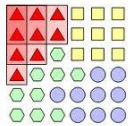
- **P(H|X)**: is a posterior prob or posteriori prob condition on X

Ex: X=(age=35, income=$40,000)

H: hypothesis that our customer will buy computer

=> P(H|X): prob that customer X will buy computer giving that we know their age and income

# Baye's theorem



- Baye's theorem

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

# Naïve Bayesian Classification

- Works as follow

1. From data set D
   - Associated class label
   - n dimensional att vector X=(x1,x2,x3,…,xn), depiction n measurement made on the tuple from n atts. A1, A2, A3,…, An
2. Suppose we have m classes c1, c2, …,cm

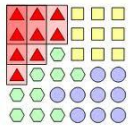
Giving tuple X, classifier will predict X belong to highest posterior probability, condition on X.

X $\in$ Ci iif P(Ci|X) > P (Cj|X) for 1 <= j <=m, j $\neq$ I

Ci, for which P(Ci|X) is maximized is called maximum posterior hypothesis;

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

# Naïve Bayesian Classification



3. P(X) is constant for all classes

$\Rightarrow$ Maximize $P(X|C_i)P(C_i)$

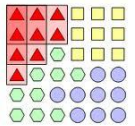
If class prior prob are not know, commonly assumed that $P(C_1)=P(C_2)=\ldots=P(C_m)$

=>maximize $P(X|C_i)$

Else maximize P(X|Ci)P(Ci)

$$P(Ci) = \frac{|C_{i,D}|}{|D|}$$

# Naïve Bayesian Classification



4. Calculate P(X|Ci) is extremely expensive

Naïve assumes class conditional independence is made.

=>

$$P(X \mid Ci) = \prod_{k=1}^{n} (x_k \mid Ci)$$

$$= P(x_1 \mid Ci).P(x_2 \mid Ci)...P(x_n \mid Ci)$$

Where $x_k$ is the value of att. $A_k$ for X

If A is category

$$\Rightarrow P(x_k \mid C_i) = \frac{\#of\_tuple\_of\_class\_Ci\_inD\_have\_value\_Xk}{\mid C_{i,D} \mid}$$

# Example

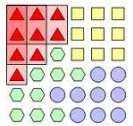
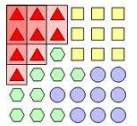
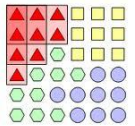


**Table 6.1** Class-labeled training tuples from the *AllElectronics* customer database.

| *RID* | *age* | *income* | *student* | *credit_rating* | *Class: buys_computer* |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |



- A={age, income, student, credit_rating}
- Class label buy_computer = Yes|No
- C1: buy_computer = Yes
- C2: buy_computer = No
- X=(age=youth, income=medium, student=y, credit-rating=fair)
- We need to maximize P(X|Ci)P(Ci)



$$P(buys\_computer = yes) = 9/14 = 0.643$$

$$P(buys\_computer = no) = 5/14 = 0.357$$

To compute $PX|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$

$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$$

$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$

$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$$

$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$

$$P(student = yes \mid buys\_computer = no) = 1/5 = 0.200$$

$$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$$

$$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.400$$

Using the above probabilities, we obtain

$$P(X|buys\_computer = yes) = P(age = youth \mid buys\_computer = yes) \times$$
$$P(income = medium \mid buys\_computer = yes) \times$$
$$P(student = yes \mid buys\_computer = yes) \times$$
$$P(credit\_rating = fair \mid buys\_computer = yes)$$
$$= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044.$$

Similarly,

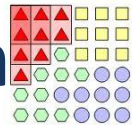
$$P(X|buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

To find the class, $C_i$, that maximizes $P(X|C_i)P(C_i)$, we compute

$$P(X|buys\_computer = yes)P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$
$$P(X|buys\_computer = no)P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts $buys\_computer = yes$ for tuple $X$.

# Advantages and Disadvantages Naïve Bayesian Classification



Advantages:

- Easy to implement
- Obtain good results in most of the cases

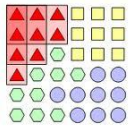
Disadvantages

- Assumption of class conditional independence usually doesn't hold
- Dependencies among these cannot be modeled by this classifier

How to deal with these Dependencies?

- Bayesian Belief Networks

# Bayesian networks



- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions

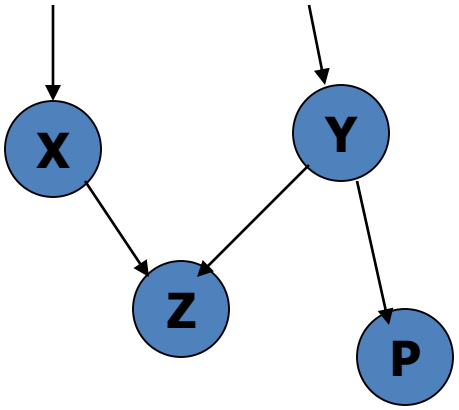
- Syntax:
  - a set of nodes, one per variable
  - a directed, acyclic graph (link ≈ "directly influences")
  - a conditional distribution for each node given its parents:

$$\mathbf{P}(X_i \mid \text{Parents}(X_i))$$

- In the simplest case, conditional distribution represented as a **conditional probability table** (CPT) giving the distribution over $X_i$ for each combination of parent values

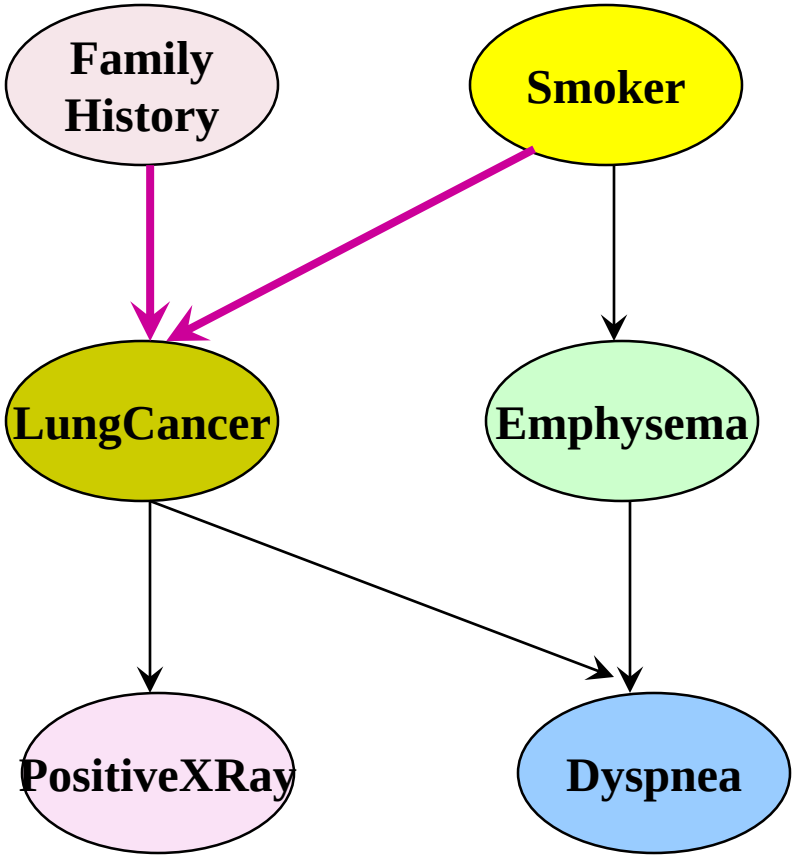
# Bayesian Networks

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution




- ❑Nodes: random variables
- ❑Links: dependency
- ❑X,Y are the parents of Z, and Y is the parent of P
- ❑No dependency between Z and P
- ❑Has no loops or cycles

# Bayesian Belief Network: An Example



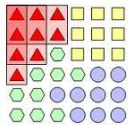

Bayesian Belief Networks

| | (FH, S) | (FH, ~S) | (~FH, S) | (~FH, ~S) |
|---|---|---|---|---|
| LC | 0.8 | 0.5 | 0.7 | 0.1 |
| ~LC | 0.2 | 0.5 | 0.3 | 0.9 |

The **conditional probability table (CPT)** for the variable LungCancer:
Shows the conditional probability for each possible combination of its parents

$$P(z1,...,zn) = \prod_{i=1}^{n} P(z_i \mid Parents(Z_i))$$

# Example



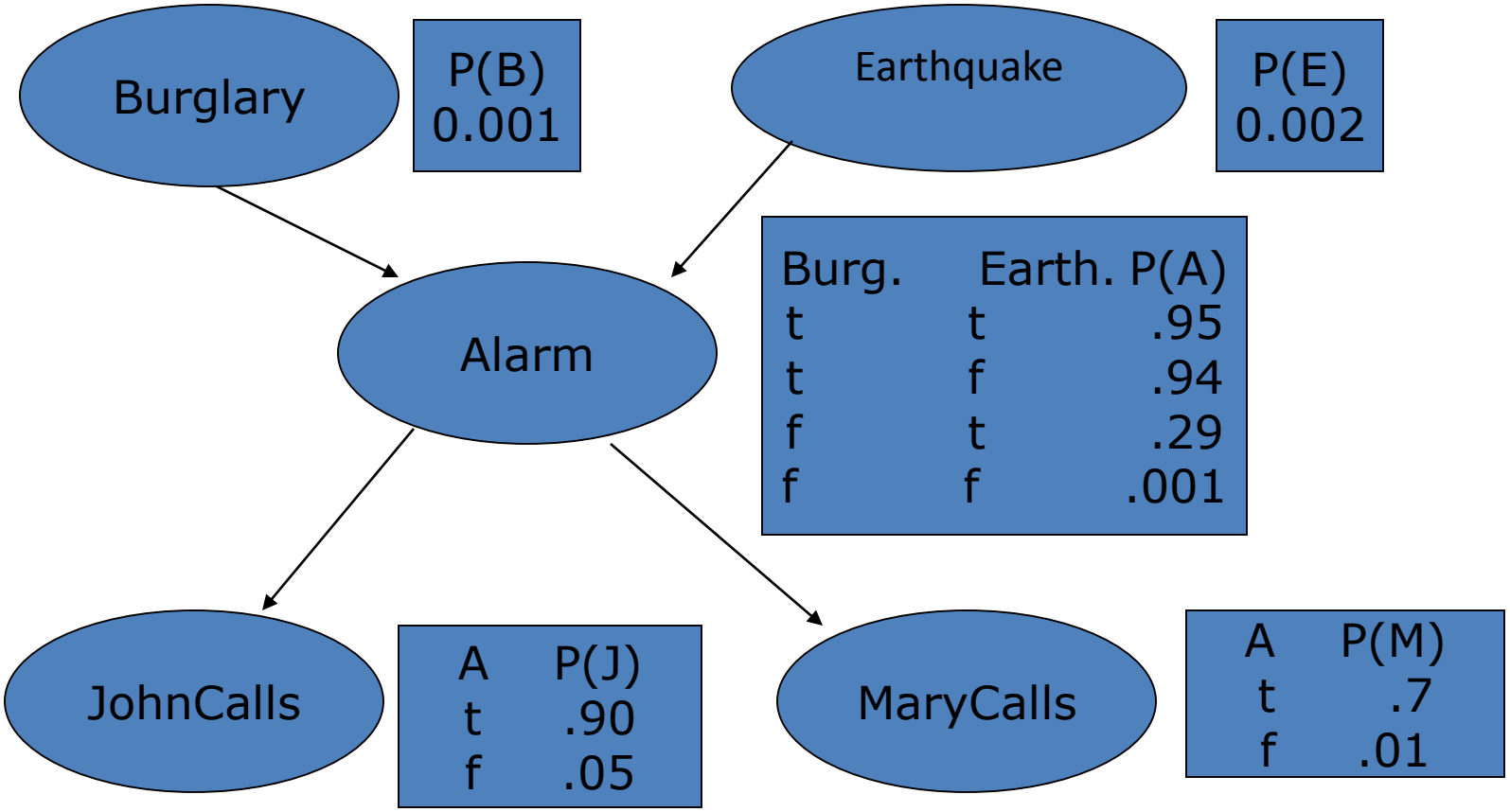
- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

- Network topology reflects "causal" knowledge:

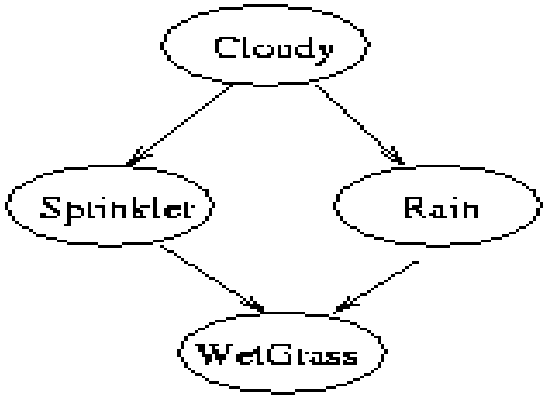
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call

# Belief Networks



# BAYESIAN BELIEF NETWORK

| P(C=F) | P(C=T) |
|---|---|
| 0.5 | 0.5 |

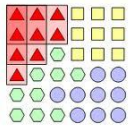



| C | P(S=F) | P(S=T) |
|---|---|---|
| F | 0.5 | 0.5 |
| T | 0.9 | 0.1 |

| C | P(R=F) | P(R=T) |
|---|---|---|
| F | 0.8 | 0.2 |
| T | 0.2 | 0.8 |

| S | R | P(W=F) | P(W=T) |
|---|---|---|---|
| F | F | 1.0 | 0.0 |
| T | F | 0.1 | 0.9 |
| F | T | 0.1 | 0.9 |
| T | T | 0.01 | 0.99 |

# BAYESIAN BELIEF NETWORK



- By the chaining rule of probability, the joint probability of all the nodes in the graph above is:

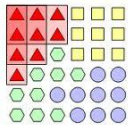
$P(C, S, R, W) = P(C) * P(S|C) * P(R|C) * P(W|S,R)$

W=Wet Grass, C=Cloudy, R=Rain, S=Sprinkler

Example: $P(W \cap -R \cap S \cap C)$

$= P(W|S,-R)*P(-R|C)*P(S|C)*P(C)$

$= 0.9*0.2*0.1*0.5 = 0.009$



# BAYESIAN BELIEF NETWORK

What is the probability of wet grass on a given day - P(W)?

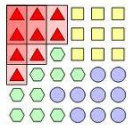
$$P(W) = P(W|SR) * P(S) * P(R) +$$
$$P(W|S\text{-}R) * P(S) * P(\text{-}R) +$$
$$P(W|\text{-}SR) * P(\text{-}S) * P(R) +$$
$$P(W|\text{-}S\text{-}R) * P(\text{-}S) * P(\text{-}R)$$

Here $P(S) = P(S|C) * P(C) + P(S|\text{-}C) * P(\text{-}C)$

$P(R) = P(R|C) * P(C) + P(R|\text{-}C) * P(\text{-}C)$

$P(W) = 0.5985$

# Fuzzy Set

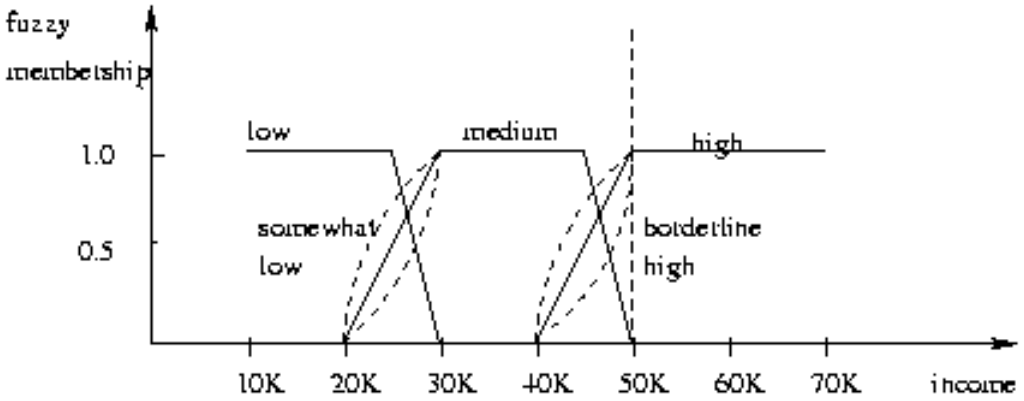


- Rule-based (Disadvantage): sharp cutoffs for continuous attributes.

$$IF\ (years\_employed \geq 2)\ AND\ (income \geq 50K)\ THEN\ credit = approved.$$

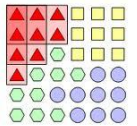
- What about income 49K or 48K?
- Solution: fuzzy set
- Discretise income: {low_income, medium_income, high_income}
- fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership that a certain value has in a given category

# Fuzzy Set Approache



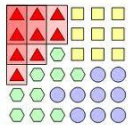

- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph)
- Attribute values are converted to fuzzy values
  - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined

# What Is Prediction?



- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

# Linear Regression



- Linear regression: involves a response variable y and a single predictor variable x

  $y = w_0 + w_1 x$

  where $w_0$ (y-intercept) and $w_1$ (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line

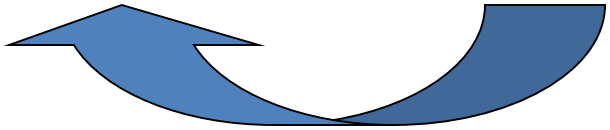
$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form $(\mathbf{X_1}, y_1)$, $(\mathbf{X_2}, y_2)$,…, $(\mathbf{X_{|D|}}, y_{|D|})$
  - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method or using SAS, S-Plus
  - Many nonlinear functions can be transformed into the above

# Linear Regression

A regression model is comprised of a dependent, or response, variable and an independent, or predictor, variable.

Dependent Variable = Independent Variable(s)

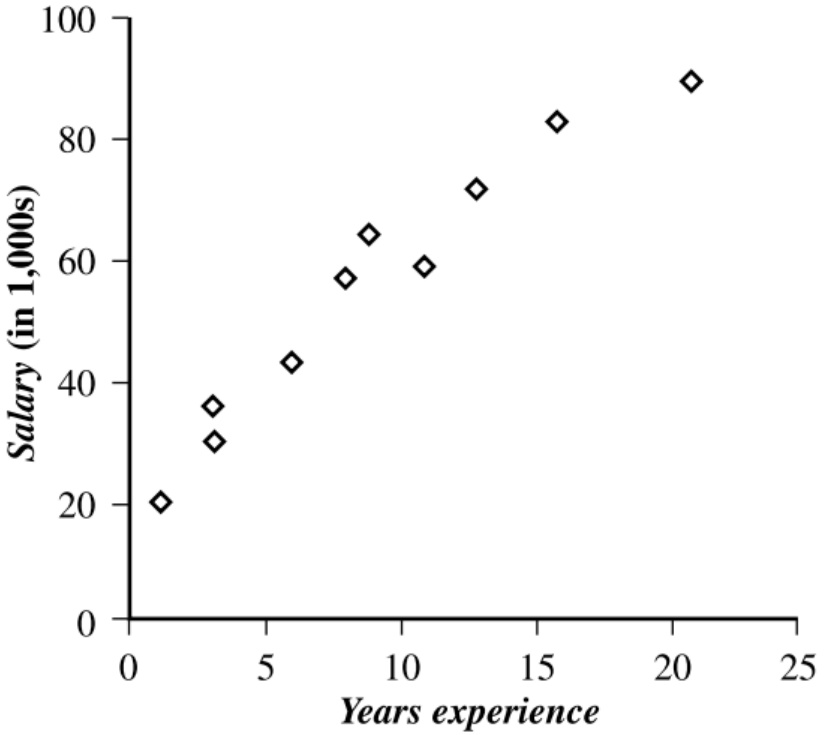


Prediction Relationship

# Linear Regression

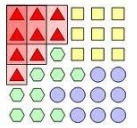
Salary data.

| *x years experience* | *y salary (in $1000s)* |
|---|---|
| 3 | 30 |
| 8 | 57 |
| 9 | 64 |
| 13 | 72 |
| 3 | 36 |
| 6 | 43 |
| 11 | 59 |
| 21 | 90 |
| 1 | 20 |
| 16 | 83 |




Plot of data: Although the points do not fall on a straight line, the overall pattern suggests a linear relationship between x(years experience) and y (salary)

# Linear Regression



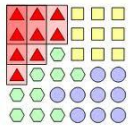
Given the above data, we compute $\bar{x} = 9.1$ and $\bar{y} = 55.4$. Substituting these values into Equations (6.50) and (6.51), we get

$$w_1 = \frac{(3-9.1)(30-55.4)+(8-9.1)(57-55.4)+\cdots+(16-9.1)(83-55.4)}{(3-9.1)^2+(8-9.1)^2+\cdots+(16-9.1)^2} = 3.5$$

$$w_0 = 55.4-(3.5)(9.1) = 23.6$$

Thus, the equation of the least squares line is estimated by $y = 23.6 + 3.5x$. Using this equation, we can predict that the salary of a college graduate with, say, 10 years of experience is \$58,600. ■

# Nonlinear Regression



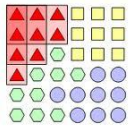
- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

convertible to linear with new variables: $x_2 = x^2$, $x_3 = x^3$

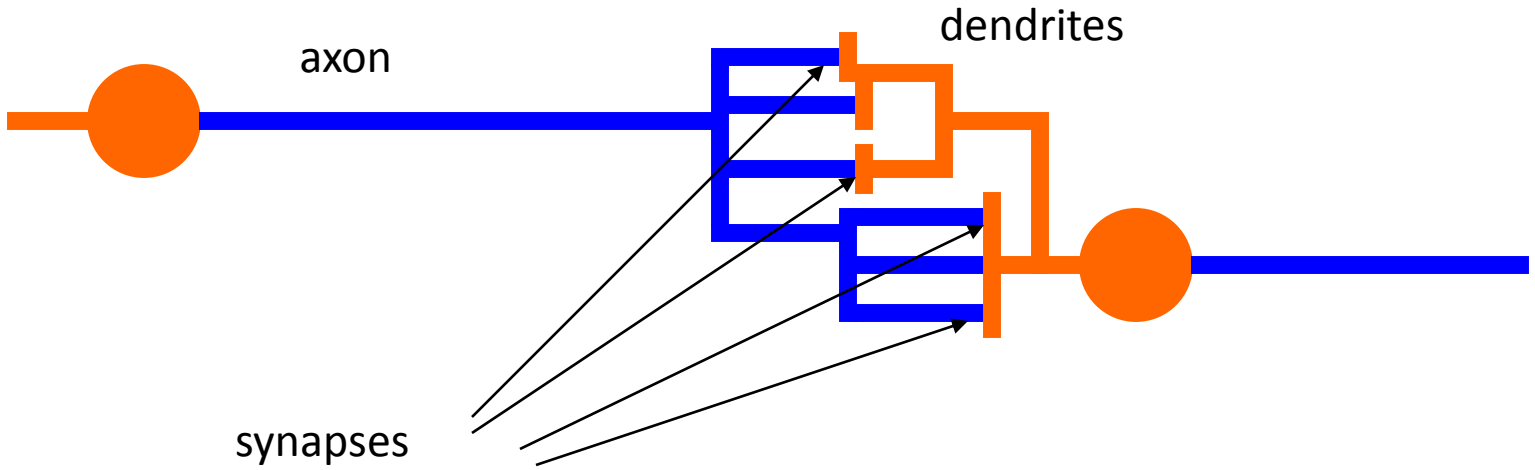
$$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$$

- Which is easily solved by the method of least squares using software for regression analysis
- Note that polynomial regression is a special case of multiple regression



# Neural Network

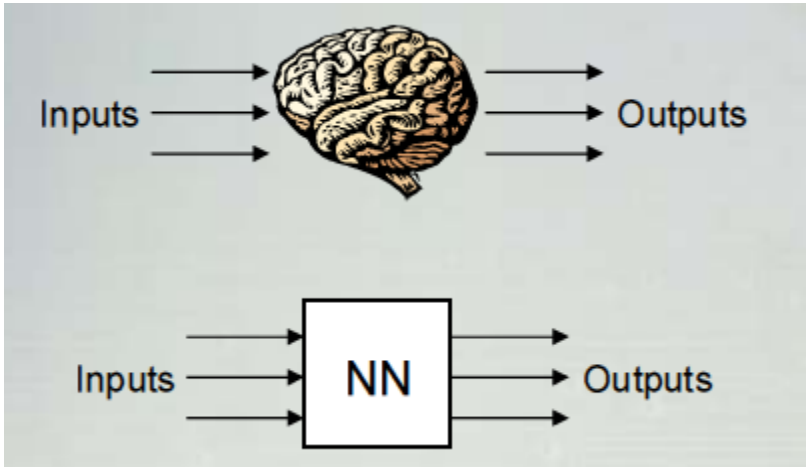
# Biological inspiration




The information transmission happens at the synapses.

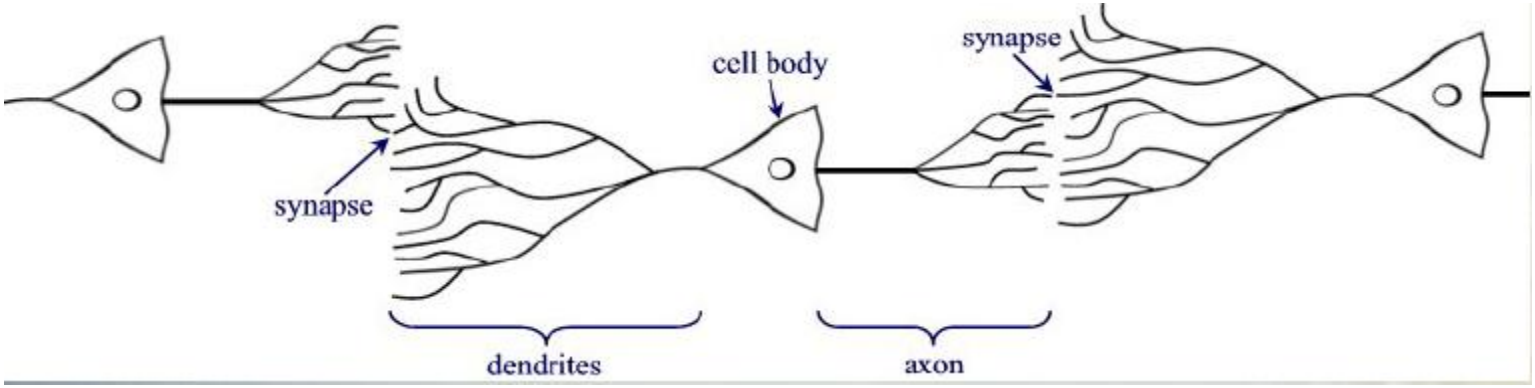
## What Are Artificial Neural Networks?

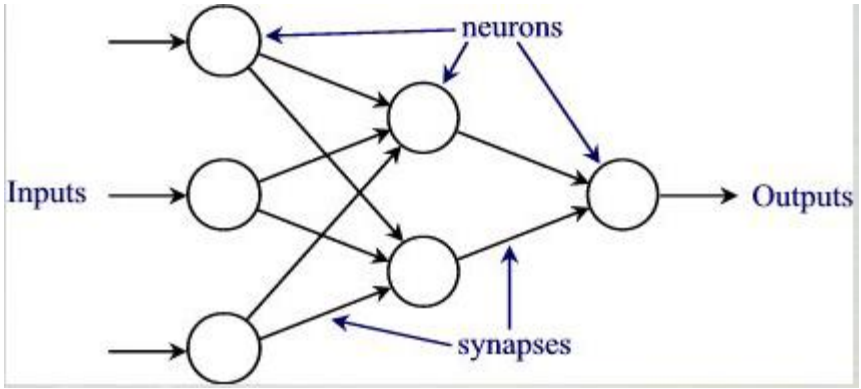
- An extremely simplified model of the brain
- Essentially a function approximator
  - ► Transforms inputs into outputs to the best of its ability



## What Are Artificial Neural Networks?

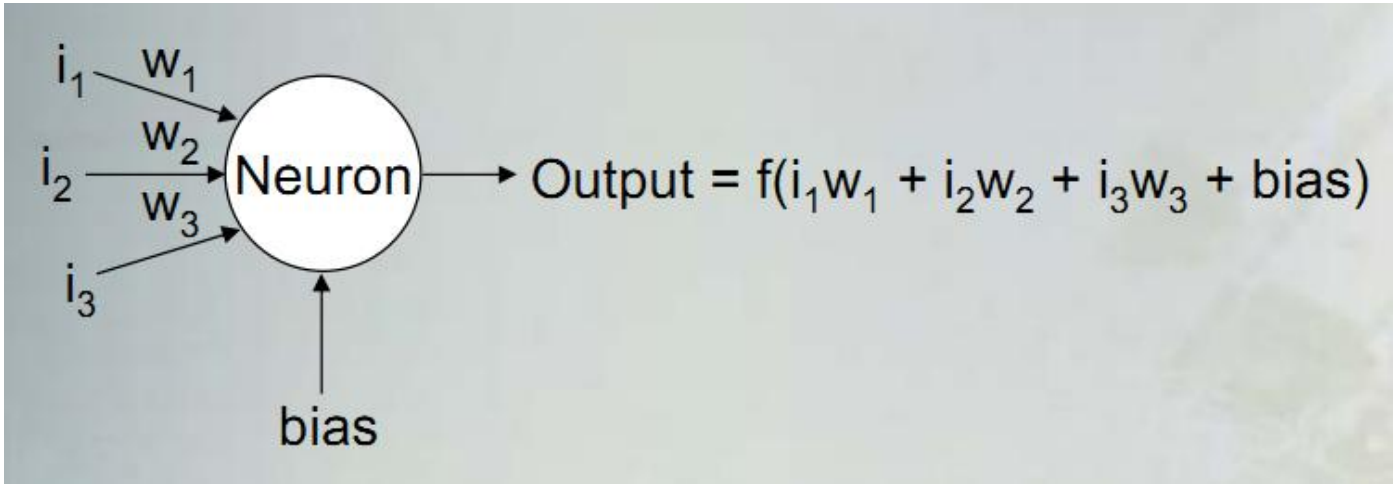
- Composed of many "neurons" that co-operate to perform the desired function



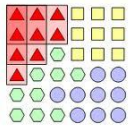
**How Neural Network Works?**

● The output of a neuron is a function of the weighted sum of the inputs plus a bias




• The function of the entire neural network is simply the computation of the outputs of all the neurons

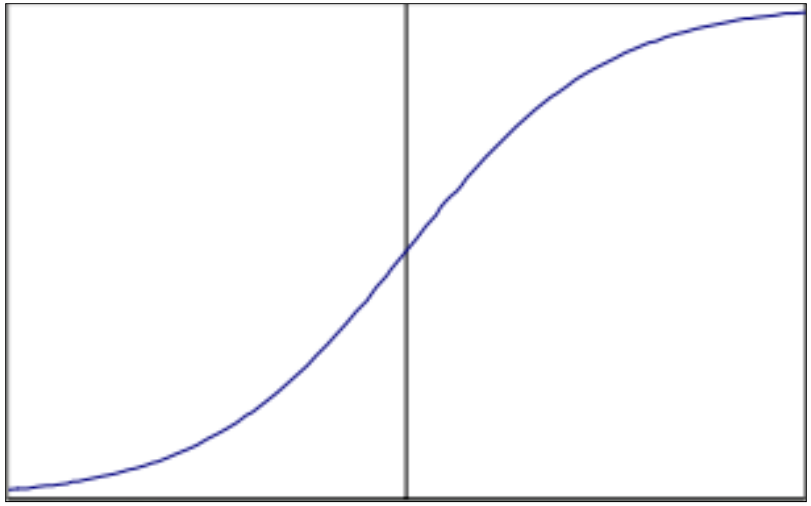
► An entirely deterministic calculation

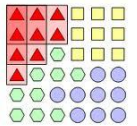


**What is Activation function?**

Applied to the weighted sum of the inputs of a neuron to produce the output

- Majority of NN's use sigmoid functions
  - ► Smooth, continuous, and monotonically increasing (derivative is always positive)
  - ► Bounded range - but never reaches max or min
- ■ Consider "ON" to be slightly less than the max and "OFF" to be slightly greater than the min





What is Activation function?

• The most common sigmoid function used is the logistic function

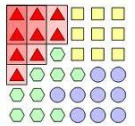
- ► f(x) = 1/(1 + e-x)
- ► The calculation of derivatives are important for neural networks and the logistic function has a very nice derivative

■ f'(x) = f(x)(1 - f(x))

● Other sigmoid functions also used

- ► hyperbolic tangent
- ► arctangent

● The exact nature of the function has little effect on the abilities of the neural network



The weights in a neural network are the most important factor in determining its function

- Training is the act of presenting the network with some sample data and modifying the weights to better approximate the desired function
- There are two main types of training

Supervised Training

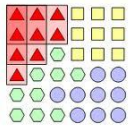
- Supplies the neural network with inputs and the desired outputs
- Response of the network to the inputs is measured

The weights are modified to reduce the difference between the actual and desired outputs

Unsupervised Training

- Only supplies inputs
- The neural network adjusts its own weights so that similar inputs cause similar outputs

The network identifies the patterns and differences in the inputs without any external assistance



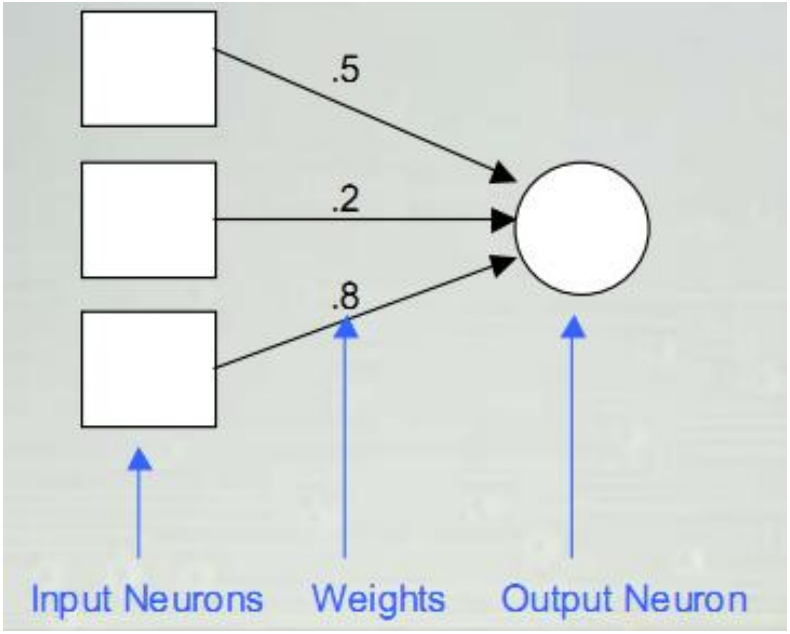
**Epoch**

- **One iteration through the process of providing the network with an input and updating the network's weights**
- **Typically many epochs are required to train the neural network**

# Perceptrons

**First neural network with the ability to learn**

- **Made up of only input neurons and output neurons**
- **Input neurons typically have two states: ON and OFF**
- **Output neurons use a simple threshold activation function**
- **In basic form, can only solve linear problems**
  - ► **Limited applications**

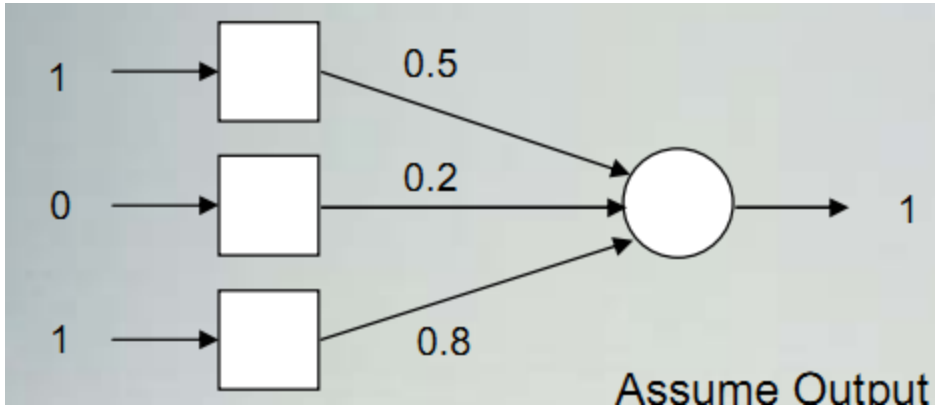


# How Do Perceptrons Learn?

**Uses supervised training**

- **If the output is not correct, the weights are adjusted according to the formula:**

$w_{new} = w_{old} + \alpha(desired - output)*input$ $\alpha$ is the learning rate

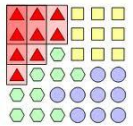



1 * 0.5 + 0 * 0.2 + 1 * 0.8 = 1.3
Assuming Output Threshold = 1.2
1.3 > 1.2

Assume $\alpha = 1$

W1new = 0.5 + 1*(0-1)*1 = -0.5
W2new = 0.2 + 1*(0-1)*0 = 0.2
W = 0.8 + 1*(0-1)*1 = -0.2

# Multilayer Feed Forward Networks



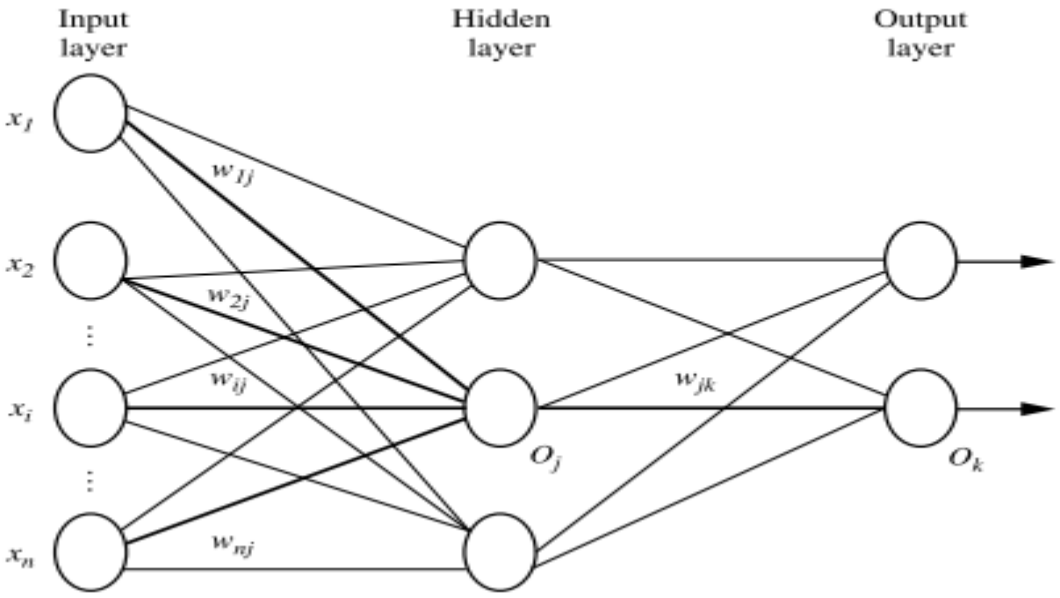
- Most common neural network
  - An extension of the perceptron
    - Multiple layers
- The addition of one or more "hidden" layers in between the input and output layers
  - Activation function is not simply a threshold
- Usually a sigmoid function
  - A general function approximate
- Not limited to linear problems
  - Information flows in one direction
    - The outputs of one layer act as inputs to the next layer

# Multilayer Feed-Forward NN

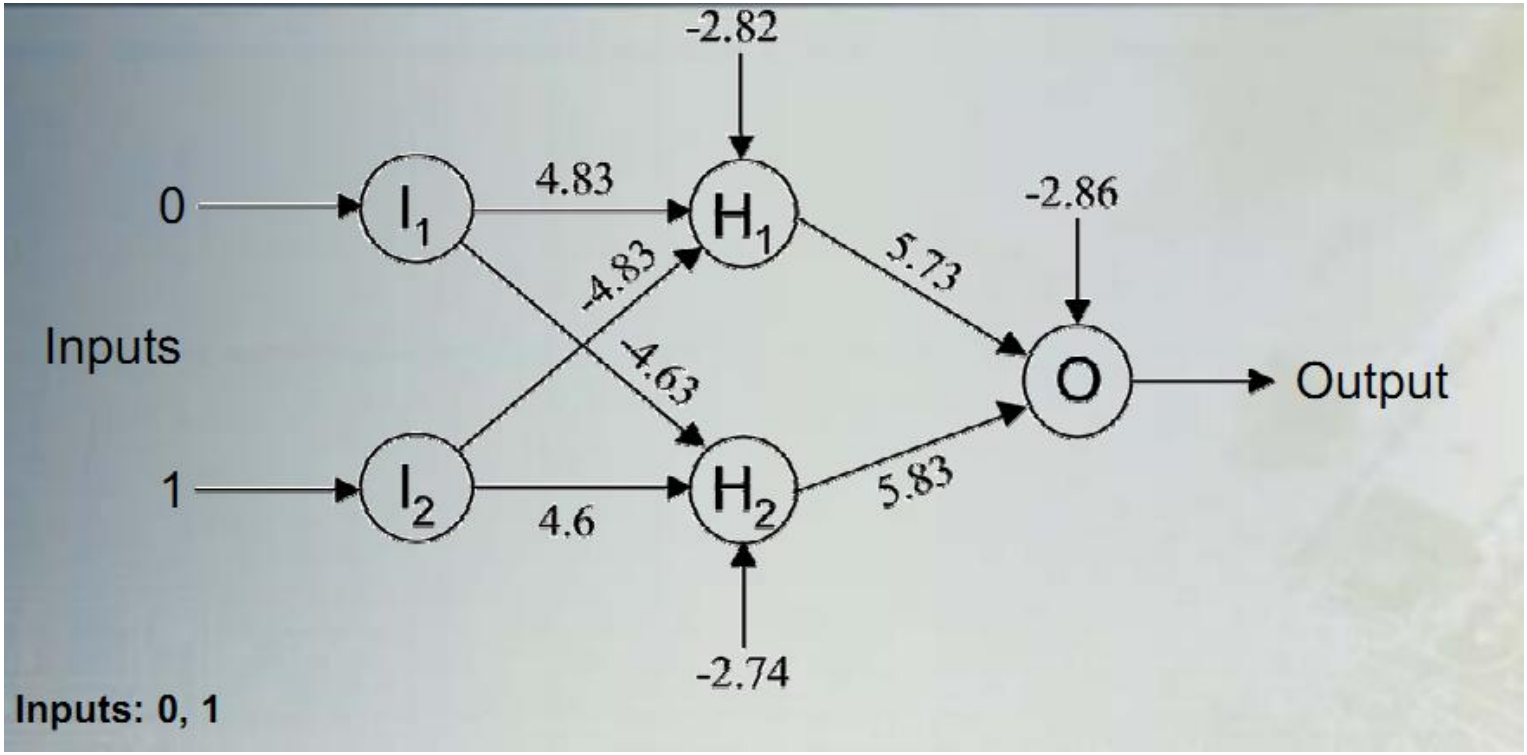
The backpropagation algorithm performs learning on a *multilayer feed-forward* neural network. It iteratively learns a set of weights for prediction of the class label of tuples. A **multilayer feed-forward** neural network consists of an *input layer*, one or more *hidden layers*, and an *output layer*. An example of a multilayer feed-forward network is shown in Figure 6.15.

Each layer is made up of units. The inputs to the network correspond to the attributes measured for each training tuple. The inputs are fed simultaneously into the units making up the **input layer**. These inputs pass through the input layer and are then weighted and fed simultaneously to a second layer of "neuronlike" units, known as a **hidden layer**. The outputs of the hidden layer units can be input to another hidden layer, and so on. The number of hidden layers is arbitrary, although in practice, usually only one is used. The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction for given tuples.

The units in the input layer are called **input units**. The units in the hidden layers and output layer are sometimes referred to as **neurodes**, due to their symbolic biological basis, or as **output units**. The multilayer neural network shown in Figure 6.15 has two layers



# XOR Example




**$H_1$:** Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65
Output = $1 / (1 + e^{7.65}) = 4.758 \times 10^{-4}$

**$H_2$:** Net = 0(-4.63) + 1(4.6) – 2.74 = 1.86
Output = $1 / (1 + e^{-1.86}) = 0.8652$

**O:** Net = $4.758 \times 10^{-4}(5.73) + 0.8652(5.83) – 2.86 = 2.187$
Output = $1 / (1 + e^{-2.187}) = 0.8991 \equiv$ "1"

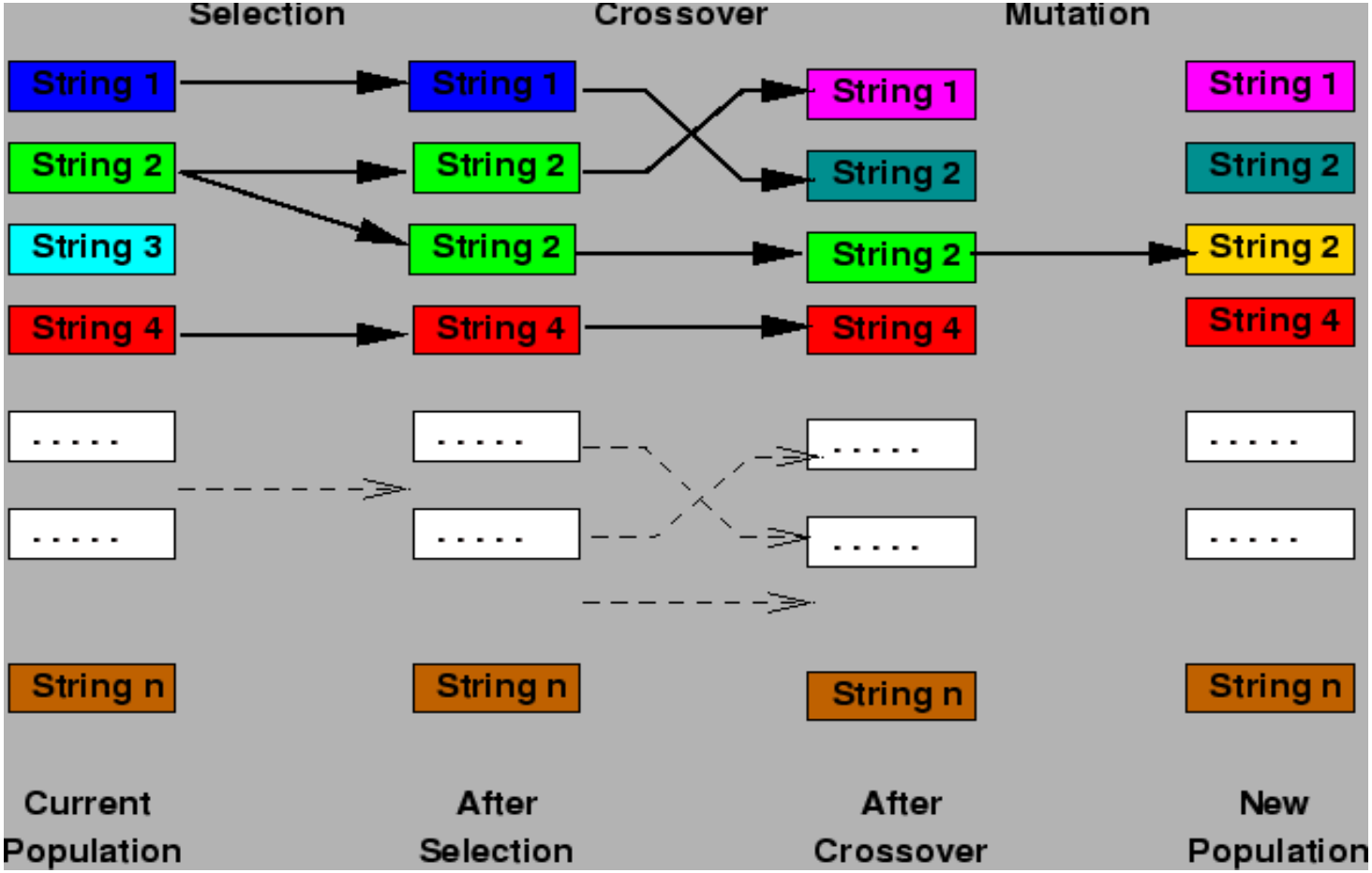
# The Canonical Genetic Algorithm

Genetic Algorithms are search algorithms that are based on concepts of natural selection and natural genetics.

```
                              /*Algorithm GA */
formulate initial population
randomly initialize population
repeat
      evaluate objective function
      find fitness function
      apply genetic operators
            reproduction
            crossover
            mutation
until stopping criteria
```

**Fig. The Working Principle of a Simple Genetic**

# The Canonical Genetic Algorithm




*The basic GA operations: One generation is broken down into a selection phase and recombination phase. Strings are assigned into adjacent slots during selection.*

# Crossover

A crossover operator is used to recombine two strings to get a better string. In crossover operation, recombination process creates different individuals in the successive generations by combining material from two individuals of the previous generation.

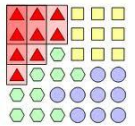
| | Before crossover | | After crossover |
|---|---|---|---|
| String 1 | 011\|**01100** | String 1 | 011\|*11001* |
| String 2 | 110\|*11001* | String 2 | 011\|**01100** |

Fig. One site crossover operation

| | Before crossover | | After crossover |
|---|---|---|---|
| String 1 | 011\|**011**\|00 | String 1 | 011\|*110*\|00 |
| String 2 | 110\|*110*\|01 | String 2 | 011\|**011**\|01 |

Two site crossover operation

# Mutation



Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima. It is an operator that introduces diversity in the population whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators. Mutation may cause the chromosomes of individuals to be different from those of their parent individuals.

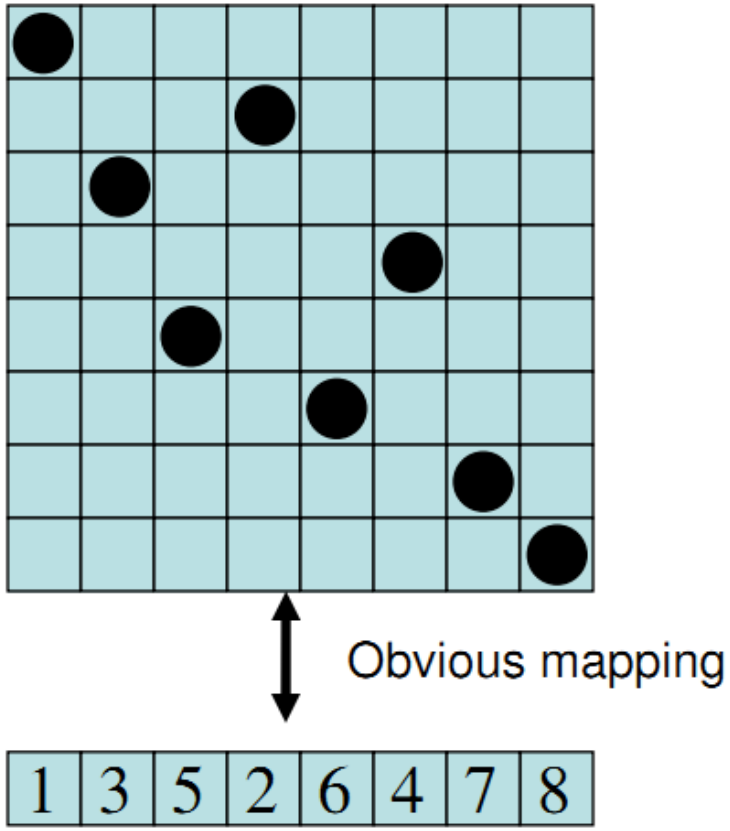
01101011
00111101
00010110
01111100

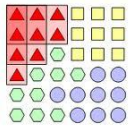
It can be noticed that all four strings have a in the left most bit position. If the true optimum solution requires in that position, then neither reproduction nor crossover operator described above will be able to create 1 in that position. The inclusion of mutation introduces probability of turning into .

# Example: the 8 queens problem

Solution:
a board configuration

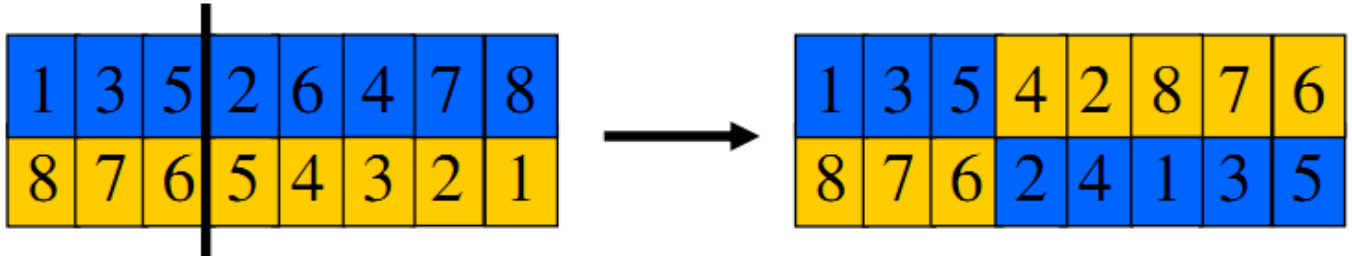
Chromosome:
a permutation of
the numbers 1 - 8

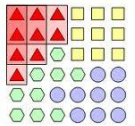




# Example: the 8 queens problem

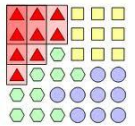
- Combining two permutations into two new permutations
  - Choose random crossover point
  - Copy first parts into children
  - Create second part by inserting values from other parent
    - In the order they appear there
    - Beginning after crossover point
    - Skipping values already in child





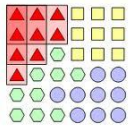
# Rough Sets Theory

# Introduction



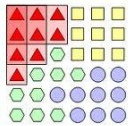
- Often, information on the surrounding world is
  - **Imprecise**
  - **Incomplete**
  - **uncertain**.
- We should be able to **process uncertain and/or incomplete information**.

# Introduction



- When dealing with inexact, uncertain, or vague knowledge, are the **fuzzy set** and the **rough set** theories.

# Introduction



- Fuzzy set theory
  - Introduced by Zadeh in 1965
  - Has demonstrated its usefulness in chemistry and in other disciplines
- Rough set theory
  - Introduced by Pawlak in 1985
  - Popular in many other disciplines

# Introduction

- Fuzzy Sets

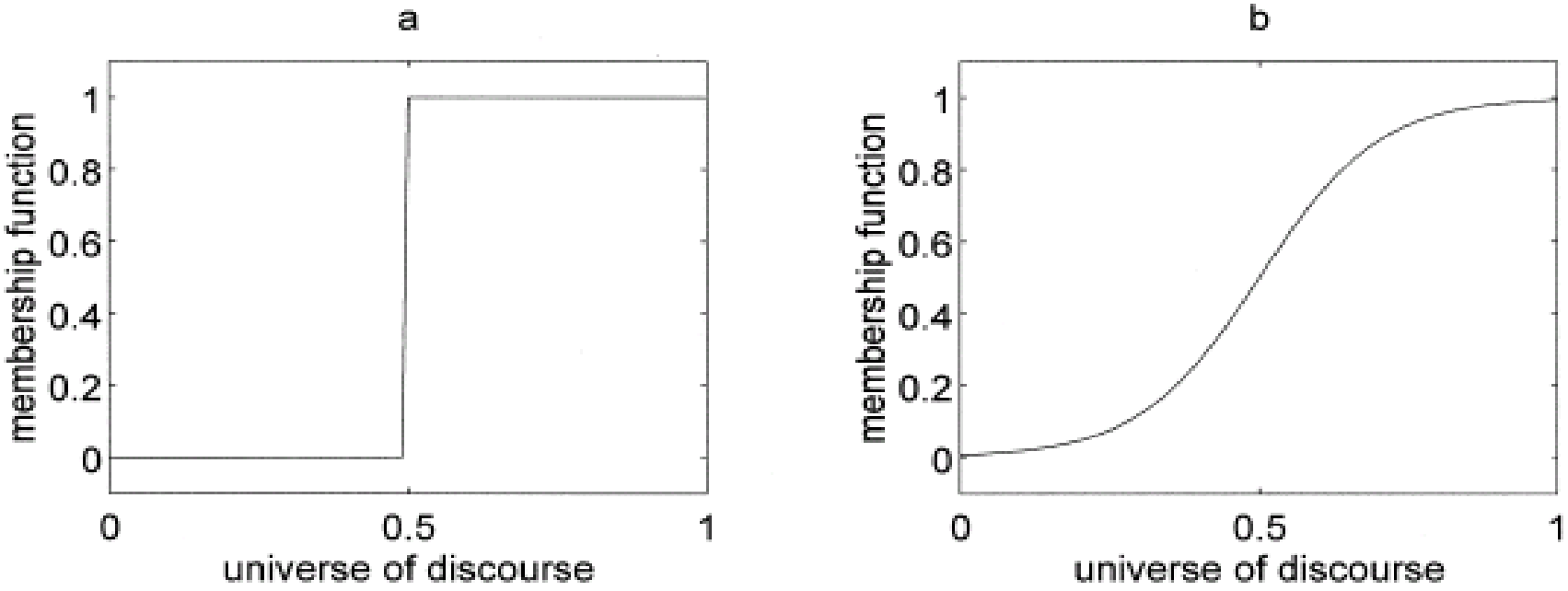
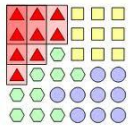


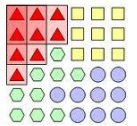

Fig. 1. Membership function according to (a) the crisp set theory and (b) the fuzzy set theory.

# Introduction



- Rough Sets
  - In the rough set theory, **membership is not** the primary concept.
  - Rough sets represent a different mathematical approach to vagueness and uncertainty.

# Basic concepts of the rough sets theory



- Information system
  - IS=($U$,$A$)
  - $U$ is the universe ( a finite set of objects, $U$={$x_1$,$x_2$,….., $x_m$}
  - $A$ is the set of attributes (features, variables)
  - $V_a$ is the set of values $a$, called the domain of attribute $a$.

# Basic concepts of the rough sets theory

Consider a data set containing the results of three measurements performed for 10 objects. The results can be organized in a matrix 10x3.

| | | |
|---|---|---|
| 2 | 1 | 3 |
| 3 | 2 | 1 |
| 2 | 1 | 3 |
| 2 | 2 | 3 |
| 1 | 1 | 4 |
| 1 | 1 | 2 |
| 3 | 2 | 1 |
| 1 | 1 | 4 |
| 2 | 1 | 3 |
| 3 | 2 | 1 |

# Basic concepts of the rough sets theory

- IS=(U,A)
- $U=\{x_1,x_2,x_3,x_4,x_5,x_6\ldots\ldots, x_{10}\}$
- $A=\{a_1,a_2,a_3\}$
- The domains of attributes are：

  $V_1=\{1,2,3\}$

  $V_2=\{1,2\}$

  $V_3=\{1,2,3,4\}$

# Basic concepts of the rough sets theory

For every set of attributes $B \subset A$, an *indiscernibility relation* $\mathrm{Ind}(B)$ is defined in the following way: two objects, $x_i$ and $x_j$, are indiscernible by the set of attributes $B$ in $A$, if $b(x_i) = b(x_j)$ for every $b \subset B$.

# Basic concepts of the rough sets theory

As one can easily notice, there are some identical objects in our data set. For instance, objects $x_1$ and $x_3$ cannot be distinguished based on the available data.

Table 1

| $U$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| $x_1$ | 2 | 1 | 3 |
| $x_2$ | 3 | 2 | 1 |
| $x_3$ | 2 | 1 | 3 |
| $x_4$ | 2 | 2 | 3 |
| $x_5$ | 1 | 1 | 4 |
| $x_6$ | 1 | 1 | 2 |
| $x_7$ | 3 | 2 | 1 |
| $x_8$ | 1 | 1 | 4 |
| $x_9$ | 2 | 1 | 3 |
| $x_{10}$ | 3 | 2 | 1 |

# Basic concepts of the rough sets theory

The notation $U/A$ means that we are considering elementary sets of the universe $U$ in the space $A$.

Table 2

| $U/A$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| $\{x_1, x_3, x_9\}$ | 2 | 1 | 3 |
| $\{x_2, x_7, x_{10}\}$ | 3 | 2 | 1 |
| $\{x_4\}$ | 2 | 2 | 3 |
| $\{x_5, x_8\}$ | 1 | 1 | 4 |
| $\{x_6\}$ | 1 | 1 | 2 |

# Basic concepts of the rough sets theory

It can happen that we are interested in the two attributes only, for instance in $a_1$ and $a_2$. Then the indiscernibility relation is limited to the subset $B = \{a_1, a_2\}$ and the resulting elementary sets are given in Table 3.

Table 3

| $U/B$ | $a_1$ | $a_2$ |
|---|---|---|
| $\{x_1, x_3, x_9\}$ | 2 | 1 |
| $\{x_2, x_7, x_{10}\}$ | 3 | 2 |
| $\{x_4\}$ | 2 | 2 |
| $\{x_5, x_6, x_8\}$ | 1 | 1 |

# Basic concepts of the rough sets theory

The rough sets approach to data analysis hinges on two basic concepts, namely the *lower* and the *upper approximations* of a set (Fig. 2), referring to:

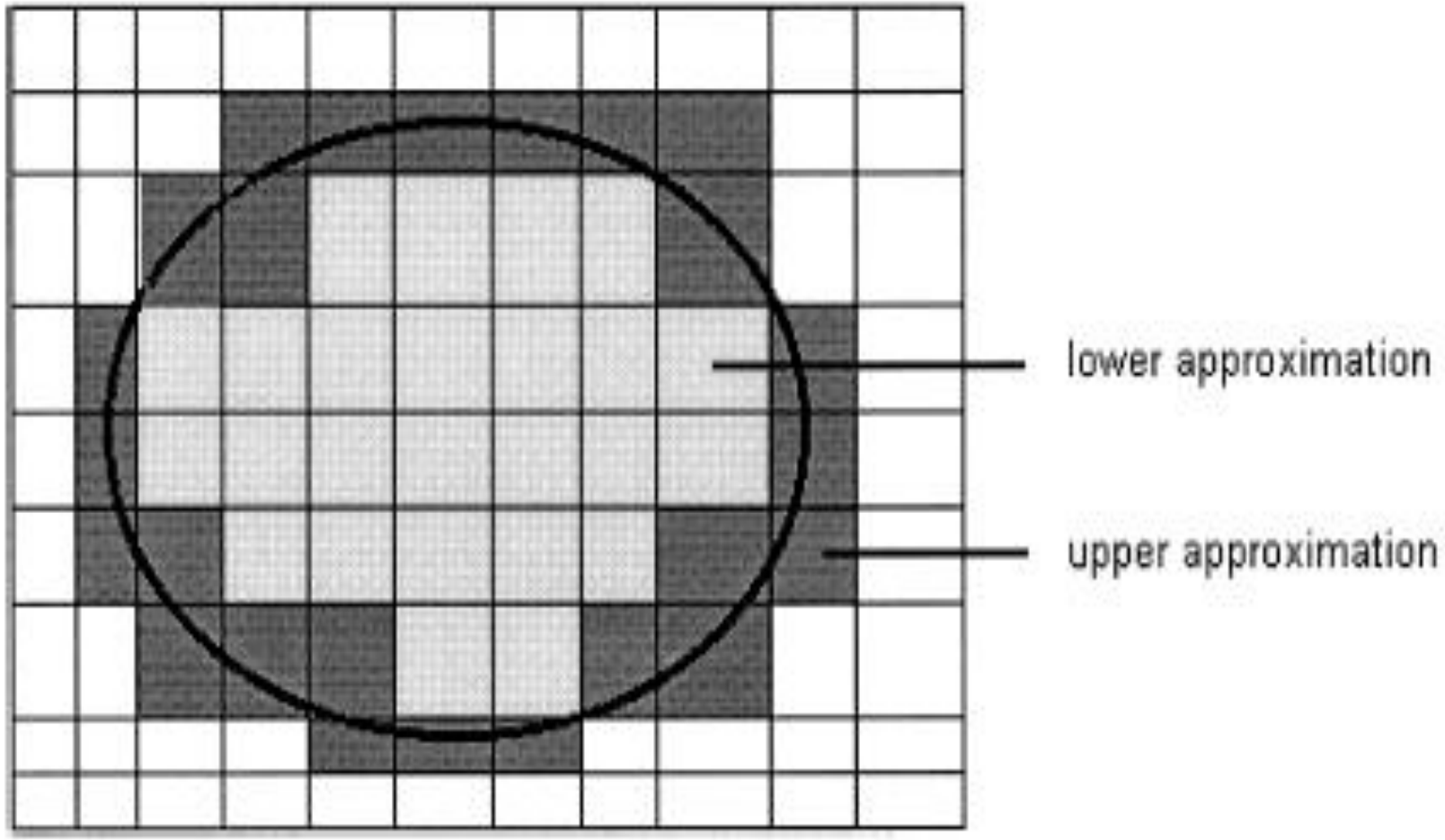
-the elements that doubtlessly belong to the set, and

-the elements that possibly belong to the set.

$$\mathrm{BX} = \left\{ x_i \in U \middle| \left[ x_i \right]_{\mathrm{Ind}(B)} \cap X \neq 0 \right\}.$$

$$\underline{\mathrm{BX}} = \left\{ x_i \in U \middle| \left[ x_i \right]_{\mathrm{Ind}(B)} \subset X \right\}.$$

# Basic concepts of the rough sets theory



# Basic concepts of the rough sets theory

Let us assume that we are interested in the subset $X$ of five objects $\{X = x_1, x_3, x_4, x_5, x_9\}$. Can we distinguish this set from the whole data set in the space of three attributes $(B = \{a_1, a_2, a_3\})$? Based on the results presented in Table 2, one can calculate the lower and upper approximations of this set in the following way.

# Basic concepts of the rough sets theory

The elementary sets presented in Table 2, which are also contained in $X$, are:

$\{x_1, x_3, x_9\}$, $\{x_4\}$.

Table 2

| $U/A$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|
| $\{x_1, x_3, x_9\}$ | 2 | 1 | 3 |
| $\{x_2, x_7, x_{10}\}$ | 3 | 2 | 1 |
| $\{x_4\}$ | 2 | 2 | 3 |
| $\{x_5, x_8\}$ | 1 | 1 | 4 |
| $\{x_6\}$ | 1 | 1 | 2 |

# Basic concepts of the rough sets theory

It means that the lower approximation is given by the following set of objects:

$$\underline{BX} = \{x_1, x_3, x_4, x_9\}.$$

To calculate the upper approximation of the subset $X$, one has to find in Table 2 all elementary sets which have at least 1 element in common with the subset $X$. These are:

$$\{x_1, x_3, x_9\}, \{x_4\}, \{x_5, x_8\}$$

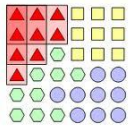
so that the upper approximation is:

$$BX = \{x_1, x_3, x_4, x_5, x_8, x_9\}.$$

# Basic concepts of the rough sets theory

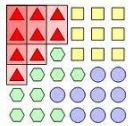
The boundary of $X$ in $U$, defined as the difference between the upper and lower approximations, contains elements which are in the upper but not in the lower approximation:

$$BNX = \{x_1, x_3, x_4, x_5, x_8, x_9\} - \{x_1, x_3, x_4, x_9\}$$
$$= \{x_5, x_8\}.$$



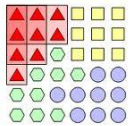
# Case Based Reasoning

# Faced this situation before?



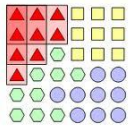
- Oops the car stopped.
  - What could have gone wrong?
- Aah.. Last time it happened, there was no petrol.
  - Is there petrol?
    - Yes.
  - Oh but wait I remember the tyre was punctured
- This is the normal thought process of a human when faced with a problem which is similar to a problem he/she had faced before.

# How do we solve problems?



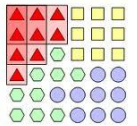
- By knowing the steps to apply
  - from symptoms to a plausible diagnosis
- How does an expert solve problems?
  - uses same “book learning” as a novice
  - but quickly selects the right knowledge to apply
- Heuristic knowledge (“rules of thumb”)
  - *“I don’t know why this works but it does and so I’ll use it again!”*

# So what?



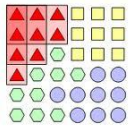
- Reuse the solution experience when faced with a similar problem.
- This is Case Based Reasoning (CBR)!
  - memory-based problem-solving
  - re-using past experiences
- Experts often find it easier to relate stories about past cases than to formulate rules

# What's CBR?

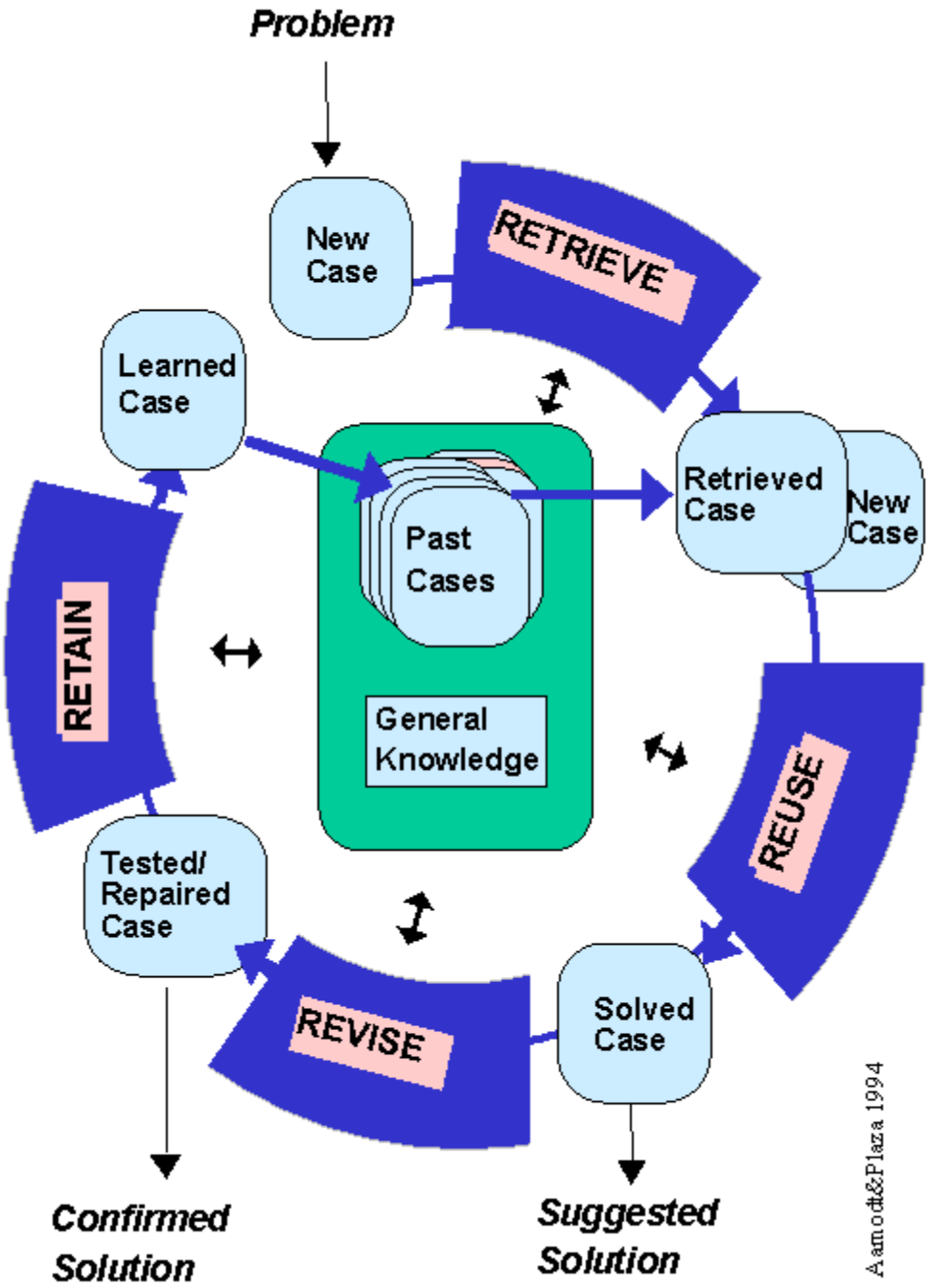


- To solve a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation
- Ex: Medicine
  - doctor remembers previous patients especially for rare combinations of symptoms
- Ex: Law
  - case histories are consulted
- Ex: Management
  - decisions are often based on past rulings
- Ex: Financial
  - performance is predicted by past results

# Definitions of CBR



- Case-based reasoning is […] reasoning by remembering - *Leake, 1996*
- A case-based reasoner solves new problems by adapting solutions that were used to solve old problems - *Riesbeck & Schank, 1989*
- Case-based reasoning is a recent approach to problem solving and learning […] - *Aamodt & Plaza, 1994*

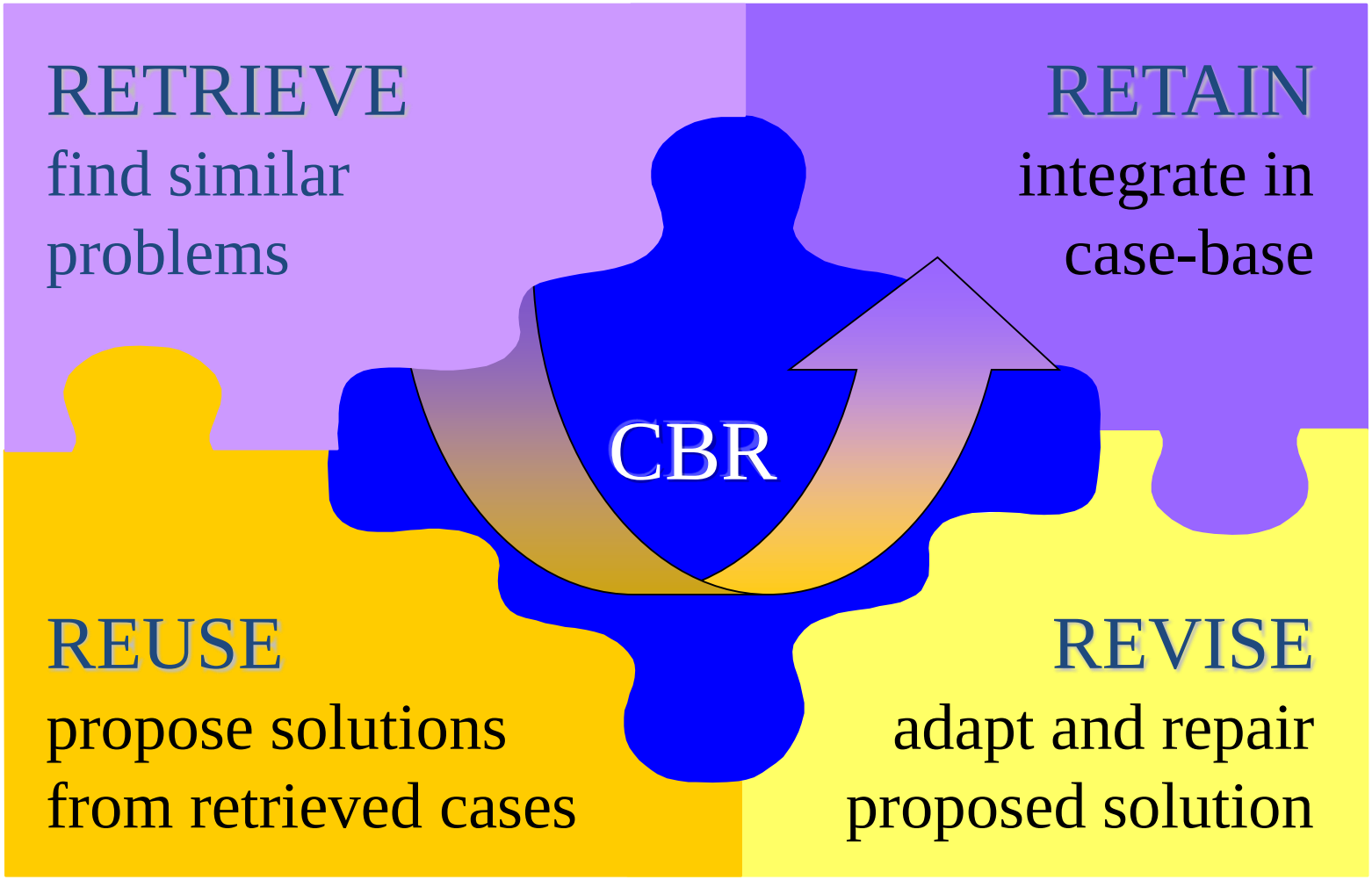



# CBR Cycle

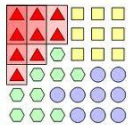


By: Suresh Pokharel

# $R^4$ Cycle

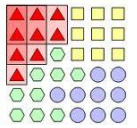

RETRIEVE
find similar
problems
RETAIN
integrate in
case-base
CBR
REUSE
propose solutions
from retrieved cases
REVISE
adapt and repair
proposed solution

# CBR System Components



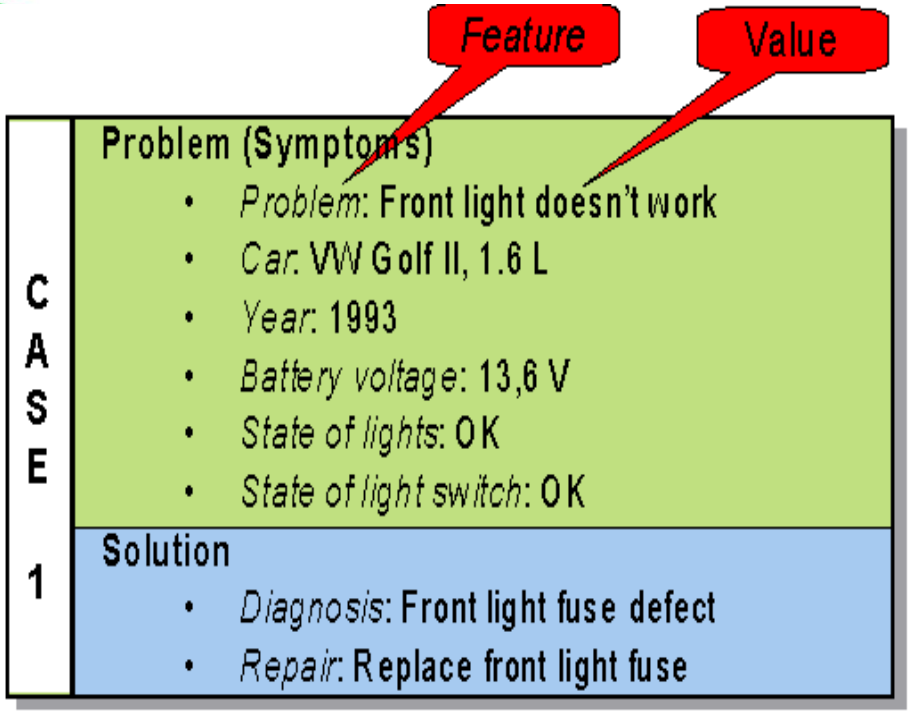
- Case-base
  - database of previous cases (experience)
- Retrieval of relevant cases
  - index for cases in library
  - matching most similar case(s)
  - retrieving the solution(s) from these case(s)
- Adaptation of solution
  - alter the retrieved solution(s) to reflect differences between new case and retrieved case(s)

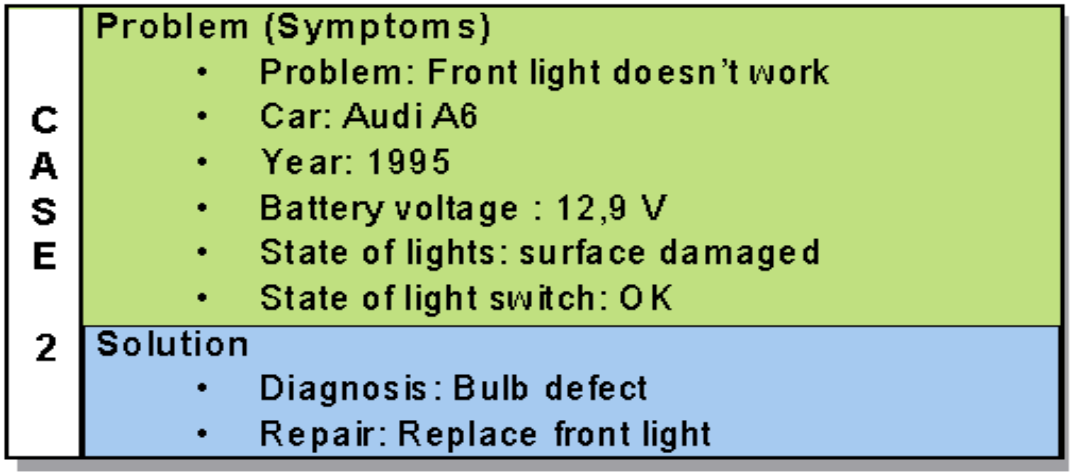
# CBR Assumption(s)

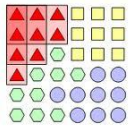


- The main assumption is that:
  - *Similar problems have similar solutions:*
    - e.g., an aspirin can be taken for any mild pain
- Two other assumptions:
  - The world is a regular place: what holds true today will probably hold true tomorrow
    - (e.g., if you have a headache, you take aspirin, because it has always helped)
  - Situations repeat: if they do not, there is no point in remembering them
    - (e.g., it helps to remember how you found a parking space near that restaurant)

# Technical Diagnosis of Car Faults

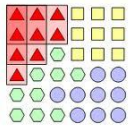




# An Introduction of
# Support Vector Machine

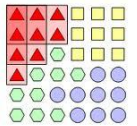
# Today: Support Vector Machine (SVM)



- A classifier derived from statistical learning theory by Vapnik, et al. in 1992
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.

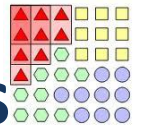
V. Vapnik

# Organization



- Basic idea of support vector machines
  - Optimal hyperplane for linearly separable patterns
  - Extend to patterns that are not linearly separable by transformations of original data to map into new space- Kernel function

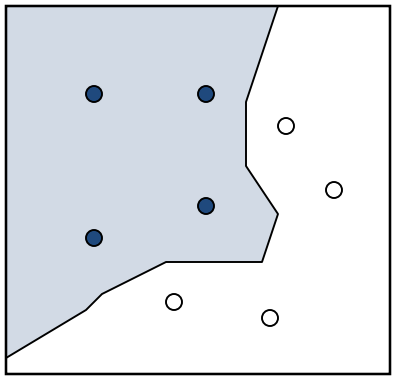
# Unique Features of SVM's and Kernel Methods



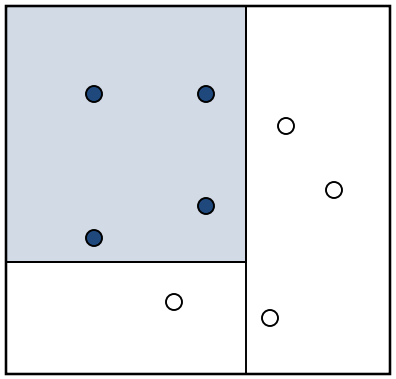
- Are explicitly based on a theoretical model of learning
- Come with theoretical guarantees about their performance
- Have a modular design that allows one to separately implement and design their components
- Do not suffer from the curse of dimensionality

# Discriminant Function

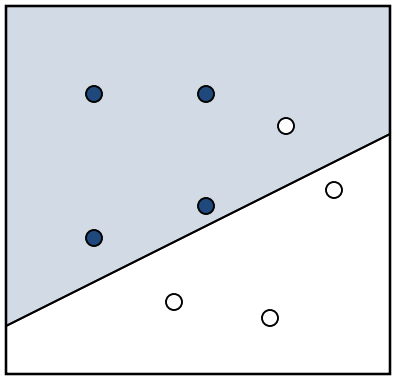
- It can be arbitrary functions of *x*, such as:



Nearest Neighbor

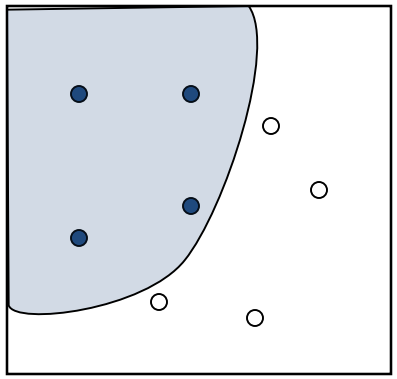


Decision Tree



Linear Functions

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$



Nonlinear Functions

# Linear Discriminant Function

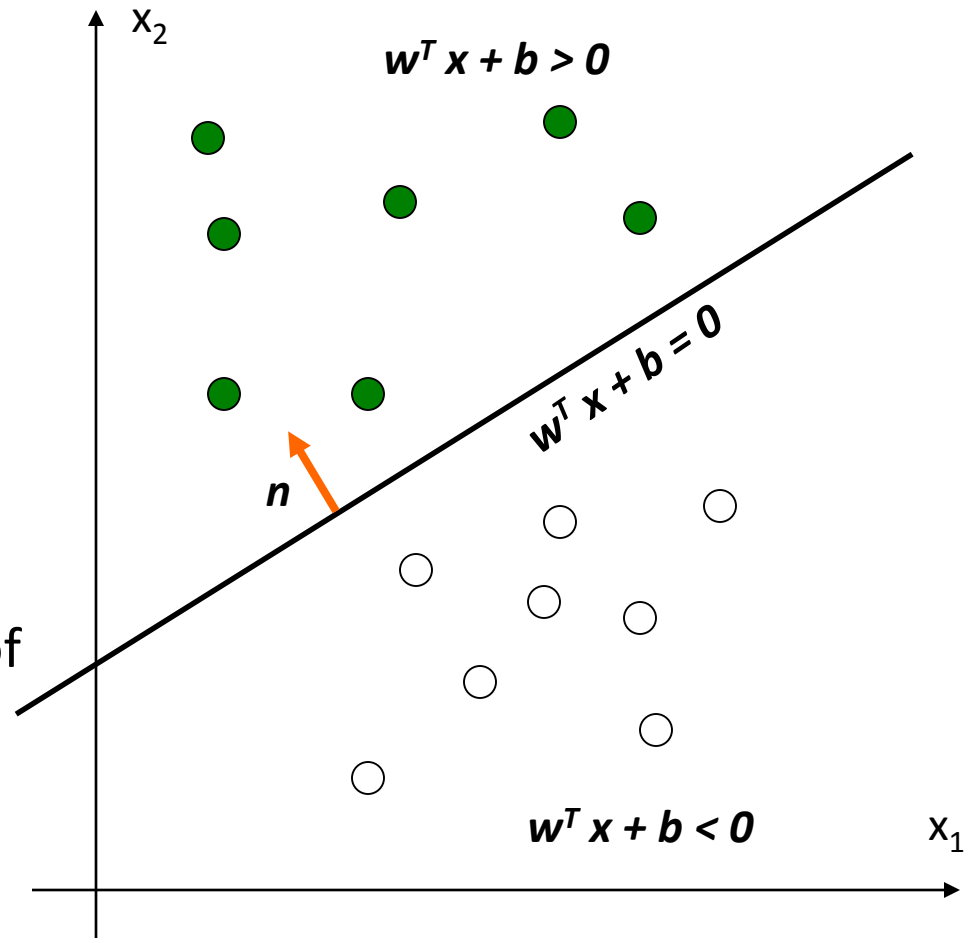
- g(x) is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

- A hyper-plane in the feature space

- (Unit-length) normal vector of the hyper-plane:

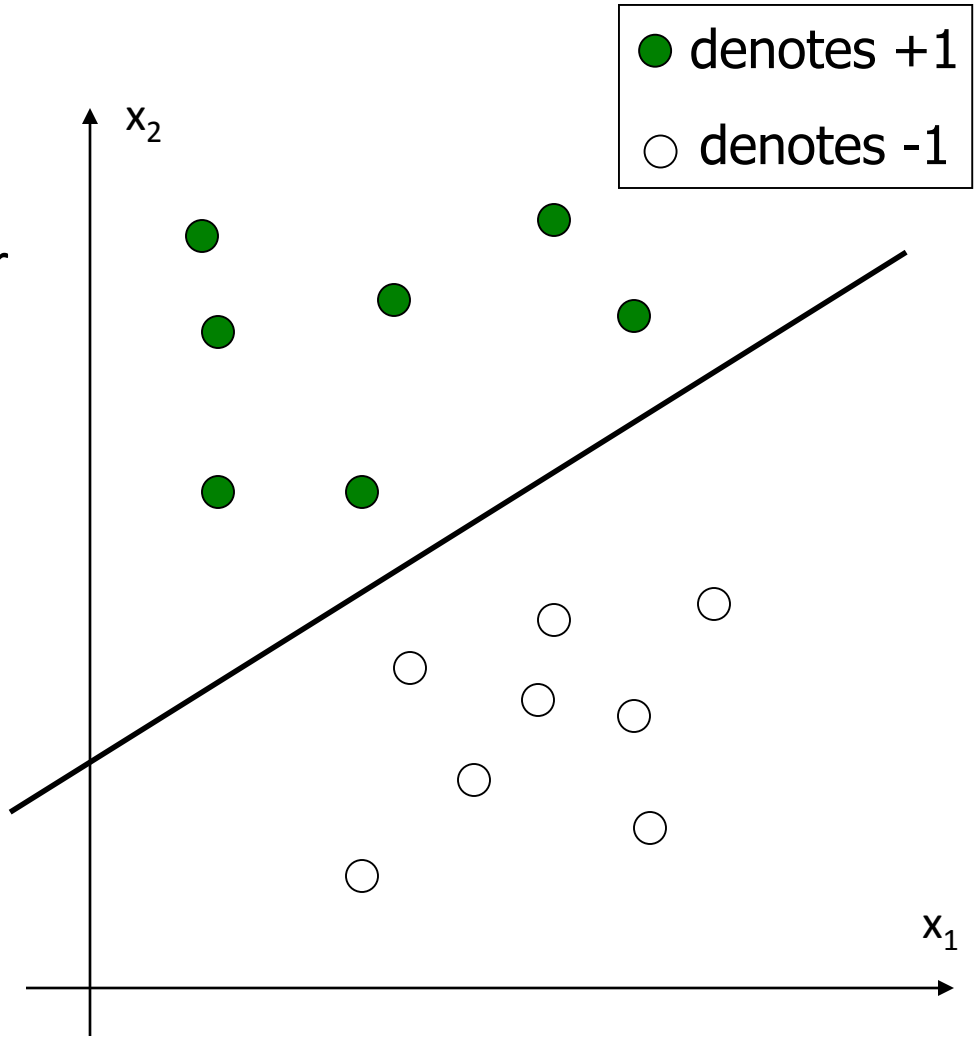
$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



# Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

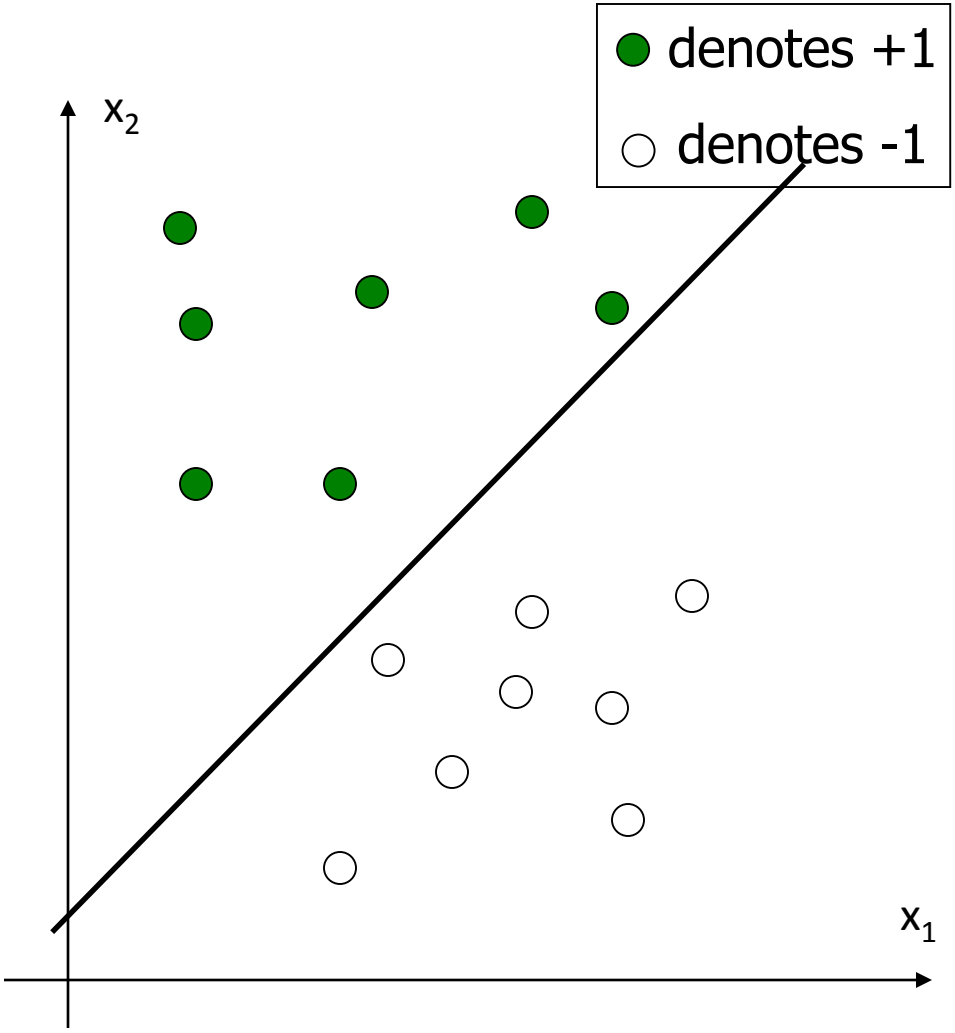
- Infinite number of answers!



# Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

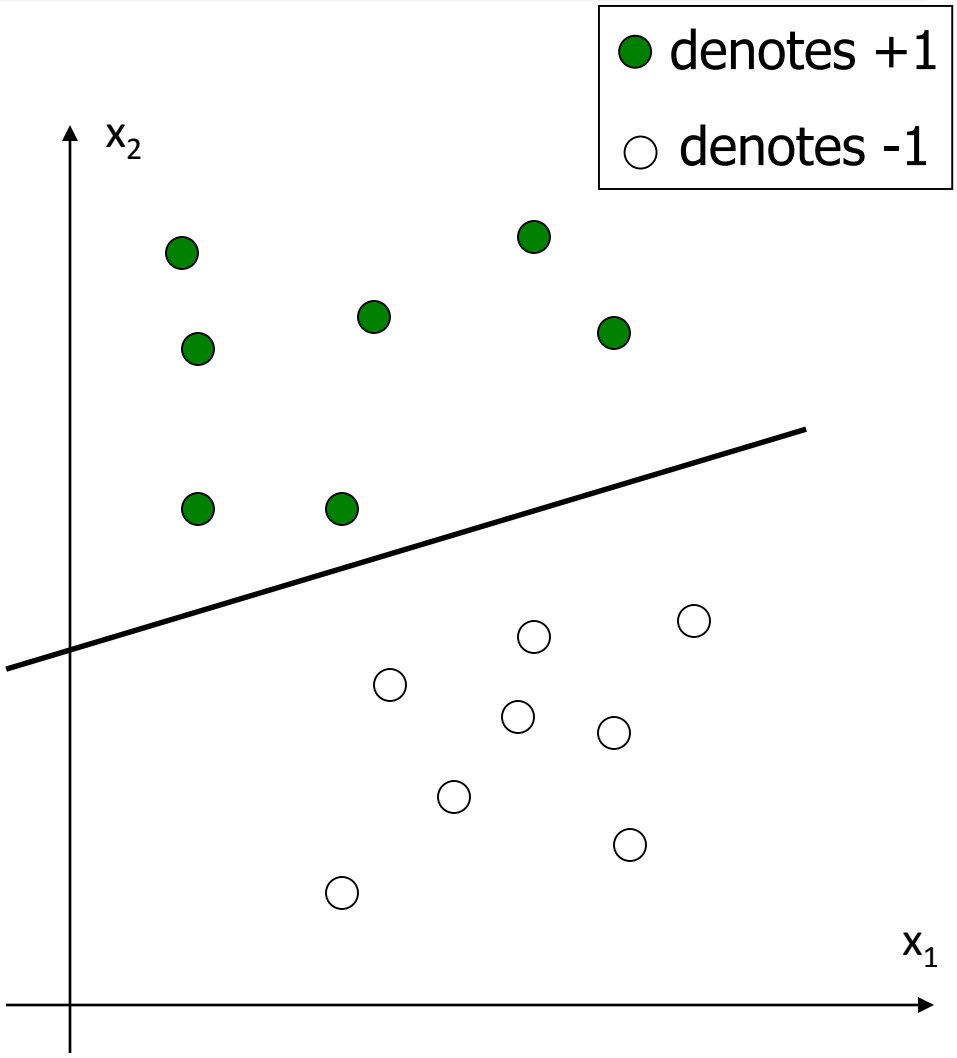
- Infinite number of answers!



# Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!

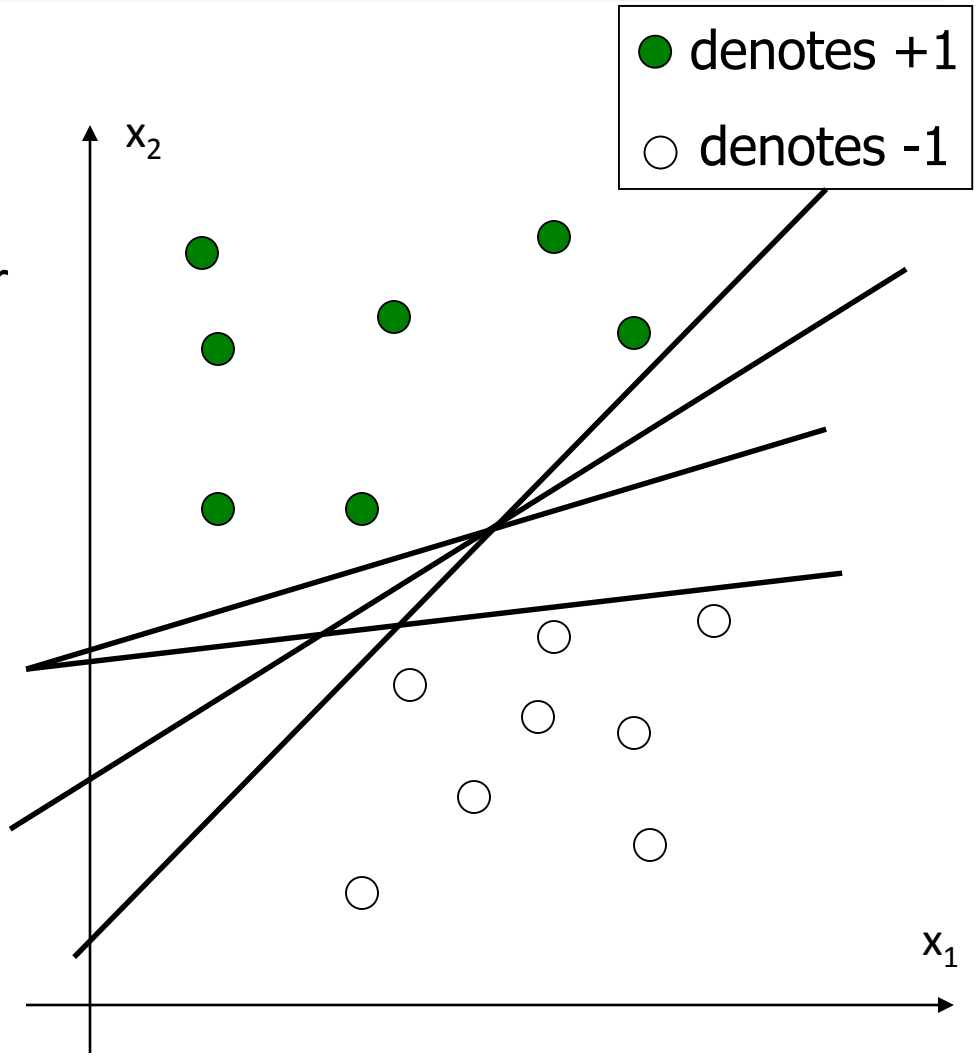


# Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

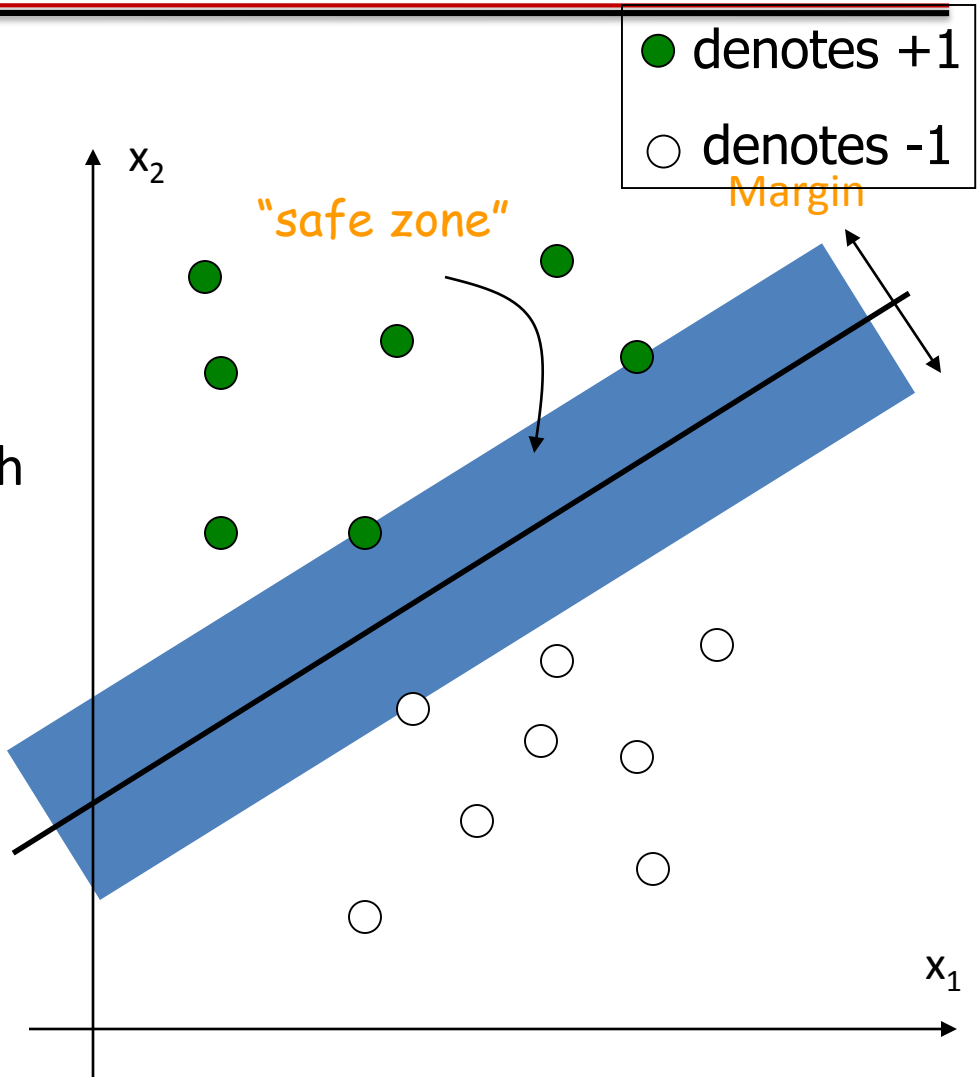
- Infinite number of answers!

- Which one is the best?



# Large Margin Linear Classifier

- The linear discriminant function (classifier) with the maximum margin is the best
- Margin is defined as the width that the boundary could be increased by before hitting a data point



# Large Margin Linear Classifier

- Given a set of data points:

$\{(\mathbf{x}_i, y_i)\},\ i = 1, 2, \cdots, n$, where

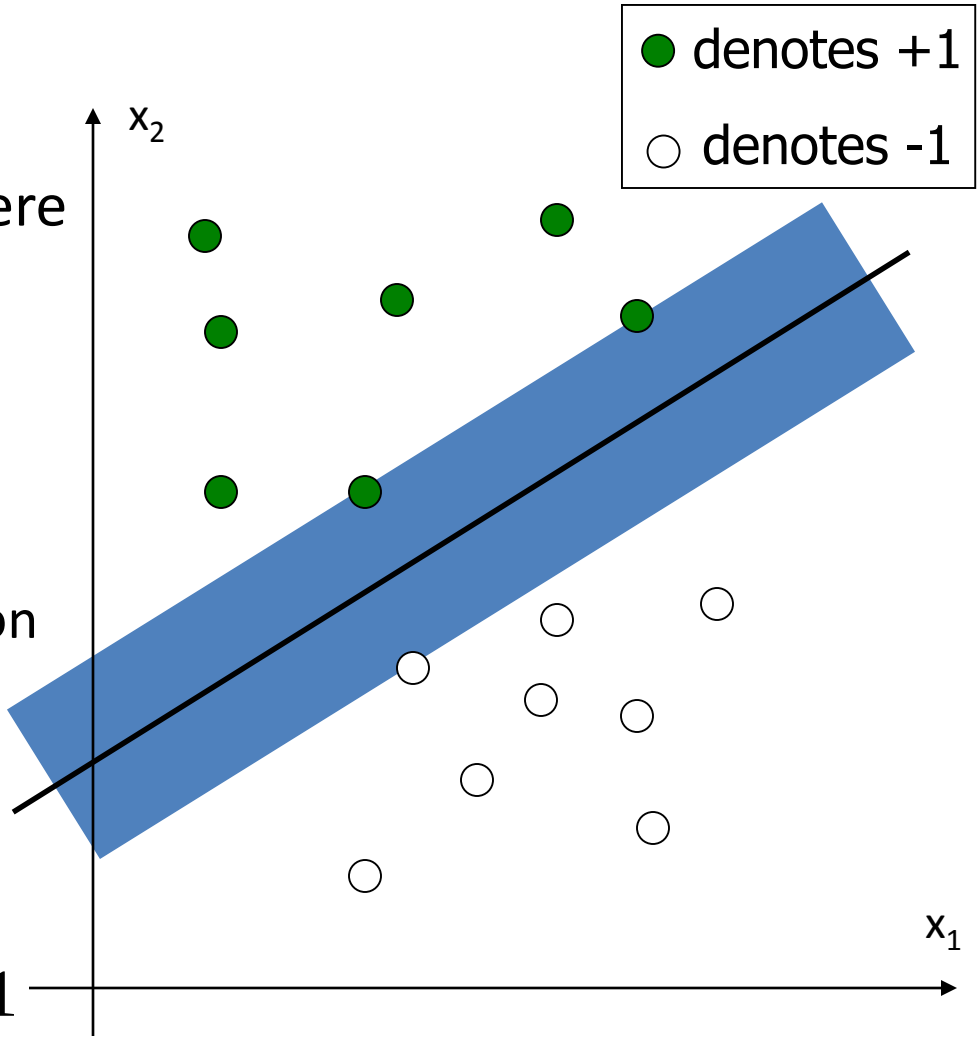
$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b > 0$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b < 0$$

- With a scale transformation on both *w* and *b*, the above is equivalent to

$$\text{For } y_i = +1,\quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1,\quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



# Large Margin Linear Classifier

- We know that

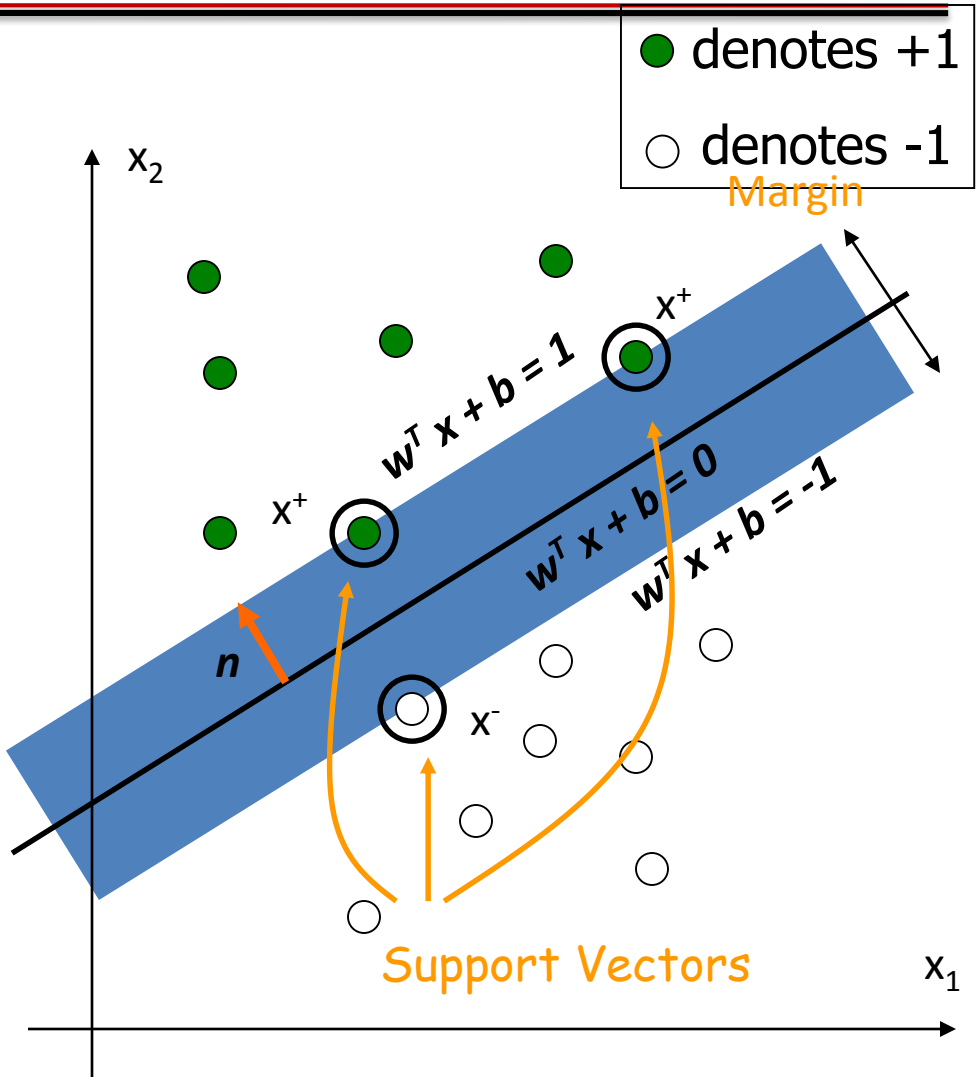
$$\mathbf{w}^T \mathbf{x}^+ + b = 1$$

$$\mathbf{w}^T \mathbf{x}^- + b = -1$$

- The margin width is:

$$M = (\mathbf{x}^+ - \mathbf{x}^-) \cdot \mathbf{n}$$

$$= (\mathbf{x}^+ - \mathbf{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$



# Large Margin Linear Classifier

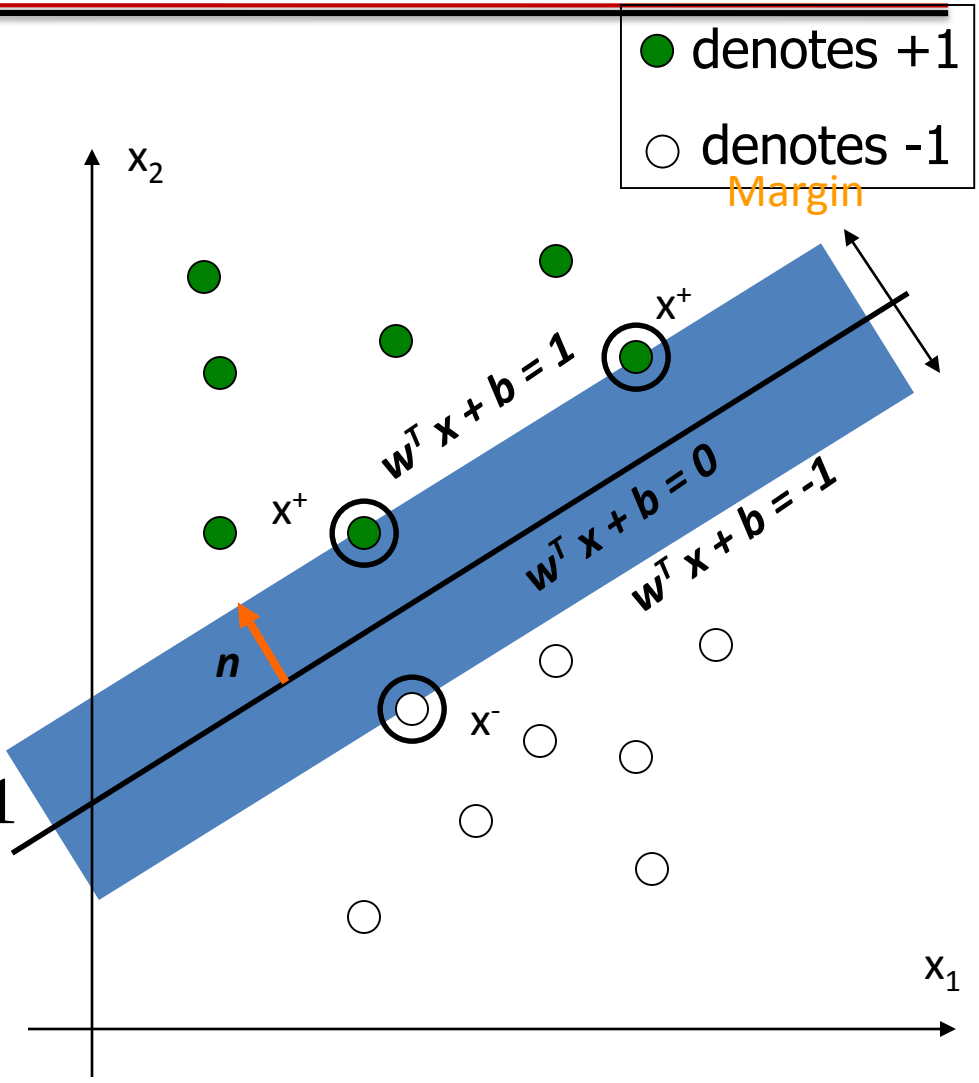
- Formulation:

$$\text{maximize} \quad \frac{2}{\|\mathbf{w}\|}$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



# Large Margin Linear Classifier

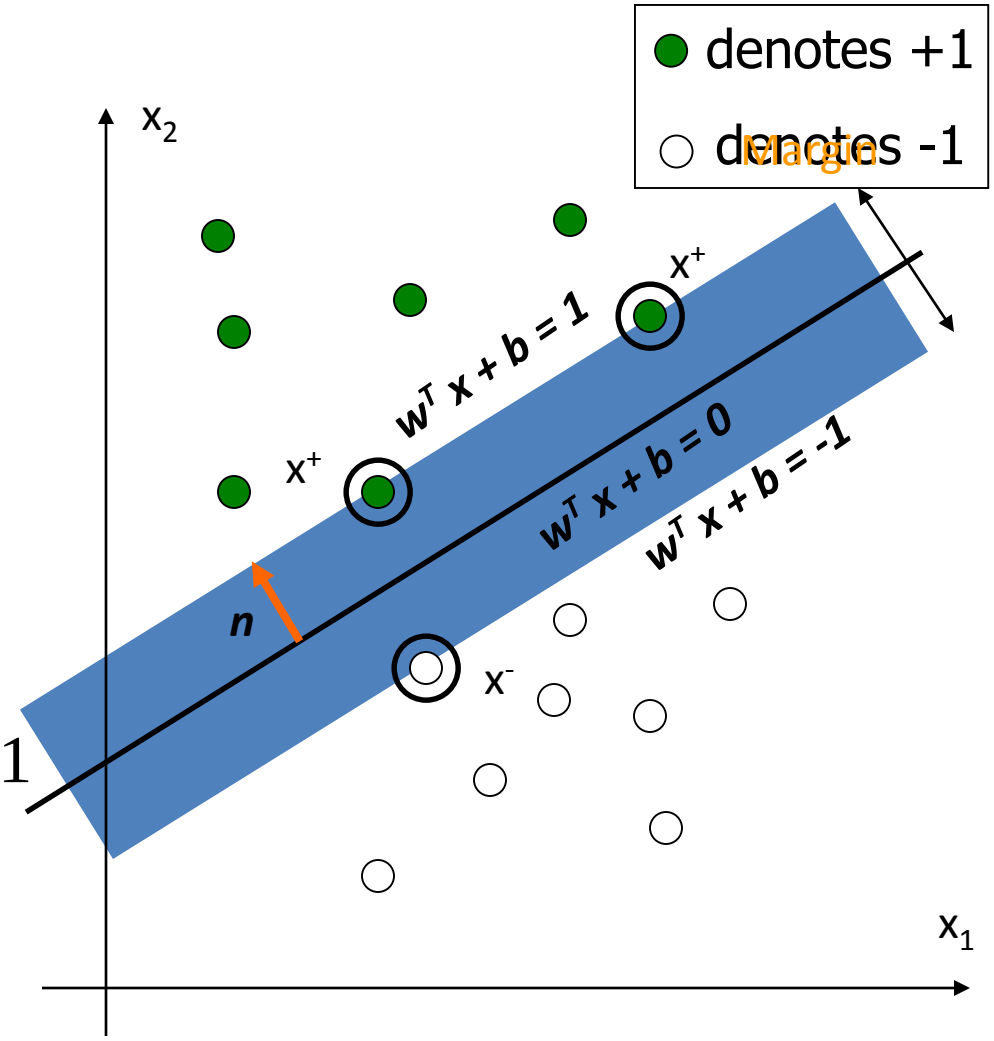
- Formulation:

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

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



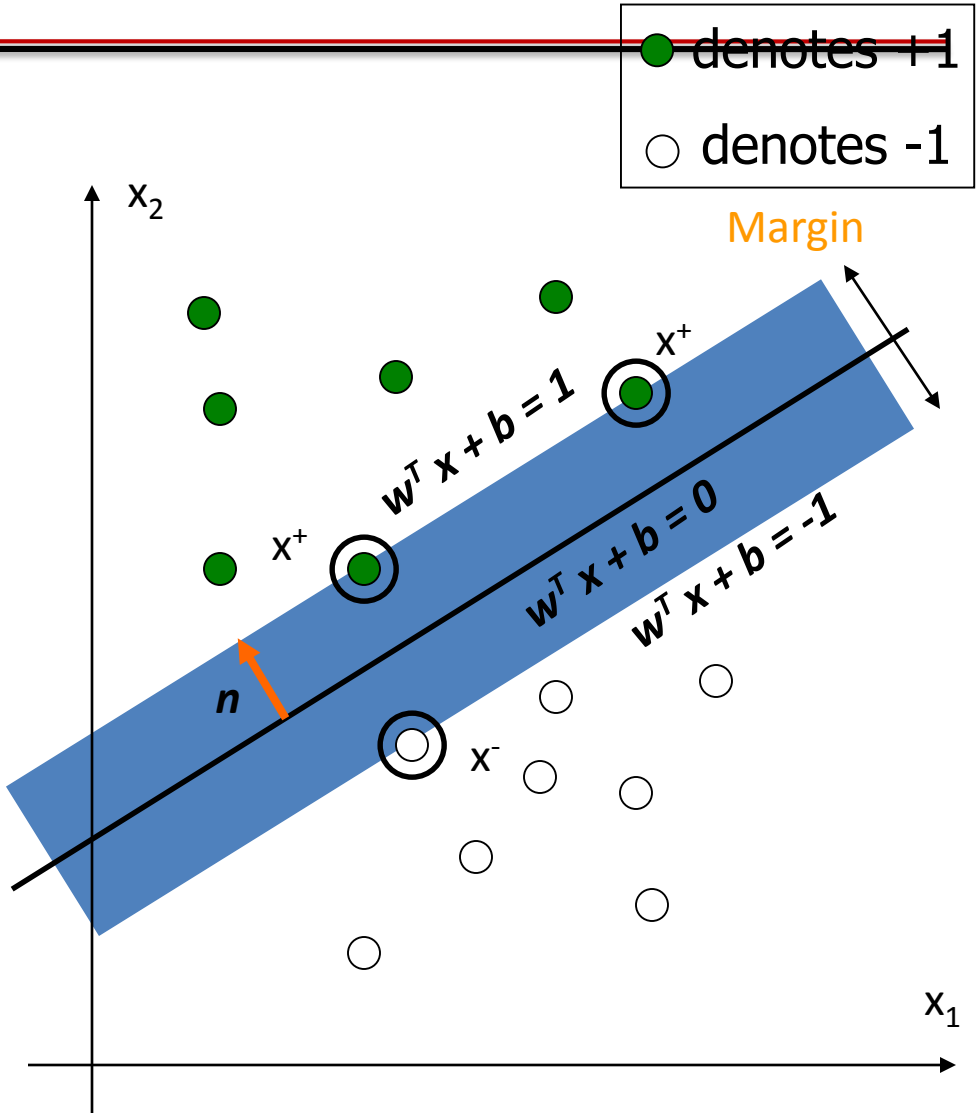
# Large Margin Linear Classifier

- Formulation:

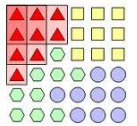
$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

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



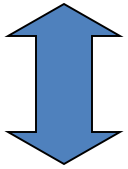
# Solving the Optimization Problem



Quadratic programming with linear constraints

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

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$



Lagrangian Function

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

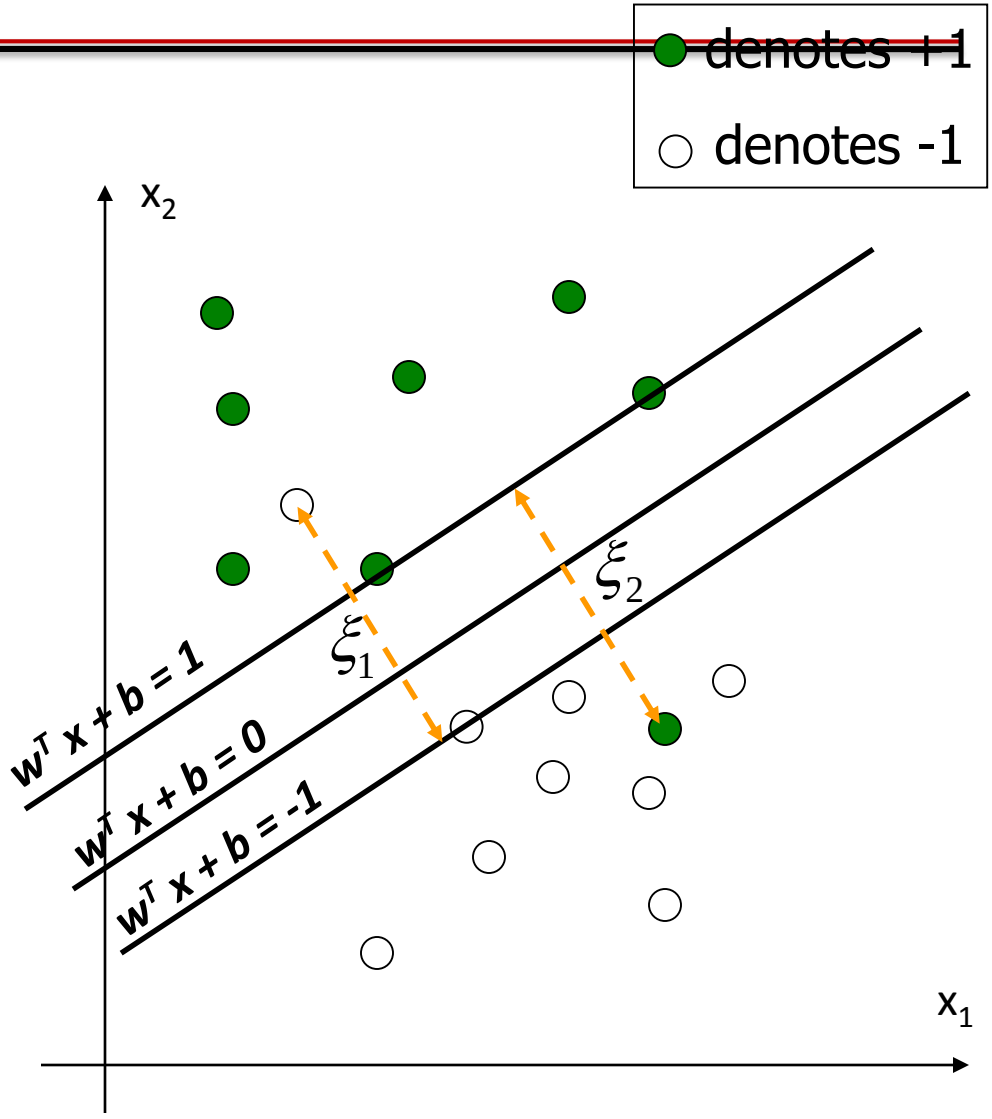
$$\text{s.t.} \quad \alpha_i \geq 0$$

Note: Mathematical proof is not cover here.

# Large Margin Linear Classifier

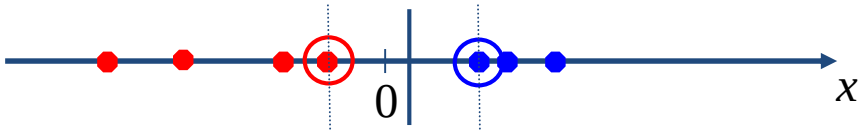
- What if data is not linear separable? (noisy data, outliers, etc.)

- Slack variables $\xi_i$ can be added to allow mis-classification of difficult or noisy data points



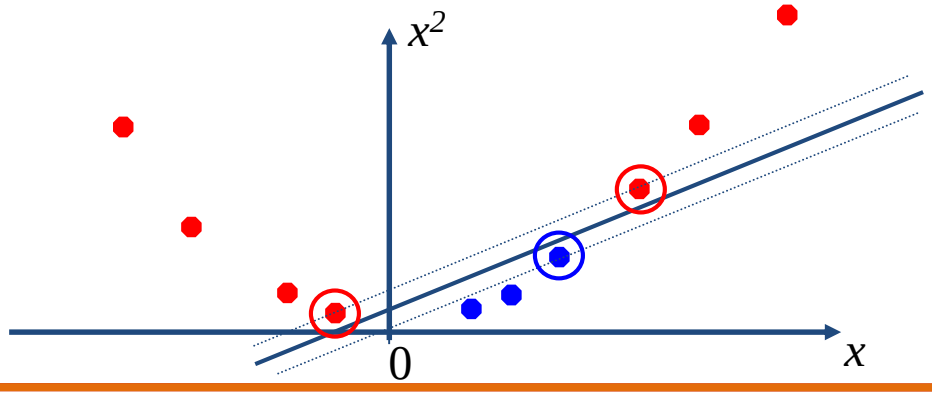
# Non-linear SVMs

- Datasets that are linearly separable with noise work out great:



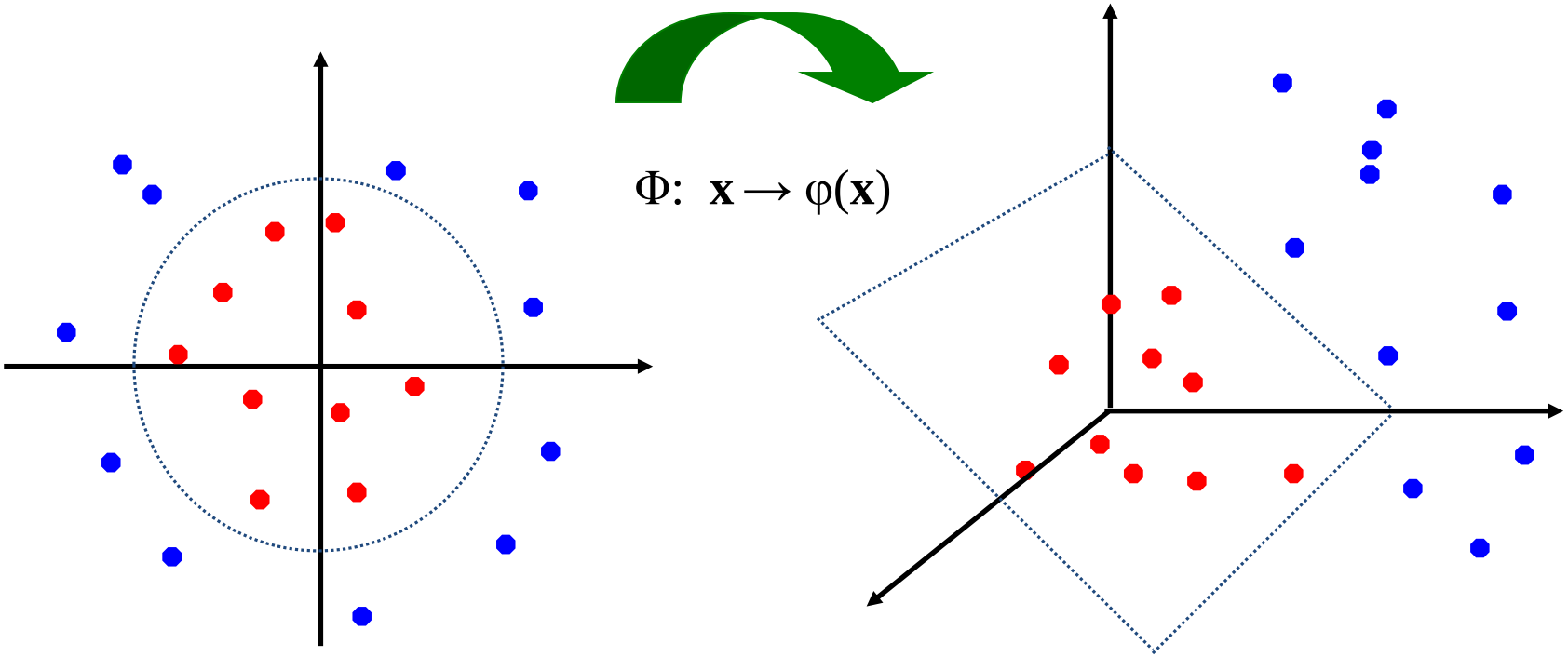

- But what are we going to do if the dataset is just too hard?


- How about… mapping data to a higher-dimensional space:

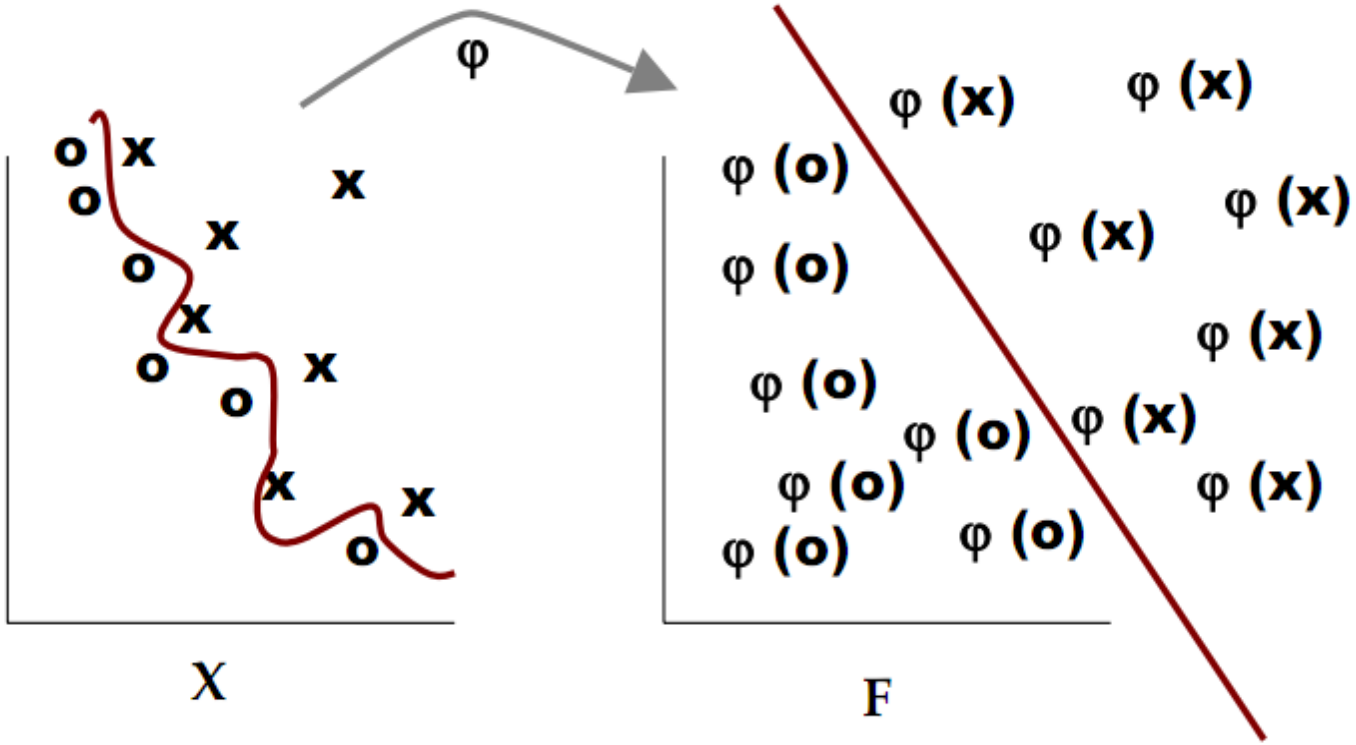


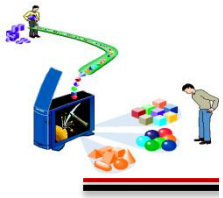
# Non-linear SVMs: Feature Space

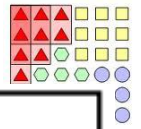
- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



# Transformation to separate







# Examples for Non Linear SVMs

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y}+1)^{p}$$

$$K(\mathbf{x},\mathbf{y}) = \exp\left\{-\|\mathbf{x}-\mathbf{y}\|^2 \Big/ 2\sigma^2\right\}$$

$$K(\mathbf{x},\mathbf{y}) = \tanh(\kappa\mathbf{x}\cdot\mathbf{y}-\delta)$$

1st is polynomial (includes x•x as special case)
2nd is radial basis function (gaussians)
3rd is sigmoid (neural net activation function)

# Classifier Accuracy

| | $C_1$ | $C_2$ |
|---|---|---|
| $C_1$ | True positive | False negative |
| $C_2$ | False positive | True negative |

| classes | buy_computer = yes | buy_computer = no | total | recognition(%) |
|---|---|---|---|---|
| buy_computer = yes | 6954 | 46 | 7000 | 99.34 |
| buy_computer = no | 412 | 2588 | 3000 | 86.27 |
| total | 7366 | 2634 | 10000 | 95.42 |

- Accuracy of a classifier M, acc(M): percentage of test set tuples that are correctly classified by the model M
  - Error rate (misclassification rate) of M = 1 – acc(M)
  - Given *m* classes, $CM_{i,j}$, an entry in a **confusion matrix**, indicates # of tuples in class *i* that are labeled by the classifier as class *j*
- Alternative accuracy measures (e.g., for cancer diagnosis)

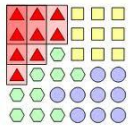
  sensitivity = t-pos/pos /* true positive recognition rate */

  specificity = t-neg/neg /* true negative recognition rate */

  precision = t-pos/(t-pos + f-pos)

  accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)

# Classifier Accuracy



$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

- Cross-validation (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use $D_i$ as test set and others as training set
  - Leave-one-out: k folds where k = # of tuples, for small sized data

# Classifier Accuracy