Classification and Regression

Decision Tree C4.5

- **ID3** favors attributes (tests) with large number of values / outcomes
  - biased towards multivalued attributes

- **C4.5** (a successor of ID3)
- Improved version of ID3
  - Gain Ratio:

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

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

- C4.5 uses gain ratio to overcome the problem
  - normalization to information gain

Decision Tree C4.5

Building a Decision Tree [C4.5]

- Consider the following dataset, we want to decide whether the customer is likely to buys_computer or not for 14 records, where
  - Class **P** = 9: buys_computer = "**yes**"
  - Class **N** = 5: buys_computer = "**no**"

1. What is the best split (among ***age***, ***income***, ***student***, and ***credit_rating***) according to the **Gain Ratio**?
2. Also, construct complete Decision Tree on the given set of training examples using ***Gain Ratio***.
3. Use the final tree to classify the record (***youth, low, no, excellent***).

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Decision Tree C4.5

Gain Ratio [C4.5] - Example

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

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

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

$$Gain(A) = Info(D) - Info_A(D)$$

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

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

- Calculate ***Entropy*** of Class attribute:

buys_computer	
yes	no
9	5

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.9403$$

- Calculate ***Gain Ratio*** of all other attributes:

$$-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)$$

		Class		
		yes	no	
age	youth	2	3	5
	middle_aged	4	0	4
	senior	3	2	5
				14

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2)$$
$$= \frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.3467 + 0 + 0.3467 = \mathbf{0.6934}$$

$$Gain(age) = Info(D) - Infoage_{(D)} = 0.9403 - 0.6934 = \mathbf{0.2469}$$

$$SplitInf0_{age}(D) = -\frac{5}{14} * \log2\left(\frac{5}{14}\right) - \frac{4}{14} * \log_2\left(\frac{4}{14}\right) - \frac{5}{14} * \log2\left(\frac{5}{14}\right) = \mathbf{1.5774}$$

$$\mathbf{GainRatio(age)} = \frac{Gain(A)}{SplitInfo(A)} = \frac{0.246}{1.5774} = \mathbf{0.1559}$$

		Class		
		yes	no	
income	low	3	1	4
	medium	4	2	6
	high	2	2	4
				14

$$Info_{income}(D) = \frac{4}{14}I(3,1) + \frac{6}{14}I(4,2) + \frac{4}{14}I(2,2)$$
$$= \frac{4}{14} * 0.8113 + \frac{6}{14} * 0.9183 + \frac{4}{14} * 1 = 0.2318 + 0.3935 + 0.2857 = \mathbf{0.911}$$

$$Gain(income) = 0.9403 - 0.911 = \mathbf{0.0293}$$

$$SplitInf0_{income}(D) = -\frac{4}{14} * \log2\left(\frac{4}{14}\right) - \frac{6}{14} * \log_2\left(\frac{6}{14}\right) - \frac{4}{14} * \log2\left(\frac{4}{14}\right) = \mathbf{1.5566}$$

$$\mathbf{GainRatio(income)} = \frac{0.0293}{1.5566} = \mathbf{0.0188}$$

Decision Tree C4.5

Gain Ratio [C4.5] - Example

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$ $$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

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

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

- Calculate ***Entropy*** of Class attribute:

buys_computer	
yes	no
9	5

$$Info(D) = I(9,8) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = \mathbf{0.9403}$$

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

- Calculate ***Gain Ratio*** of all other attributes:

		Class		
		yes	no	
student	yes	6	1	7
	no	3	4	7
				14

$$Info_{student}(D) = \frac{7}{14}I(6,1) + \frac{7}{14}I(3,4)$$
$$= \frac{7}{14} * 0.5917 + \frac{7}{14} * 0.9852 = 0.2958 + 0.4926 = \mathbf{0.7884}$$

$$Gain(student) = 0.9403 - 0.7884 = \mathbf{0.1519}$$

$$SplitInf0_{student}(D) = -\frac{7}{14} * \log2\left(\frac{7}{14}\right) - \frac{7}{14} * \log_2\left(\frac{7}{14}\right) = \mathbf{1}$$

$$GainRatio(student) = \frac{0.1519}{1} = \mathbf{0.1519}$$

		Class		
		yes	no	
credit_rating	fair	6	2	8
	excellent	3	3	6
				14

$$Info_{credit\_rating}(D) = \frac{8}{14}I(6,2) + \frac{6}{14}I(3,3)$$
$$= \frac{8}{14} * 0.8113 + \frac{6}{14} * 1 = 0.4636 + 0.4286 = \mathbf{0.8922}$$

$$Gain(credit - rating) = 0.9403 - 0.8922 = \mathbf{0.0481}$$

$$SplitInf0_{credit-rating}(D) = -\frac{8}{14} * \log2\left(\frac{8}{14}\right) - \frac{6}{14} * \log_2\left(\frac{6}{14}\right) = \mathbf{0.9852}$$

$$GainRatio(credit - rating) = \frac{0.0481}{0.9852} = \mathbf{0.0488}$$

- As, the Gain Ratio of "age" is highest,
- So "**age**" is the best attribute & becomes the root node of the decision tree.

Decision Tree C4.5

Age?

youth

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

middle_aged

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

senior

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

Age?

youth

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

middle_aged

yes

senior

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

Decision Tree C4.5

Gain Ratio [C4.5] - Example

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

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

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

$$Gain(A) = Info(D) - Info_A(D)$$

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

- For Left subtree: Calculate ***Entropy*** of Class attribute:

buys_computer	
yes	no
2	3

$$Info(D) = I(2,3) = -\frac{2}{5}log_2\left(\frac{2}{5}\right) - \frac{3}{5}log_2\left(\frac{3}{5}\right) = 0.971$$

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

- Calculate ***Gain Ratio*** of all other attributes:

		Class		
		yes	no	
income	low	1	0	1
	medium	1	1	2
	high	0	2	2
				5

$$Info_{income}(D) = \frac{1}{5}I(1,0) + \frac{2}{5}I(1,1) + \frac{2}{5}I(0,2)$$

$$= \frac{1}{5} * 0 + \frac{2}{5} * 1 + \frac{2}{5} * 0 = 0 + 0.4 + 0 = \mathbf{0.4}$$

$$Gain(income) = 0.971 - 0.4 = \mathbf{0.571}$$

$$SplitInf0_{income}(D) = -\frac{1}{5} * \log 2\left(\frac{1}{5}\right) - \frac{2}{5} * \log_2\left(\frac{2}{5}\right) - \frac{2}{5} * \log_2\left(\frac{2}{5}\right) = \mathbf{1.5219}$$

$$GainRatio(income) = \frac{0.571}{1.5219} = \mathbf{0.3751}$$

		Class		
		yes	no	
student	yes	2	0	2
	no	0	3	3
				5

$$Info_{student}(D) = \frac{2}{5}I(2,0) + \frac{3}{5}I(0,3) = \frac{2}{5} * 0 + \frac{3}{5} * 0 = \mathbf{0}$$

$$Gain(age) = 0.971 - 0 = \mathbf{0.971}$$

$$SplitInf0_{student}(D) = -\frac{2}{5} * \log_2\left(\frac{2}{5}\right) - \frac{3}{5} * \log 2\left(\frac{3}{5}\right) = \mathbf{0.9709}$$

$$GainRatio(student) = \frac{0.971}{0.9709} = \mathbf{1}$$

		Class		
		yes	no	
credit_ rating	fair	1	2	3
	excellent	1	1	2
				5

$$Info_{credit\_rating}(D) = \frac{3}{5}I(1,2) + \frac{2}{5}I(1,1)$$

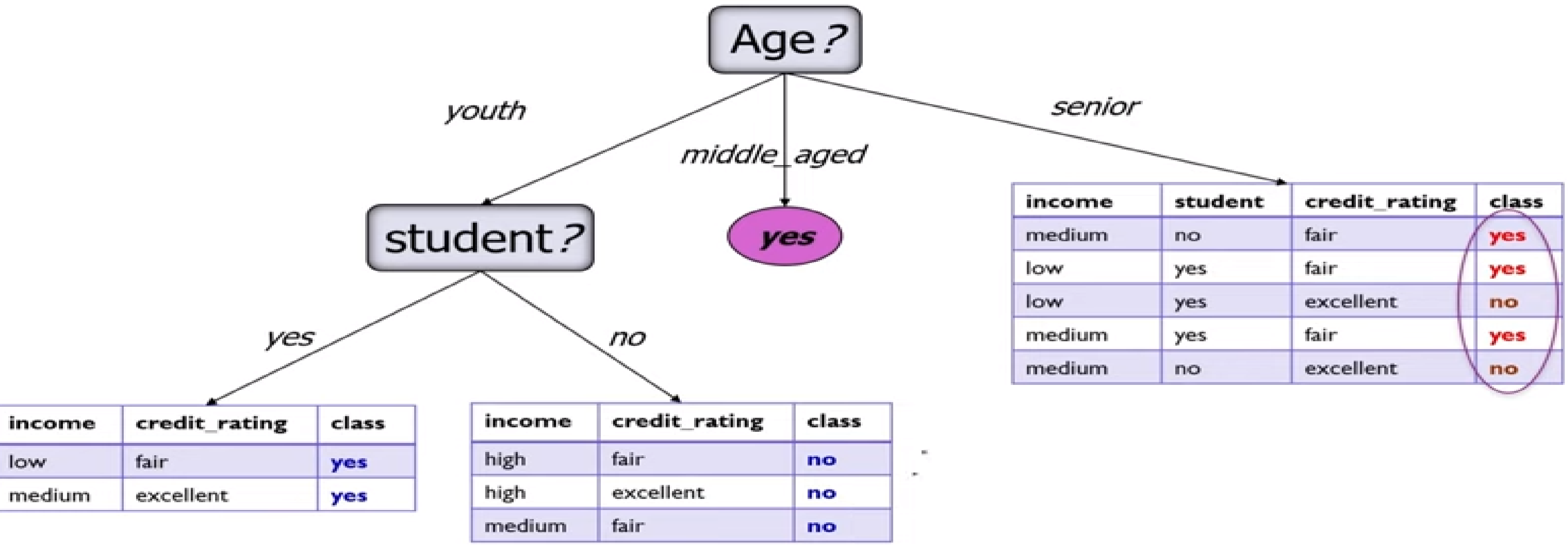
$$= \frac{3}{5} * 0.9183 + \frac{2}{5} * 1 = 0.3443 + 0.4 = \mathbf{0.7443}$$

$$Gain(credit\_rating) = 0.971 - 0.7443 = \mathbf{0.2267}$$

$$SplitInf0_{credit\_rating}(D) = -\frac{3}{5} * \log 2\left(\frac{3}{5}\right) - \frac{2}{5} * \log_2\left(\frac{2}{5}\right) = \mathbf{0.9709}$$

$$GainRatio(credit\_rating) = \frac{0.2267}{0.9709} = \mathbf{0.2335}$$

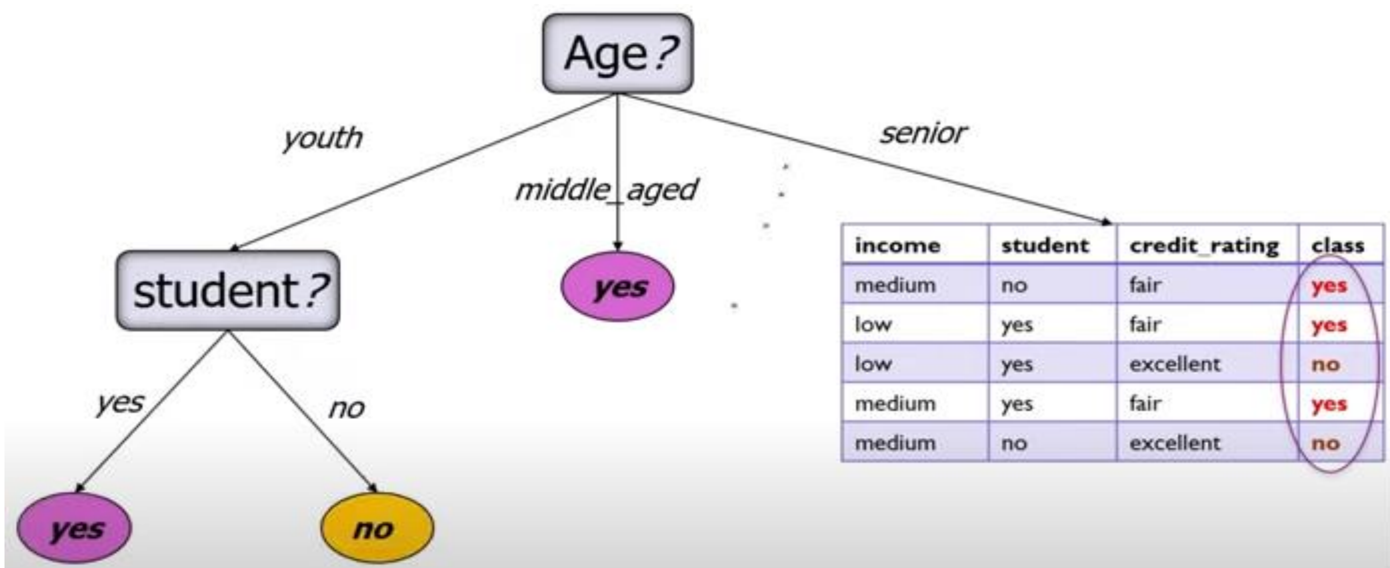
Decision Tree C4.5



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	credit_rating	class
low	fair	yes
medium	excellent	yes

income	credit_rating	class
high	fair	no
high	excellent	no
medium	fair	no



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

Decision Tree C4.5

Gain Ratio [C4.5] - Example

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

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

$$Gain(A) = Info(D) - Info_A(D)$$

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

- For Right subtree: Calculate ***Entropy*** of Class attribute:

buys_computer	
yes	no
3	2

$$Info(D) = I(3,2) = -\frac{3}{5}log_2\left(\frac{3}{5}\right) - \frac{2}{5}log_2\left(\frac{2}{5}\right) = 0.971 \checkmark$$

- Calculate ***Gain Ratio*** of all other attributes:

		Class		
		yes	no	
income	low	1	1	2
	medium	2	1	3
	high	0	0	0
				5

$$Info_{income}(D) = \frac{2}{5}I(1,1) + \frac{3}{5}I(2,1)$$
$$= \frac{2}{5} * 1 + \frac{3}{5} * 0.9183 = 0.4 + 0.551 = \mathbf{0.951}$$

$$Gain(income) = 0.971 - 0.951 = \mathbf{0.02}$$

$$SplitInfo_{income}(D) = -\frac{2}{5} * \log2\left(\frac{2}{5}\right) - \frac{3}{5} * \log_2\left(\frac{3}{5}\right) = \mathbf{0.9709}$$

$$\mathbf{GainRatio(income)} = \frac{0.02}{0.9709} = \mathbf{0.0205} \checkmark$$

		Class		
		yes	no	
credit_ rating	fair	3	0	3
	excellent	0	2	2
				5

$$Info_{credit\_rating}(D) = \frac{3}{5}I(3,0) + \frac{2}{5}I(0,2) = \frac{3}{5} * 0 + \frac{2}{5} * 0 = \mathbf{0}$$

$$Gain(credit\_rating) = 0.971 - 0 = \mathbf{0.971}$$

$$SplitInfo_{credit\_rating}(D) = -\frac{3}{5} * \log2\left(\frac{3}{5}\right) - \frac{2}{5} * \log_2\left(\frac{2}{5}\right) = \mathbf{0.9709}$$

$$\mathbf{GainRatio(credit\_rating)} = \frac{0.971}{0.9709} = \mathbf{1}$$

		Class		
		yes	no	
student	yes	2	1	3
	no	1	1	2
				5

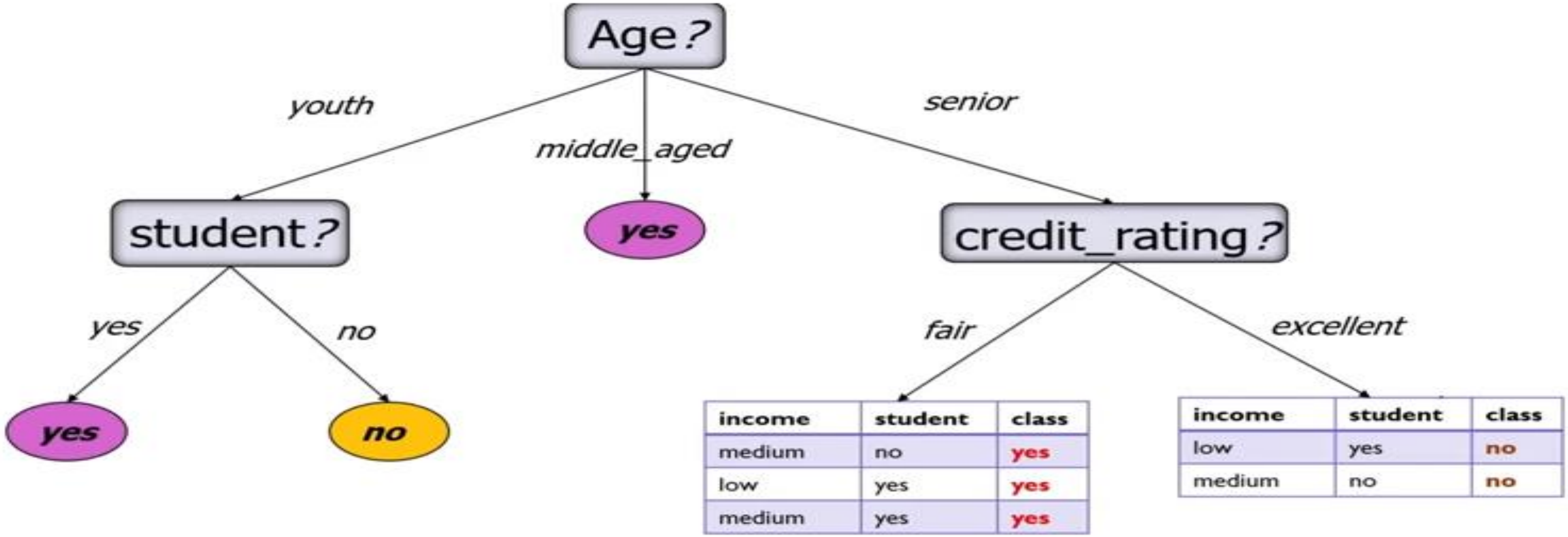
$$Info_{student}(D) = \frac{3}{5}I(2,1) + \frac{2}{5}I(1,1)$$
$$= \frac{3}{5} * 0.9183 + \frac{2}{5} * 1 = 0.551 + 0.4 = \mathbf{0.951}$$

$$Gain(age) = 0.971 - .951 = \mathbf{0.02}$$

$$SplitInfo_{student}(D) = -\frac{3}{5} * \log_2\left(\frac{3}{5}\right) - \frac{2}{5} * \log2\left(\frac{2}{5}\right) = \mathbf{0.9709}$$

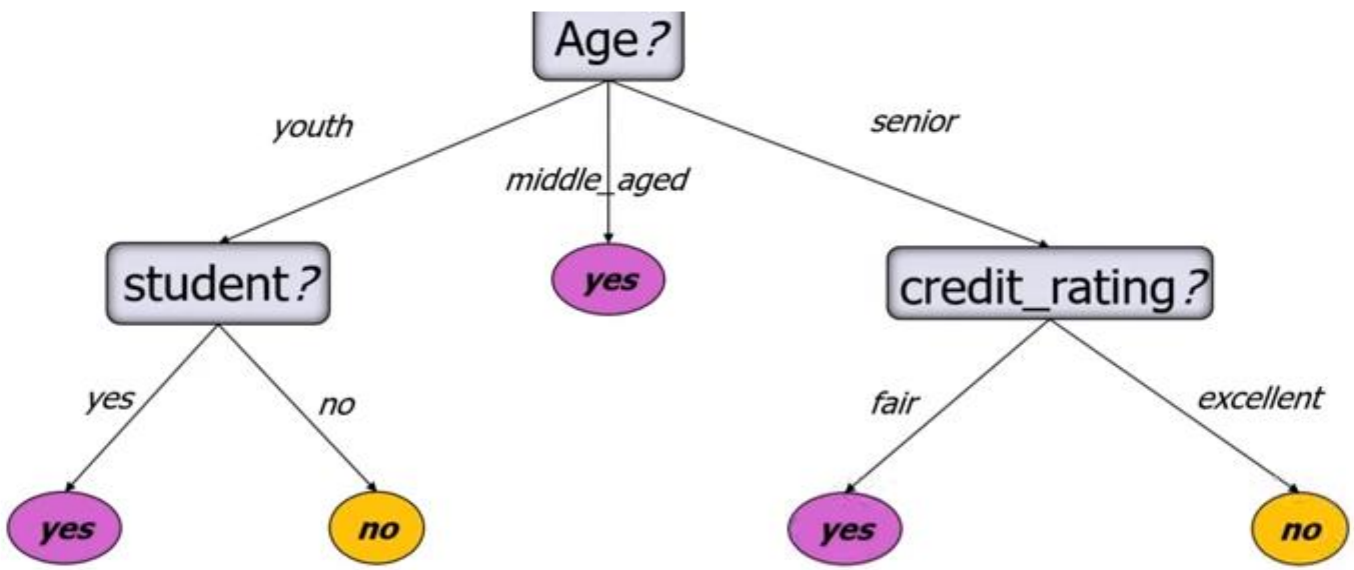
$$\mathbf{GainRatio(student)} = \frac{0.02}{0.9709} = \mathbf{0.0205}$$

Decision Tree C4.5



income	student	class
medium	no	yes
low	yes	yes
medium	yes	yes

income	student	class
low	yes	no
medium	no	no



CART Algorithm

- It is observed that information gain measure used in ID3 is biased towards test with many outcomes, that is, it prefers to select attributes having a large number of values.

- L. Breiman, J. Friedman, R. Olshen and C. Stone in 1984 proposed an algorithm to build a binary decision tree also called CART decision tree.
  - CART stands for **Classification and Regression Tree**
  - In fact, invented independently at the same time as ID3 (1984).
  - ID3 and CART are two cornerstone algorithms spawned a flurry of work on decision tree induction.

- CART is a technique that generates a **binary decision tree;** That is, unlike ID3, in CART, for each node only two children is created.

- ID3 uses Information gain as a measure to select the best attribute to be splitted, whereas CART do the same but using another measurement called **Gini index** . It is also known as **Gini Index of Diversity** and is denote as γ.

Gini Index of Diversity

Definition 9.6: **Gini Index**

Suppose, D is a training set with size $|D|$ and $C = \{c_1, c_2, \ldots, c_k\}$ be the set of k classifications and $A = \{a_1, a_2, \ldots, a_m\}$ be any attribute with m different values of it. Like entropy measure in ID3, CART proposes Gini Index (denoted by G) as the measure of impurity of D. It can be defined as follows.

$$G(D) = 1 - \sum_{i=1}^{k} p_i^2$$

where p_i is the probability that a tuple in D belongs to class c_i and p_i can be estimated as

$$p_i = \frac{|C_{i,D}|}{D}$$

where $|C_{i,D}|$ denotes the number of tuples in D with class c_i.

Gini Index of Diversity

Note

- $G(D)$ measures the "impurity" of data set D.
- The smallest value of $G(D)$ is zero
  - which it takes when all the classifications are same.
- It takes its largest value $= 1 - \frac{1}{k}$
  - when the classes are evenly distributed between the tuples, that is the frequency of each class is $\frac{1}{k}$.

Gini Index of Diversity

Definition 9.7: **Gini Index of Diversity**

Suppose, a binary partition on A splits D into D_1 and D_2, then the weighted average Gini Index of splitting denoted by $G_A(D)$ is given by

$$G_A(D) = \frac{|D_1|}{D}.G(D_1) + \frac{|D_2|}{D}.G(D_2)$$

This binary partition of D reduces the impurity and the reduction in impurity is measured by

$$\gamma(A, D) = G(D) - G_A(D)$$

Gini Index of Diversity and CART

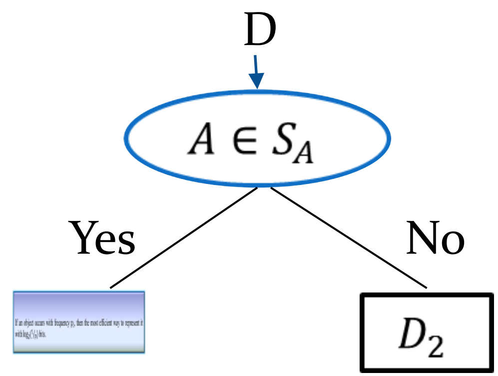
- This $\gamma(A, D)$ is called the Gini Index of diversity.
- It is also called as "impurity reduction".
- The attribute that maximizes the reduction in impurity (or equivalently, has the minimum value of $G_A(D)$) is selected for the attribute to be splitted.

n-ary Attribute Values to Binary Splitting

- The CART algorithm considers a binary split for each attribute.
- We shall discuss how the same is possible for attribute with more than two values.
- **Case 1: Discrete valued attributes**
- Let us consider the case where A is a discrete-valued attribute having m discrete values $a_1, a_2, \ldots, a_m$.
- To determine the best binary split on A, we examine all of the possible subsets say 2^A of A that can be formed using the values of A.
- Each subset $S_A \in 2^A$ can be considered as a binary test for attribute A of the form "$A \in S_A$?".

n-ary Attribute Values to Binary Splitting

- Thus, given a data set D, we have to perform a test for an attribute value A like



- This test is satisfied if the value of A for the tuples is among the values listed in S_A.
- If A has m distinct values in D, then there are 2^m possible subsets, out of which the empty subset $\{\}$ and the power set $\{a_1, a_2, \ldots, a_n\}$ should be excluded (as they really do not represent a split).
- Thus, there are $2^m - 2$ possible ways to form two partitions of the dataset D, based on the binary split of A.

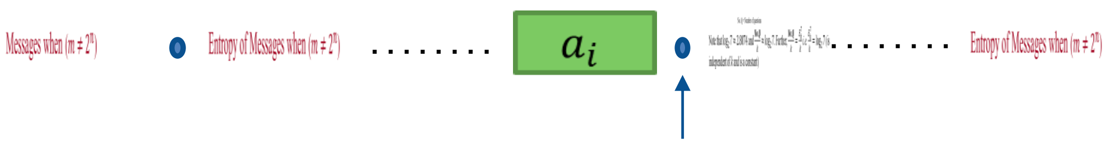
n-ary Attribute Values to Binary Splitting

Case2: Continuous valued attributes

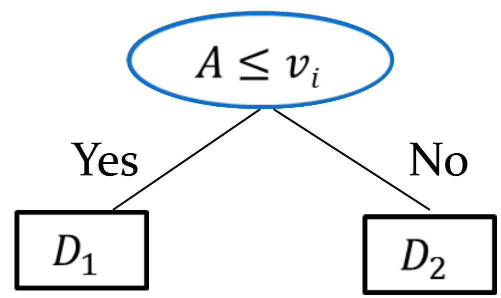
- For a continuous-valued attribute, each possible split point must be taken into account.

- The strategy is similar to that followed in ID3 to calculate information gain for the continuous –valued attributes.

- According to that strategy, the mid-point between a_i and a_{i+1} , let it be v_i, then



$$v_i = \frac{a_i + a_{i+1}}{2}$$



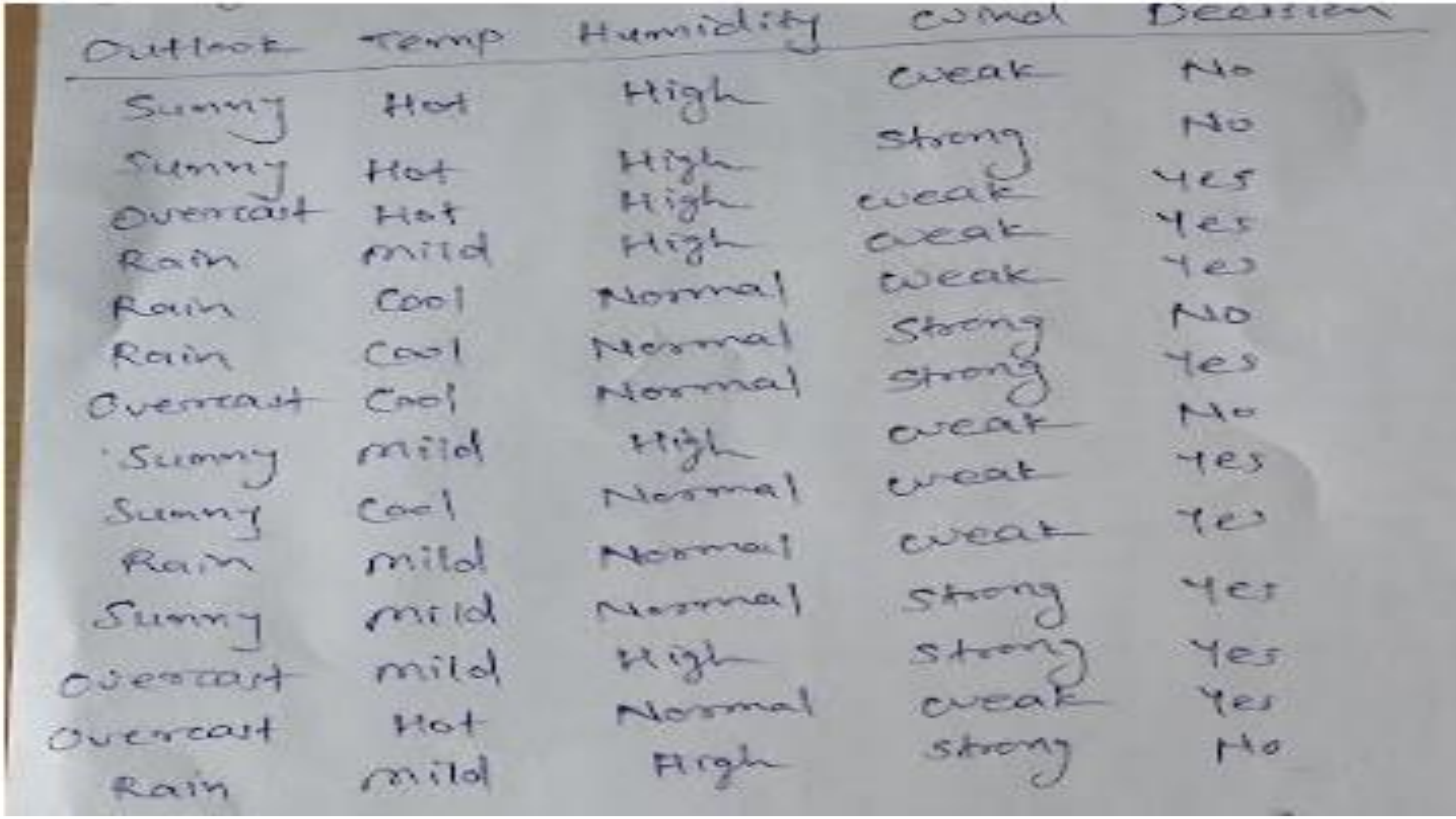
n-ary Attribute Values to Binary Splitting

- Each pair of (sorted) adjacent values is taken as a possible split-point say v_i.
- D_1 is the set of tuples in D satisfying $A \leq v_i$ and D_2 in the set of tuples in D satisfying $A > v_i$.
- The point giving the minimum Gini Index $G_A(D)$ is taken as the split-point of the attribute A.

Note

- The attribute A and either its splitting subset S_A (for discrete-valued splitting attribute) or split-point v_i (for continuous valued splitting attribute) together form the splitting criteria.

Decision Tree using CART



Outlook	Temp	Humidity	wind	Decision
Sunny	Hot	High	weak	No
Sunny	Hot	High	strong	No
Overcast	Hot	High	weak	Yes
Rain	mild	High	weak	Yes
Rain	Cool	Normal	weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	strong	Yes
Sunny	mild	High	weak	No
Sunny	Cool	Normal	weak	Yes
Rain	mild	Normal	weak	Yes
Sunny	mild	Normal	strong	Yes
Overcast	mild	High	strong	Yes
Overcast	Hot	Normal	weak	Yes
Rain	mild	High	strong	No

- To given dataset there are 14 instances of golf playing decision based on outlook, temperature, humidity and wind factor.

Decision Tree using CART

⇒ Gini index is a metric for classification task in CART.

$$\left.\begin{aligned} \text{Gini index (Attribute = value)} &= 1 - \sum_{i=1}^{N} (p_i)^2 \\ \text{Gini Index (Attribute)} &= \sum_{v = values} p_v * GI(v) \end{aligned}\right\}$$

→ Outlook

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	[redacted]

Decision Tree using CART

...g the 100% confidence.

$$Gini(outlook = sunny) = 1 - (2/5)^2 - (3/5)^2$$

$$\boxed{= 0.48}$$

$$Gini(outlook = overcast) = 1 - (4/4)^2 - (0/4)^2$$

$$\boxed{= 0}$$

$$Gini(outlook = Rain) = 1 - (3/5)^2 - (2/5)^2$$

$$\boxed{= 0.48}$$

Now calculate weighted sum of Gini indices

$$Gini(outlook) = (5/14) * 0.48 + (4/14) * 0 + (5/14) * 0.48$$

$$\boxed{= 0.342}$$

Decision Tree using CART

Temperature	Yes	No	No. of Instances
Hot	2	2	4
cool	3	1	4
mild	4	2	6

$$Gini(Temp = Hot) = 1 - (2/4)^2 - (2/4)^2 \boxed{= 0.5}$$

$$Gini(Temp = cool) = 1 - (3/4)^2 - (1/4)^2 \boxed{= 0.375}$$

$$Gini(Temp = mild) = 1 - (4/6)^2 - (2/6)^2 \boxed{= 0.445}$$

Calculate weighted sum of Gini Index

$$Gini(Temp) = (4/14) * 0.5 + (4/14) * 0.375 + (6/14) * 0.445$$

$$\boxed{= 0.439}$$

Decision Tree using CART

Humidity	Yes	No	No. of Instances
High	3	4	7
Normal	6	1	7

$$Gini(Humidity = High) = 1 - (3/7)^2 - (4/7)^2 = 0.489$$

$$Gini(Humidity = Normal) = 1 - (6/7)^2 - (1/7)^2 = 0.244$$

$$Gini(Humidity) = (7/14) * 0.489 + (7/14) * 0.244$$

$$\boxed{= 0.367}$$

Decision Tree using CART

Wind	Yes	No	No. of instances
weak	6	2	8
strong	3	3	6

$$Gini(wind = weak) = 1 - (6/8)^2 - (2/8)^2 = 0.375$$

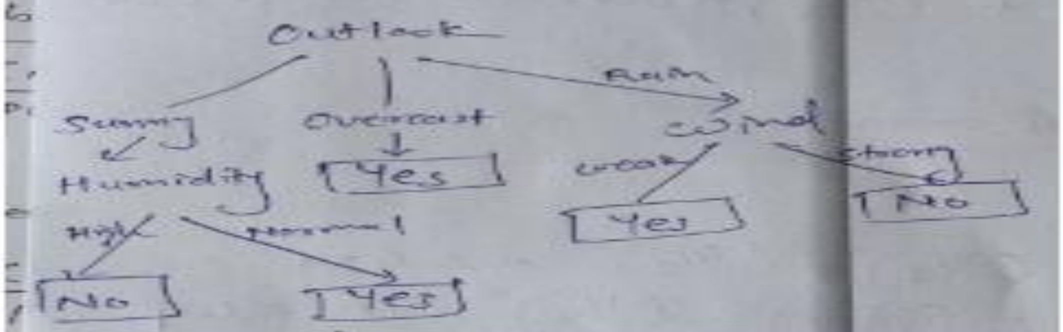
$$Gini(wind = strong) = 1 - (3/6)^2 - (3/6)^2 = 0.5$$

$$Gini(wind) = (8/14) * 0.375 + (6/14) * 0.5$$

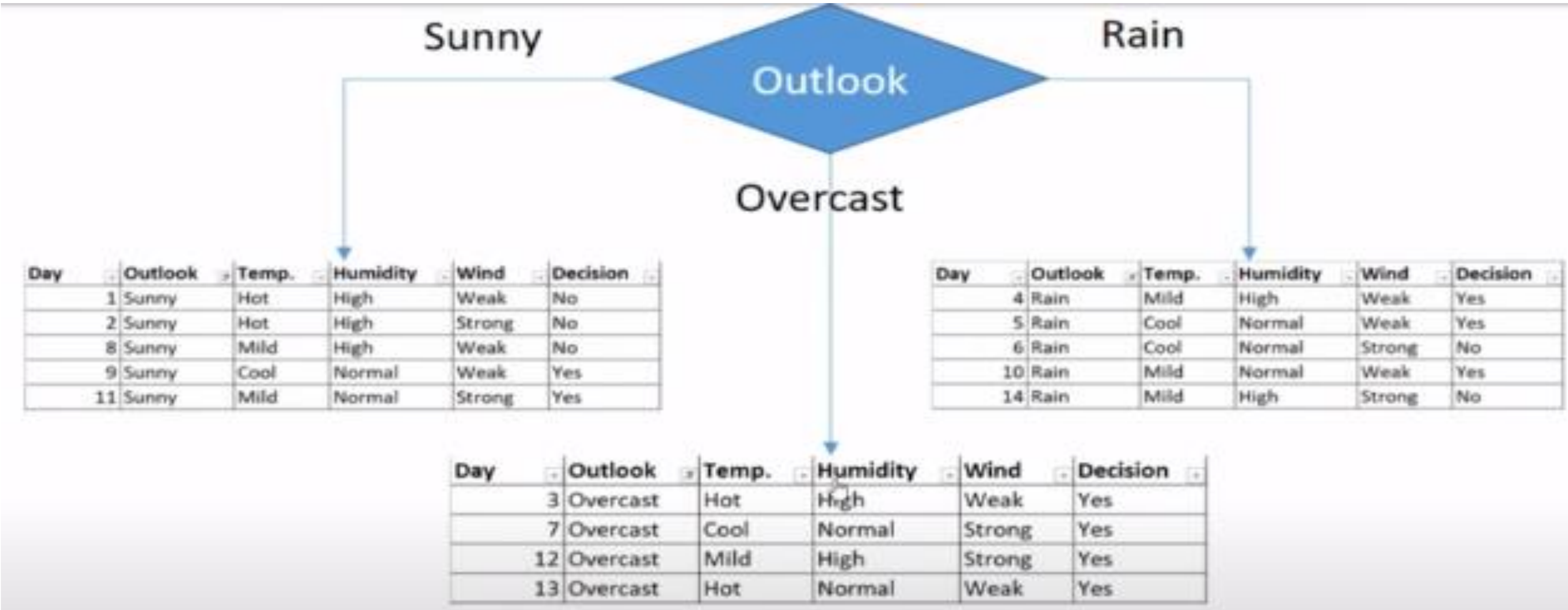
$$\boxed{= 0.428}$$

Feature	Gini Index
Outlook	0.342
Temp.	0.439
Humidity	0.367
wind	0.428

least GI ⇒ Root node



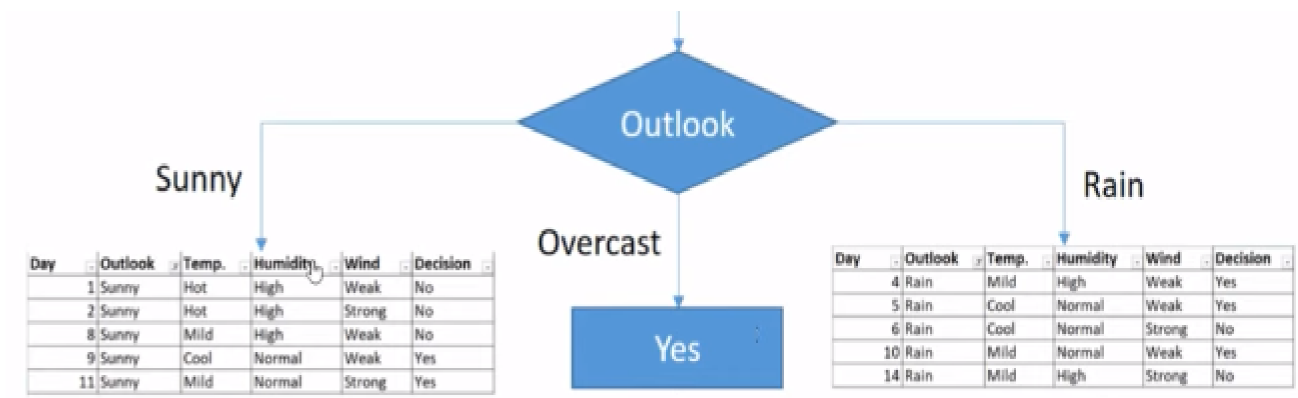
Decision Tree using CART



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Tree is over for overcast outlook leaf

Decision Tree using CART

We will apply same principles to those sub datasets in the following steps.
Focus on the sub dataset for sunny outlook. We need to find the Gini Index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

Gini(Outlook=Sunny and Temp.=Hot) = $1 - (0/2)^2 - (2/2)^2 = 0$
Gini(Outlook=Sunny and Temp.=Cool) = $1 - (1/1)^2 - (0/1)^2 = 0$
Gini(Outlook=Sunny and Temp.=Mild) = $1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$

Gini(Outlook=Sunny and Temp.) = $(2/5)*0 + (1/5)*0 + (2/5)*0.5 = 0.2$

Decision Tree using CART

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

Gini(Outlook=Sunny and Humidity=High) = $1 - (0/3)^2 - (3/3)^2 = 0$
Gini(Outlook=Sunny and Humidity=Normal) = $1 - (2/2)^2 - (0/2)^2 = 0$
Gini(Outlook=Sunny and Humidity) = (3/5)*0 + (2/5)*0 = 0

Gini of wind for sunny outlook

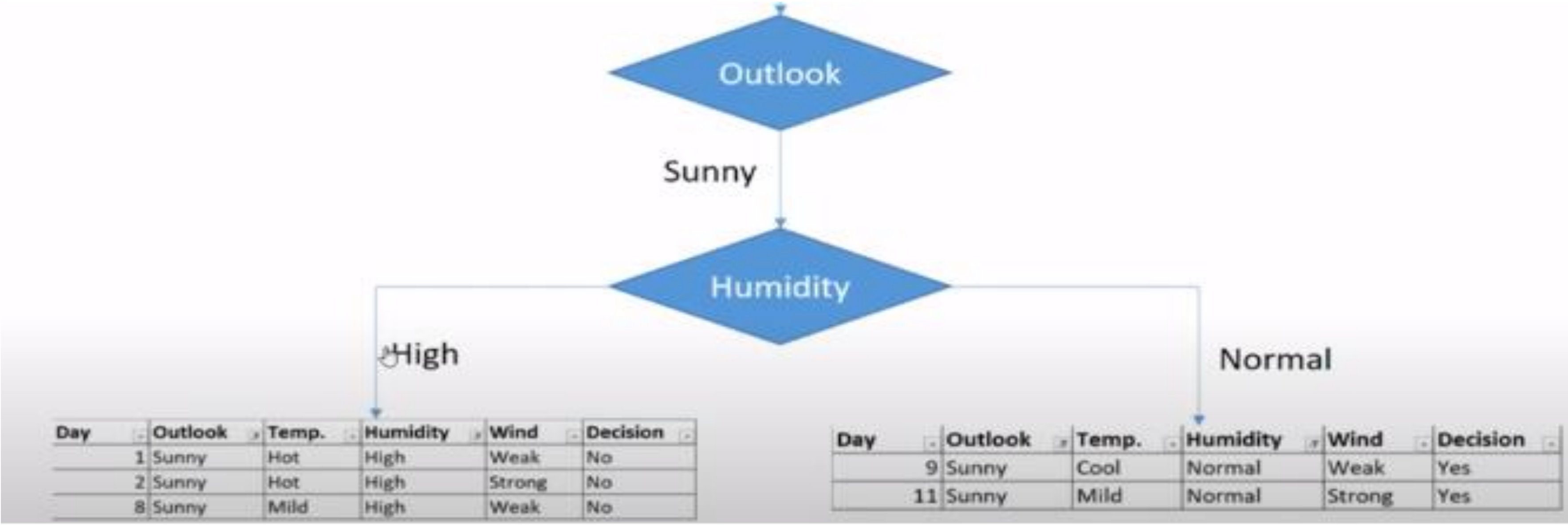
Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

Gini(Outlook=Sunny and Wind=Weak) = $1 - (1/3)^2 - (2/3)^2 = 0.266$
Gini(Outlook=Sunny and Wind=Strong) = $1 - (1/2)^2 - (1/2)^2 = 0.2$
Gini(Outlook=Sunny and Wind) = (3/5)*0.266 + (2/5)*0.2 = 0.466

Decision for sunny outlook
We've calculated Gini Index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

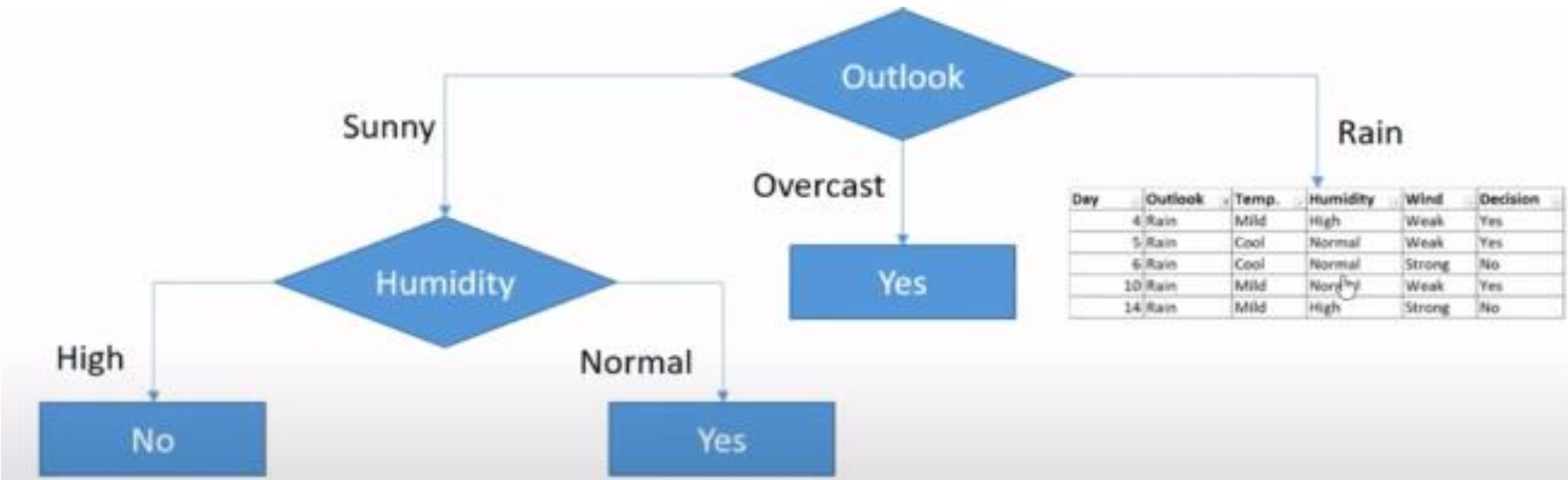
Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

Decision Tree using CART



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decision Tree using CART

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Gini of temperature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

Gini(Outlook=Rain and Temp.=Cool) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Temp.=Mild) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Temp.) = $(2/5)*0.5 + (3/5)*0.444 = 0.466$

Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

Gini(Outlook=Rain and Humidity=High) = $1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Humidity=Normal) = $1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Humidity) = $(2/5)*0.5 + (3/5)*0.444 = 0.466$

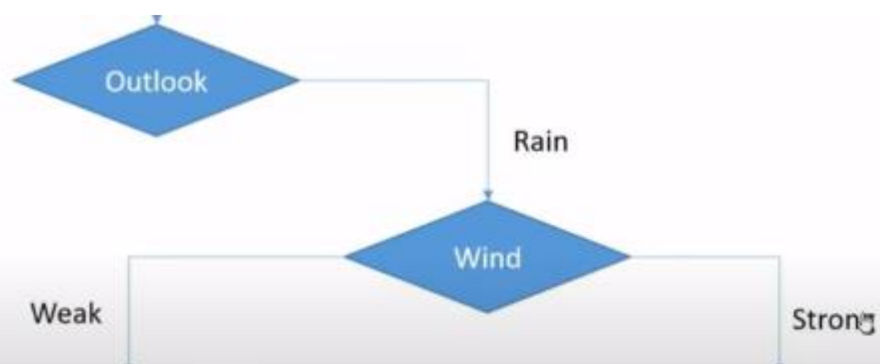
Decision Tree using CART

Gini of wind for rain outlook

Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak) = $1 - (3/3)^2 - (0/3)^2 = 0$
Gini(Outlook=Rain and Wind=Strong) = $1 - (0/2)^2 - (2/2)^2 = 0$
Gini(Outlook=Rain and Wind) = $(3/5)*0 + (2/5)*0 = 0$

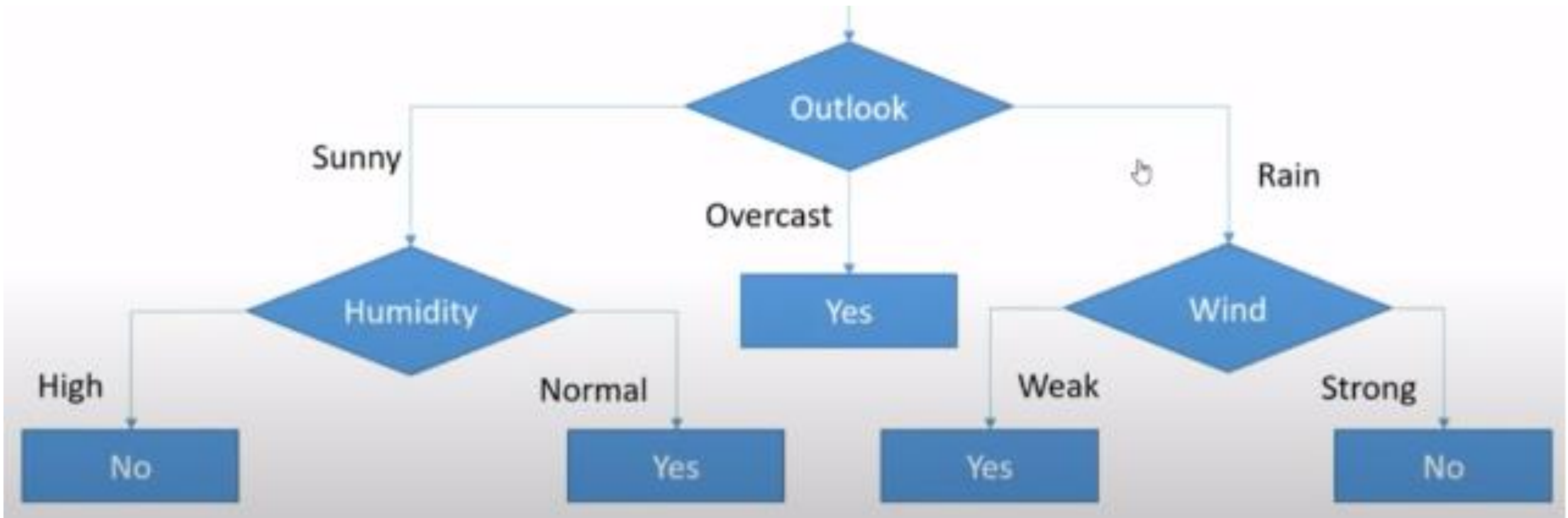
Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

Decision Tree using CART



CART Algorithm : Illustration

Example 2 : CART Algorithm

Suppose we want to build decision tree for the data set EMP as given in the table below.

Age
Y : young
M : middle-aged
O : old

Salary
L : low
M : medium
H : high

Job
G : government
P : private

Performance
A : Average
E : Excellent

Class : Select
Y : yes
N : no

Tuple#	Age	Salary	Job	Performance	Select
1	Y	H	P	A	N
2	Y	H	P	E	N
3	M	H	P	A	Y
4	O	M	P	A	Y
5	O	L	G	A	Y
6	O	L	G	E	N
7	M	L	G	E	Y
8	Y	M	P	A	N
9	Y	L	G	A	Y
10	O	M	G	A	Y
11	Y	M	G	E	Y
12	M	M	P	E	Y
13	M	H	G	A	Y
14	O	M	P	E	N

CART Algorithm : Illustration

For the EMP data set,

$$G(EMP) = 1 - \sum_{i=1}^{2} p_i^2$$

$$= 1 - \left[\left(\frac{9}{14} \right)^2 + \left(\frac{5}{14} \right)^2 \right]$$

$$= \mathbf{0.4592}$$

Now let us consider the calculation of $G_A(EMP)$ for **Age**, **Salary**, **Job** and **Performance**.

CART Algorithm : Illustration

Attribute of splitting: Age

The attribute age has three values, namely Y, M and O. So there are 6 subsets, that should be considered for splitting as:

$$\underset{age_1}{\{Y\}}, \quad \underset{age_2}{\{M\}}, \quad \underset{age_3}{\{O\}}, \quad \underset{age_4}{\{Y,M\}}, \quad \underset{age_5}{\{Y,O\}}, \quad \underset{age_6}{\{M,O\}}$$

$$G_{age_1}(D) = \frac{5}{14} * \left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right) + \frac{9}{14}\left(1 - \left(\frac{6}{14}\right)^2 - \left(\frac{8}{14}\right)^2\right) = \mathbf{0.4862}$$

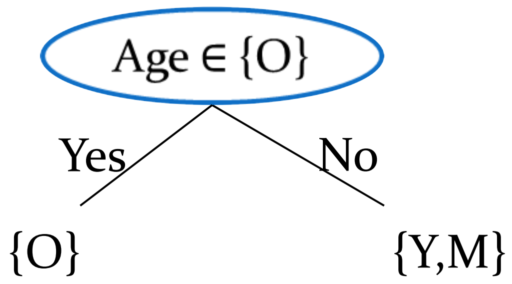
$G_{age_2}(D) = \ \mathbf{?}$

$G_{age_3}(D) = \ \mathbf{?}$

$G_{age_4}(D) = G_{age_3}(D)$

$G_{age_5}(D) = G_{age_2}(D)$

$G_{age_6}(D) = G_{age_1}(D)$



The best value of Gini Index while splitting attribute Age is $\gamma(Age_3, D) = \mathbf{0.3750}$

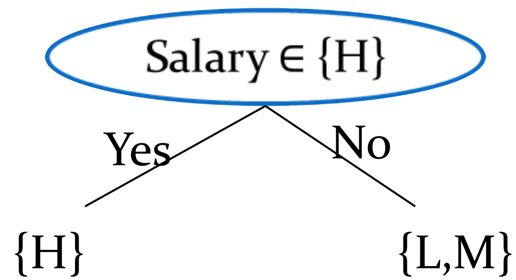
CART Algorithm : Illustration

Attribute of Splitting: Salary

The attribute salary has three values namely *L*, *M* and *H*. So, there are 6 subsets, that should be considered for splitting as:

$$\underset{sal_1}{\{L\}}, \quad \underset{sal_2}{\{M,H\}}, \quad \underset{sal_3}{\{M\}}, \quad \underset{sal_4}{\{L,H\}}, \quad \underset{sal_5}{\{H\}}, \quad \underset{sal_6}{\{L,M\}}$$

$$G_{sal_1}(D) = G_{sal_2}(D) = 0.3000$$
$$G_{sal_3}(D) = G_{sal_4}(D) = 0.3150$$
$$G_{sal_5}(D) = G_{sal_6}(D) = 0.4508$$



$$\gamma(salary_{(5,6)}, D) = 0.4592 - 0.4508 = 0.0084$$

CART Algorithm : Illustration

Attribute of Splitting: job

Job being the binary attribute , we have

$$G_{job}(D) = \frac{7}{14}G(D_1) + \frac{7}{14}G(D_2)$$

$$= \frac{7}{14}\left[1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2\right] + \frac{7}{14}\left[1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2\right] = ?$$

$\gamma(job, D) = ?$

CART Algorithm : Illustration

Attribute of Splitting: Performance

Job being the binary attribute , we have

$G_{Performance}(D) =$ **?**

$\gamma(performance, D) =$ **?**

Out of these $\gamma(salary, D)$ gives the maximum value and hence, the attribute **Salary** would be chosen for splitting subset $\{M, H\}$ or $\{L\}$.

Note:

It can be noted that the procedure following "information gain" calculation (i.e. $\propto (A, D)$) and that of "impurity reduction" calculation (i.e. $\gamma(A, D)$) are near about.

Notes on Decision Tree Induction algorithms

1. **Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms employ a heuristic based approach to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.

1. **Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.

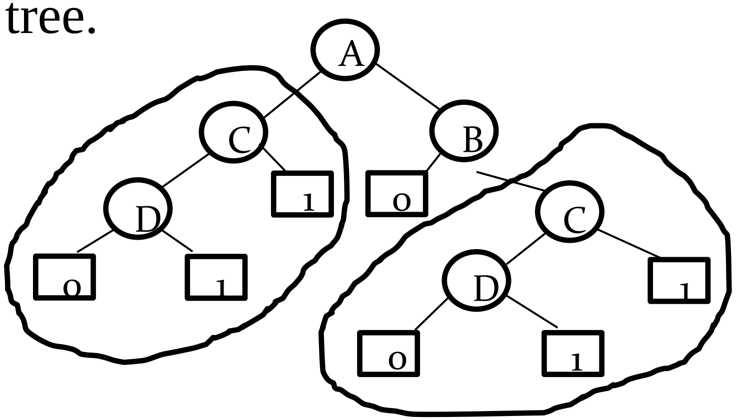
1. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.

1. **Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of O(d), where d is the maximum depth of the tree.

Notes on Decision Tree Induction algorithms

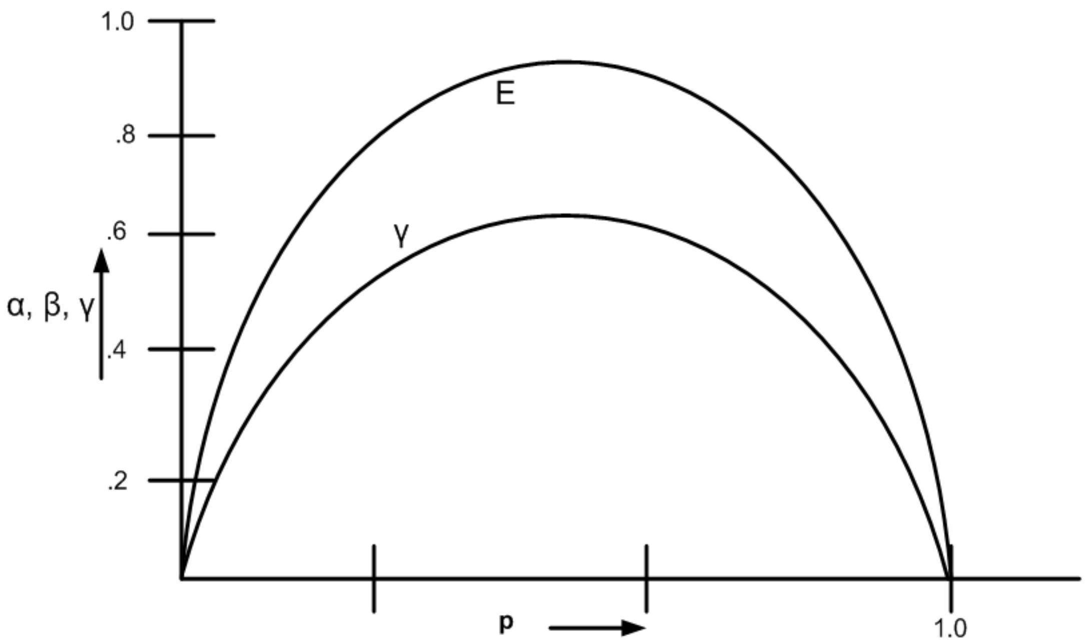
5. **Data Fragmentation Problem:** Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.

5. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



Notes on Decision Tree Induction algorithms

7. **Decision tree equivalence:** The different splitting criteria followed in different decision tree induction algorithms have little effect on the performance of the algorithms. This is because the different heuristic measures (such as information gain (α), Gini index (γ) and Gain ratio (β) are quite consistent with each other); also see the figure below.



Bayesian Classification

Bayesian Classification

→ Bayesian classifiers are statistical classifiers

→ They can predict the probabilities of class items (like it gives the probability that a given data item belongs to that class or not.)

Baye's Theorem

$$P(A/B) = \frac{\overbrace{P(B/A)}^{\text{likelihood}} \cdot \overbrace{P(A)}^{\text{prior}}}{\underbrace{P(B)}_{\text{marginal (Probability of evidence)}}}$$

posterior

{ gives a probability of an event based on prior knowledge of conditions

Conditional probability

① $P(A/B) = \frac{P(A \cap B)}{P(B)}$

probability of event A given that event B is true / already occured

② $P(B/A) = \frac{P(B \cap A)}{P(A)}$

Bayesian Classification

$P(A|B) \rightarrow$ Posterior Prob.

Find the probability of hypothesis given that we have observed some evidence

$P(B|A)$ - likelihood

Probability of evidence given that hypothesis is true.

$P(A)$ - Prior Prob.

Probability of hypothesis before observing / considering evidence.

Ex $P(\text{King} \mid \text{face}) = ?$

$$P(\text{King} \mid \text{face}) = \frac{P(\text{face} \mid \text{King}) \cdot P(\text{King})}{P(\text{face})}$$

$$= \frac{1 \cdot (4/52)}{12/52} = 1/3$$

Naïve Bayes Classification

Naive Bayes Classification Algo.

Fruit = { Yellow, Sweet, long } ⇒ Given Attribute set } based on Bayes theorem with an assumption of independence among features

Fruit	Yellow	Sweet	long	Total
Orange	350	450	0	650
Banana	400	300	350	400
Others	50	100	50	150
Total	800	850	400	1200

$$P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$$

$$P(\text{Yellow} / \text{orange}) = \frac{P(\text{orange} / \text{Yellow}) \cdot P(\text{Yellow})}{P(\text{orange})}$$

$$= \frac{(350/800) \cdot (800/1200)}{650/1200} \boxed{= 0.5}$$

$P(\text{sweet} / \text{orange}) = \boxed{0.69}$

$P(\text{long} / \text{orange}) = \boxed{0}$

$P(\text{Fruit} / \text{orange}) = (0.5) * (0.69) * 0$ } orange fruit is Not possible

$\boxed{= 0}$

$P(\text{Fruit} / \text{Banana}) = \underbrace{1}_{P(\text{Yellow}/\text{Banana})} * \underbrace{0.75}_{P(\text{sweet}/\text{Banana})} * \underbrace{0.87}_{P(\text{long}/\text{Banana})}$

$\boxed{= 0.65}$

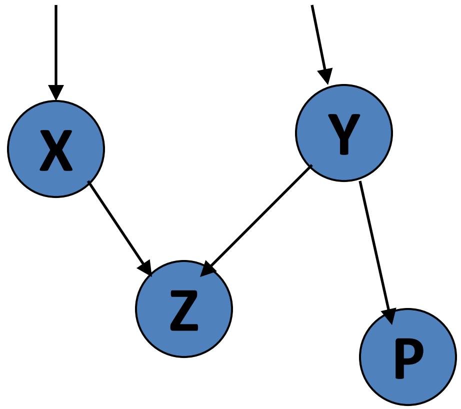
$P(\text{fruit} / \text{others}) = 0.33 * 0.66 * 0.33$

$\boxed{= 0.072}$

max / highest probability ⇒ $\boxed{\text{Banana}}$ = fruit

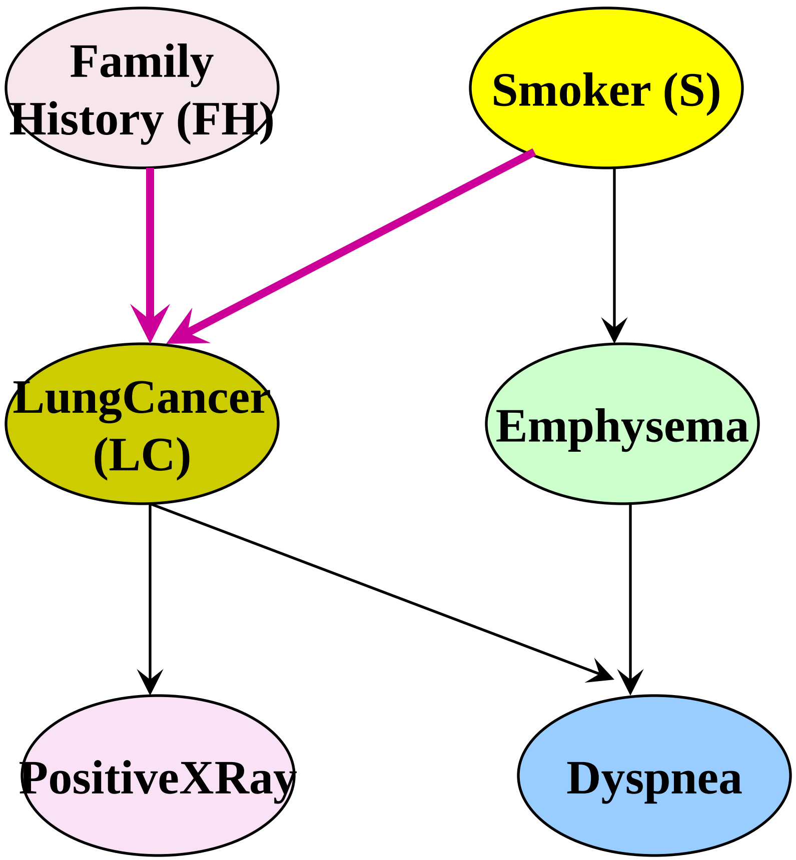
Bayesian Belief Networks

- **Bayesian belief networks** (also known as **Bayesian networks**, **probabilistic networks**): allow *class conditional independencies* between *subsets* of variables
- A (*directed acyclic*) graphical model of causal relationships
  - Represents <u>dependency</u> among the variables
  - Gives a specification of joint probability distribution



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

Bayesian Belief Network: An Example



Bayesian Belief Network

CPT: **Conditional Probability Table** for variable LungCancer:

	(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

shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of **X**, from CPT:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

Training Bayesian Networks: Several Scenarios

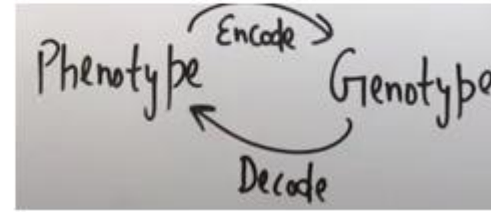
- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models,* M. Jordan, ed.. MIT Press, 1999.

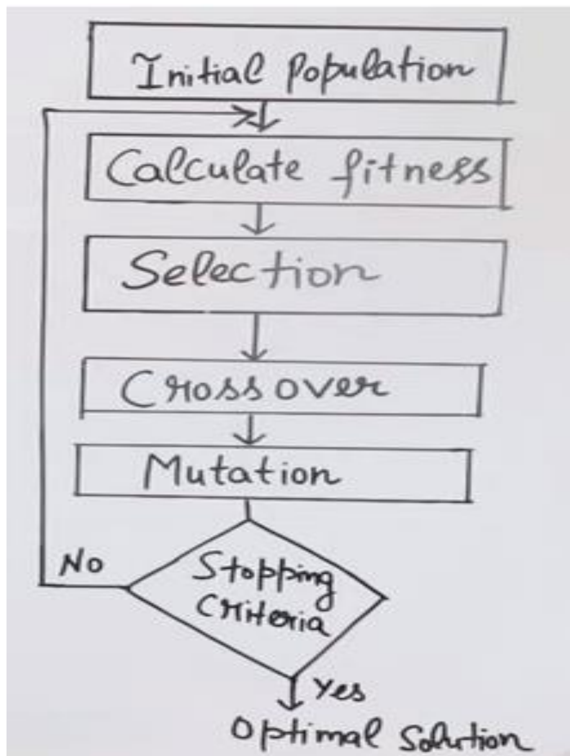
Genetic Algorithms (GA)

- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
  - If an attribute has $k > 2$ values, k bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offspring
- The *fitness of a rule* is represented by its classification accuracy on a set of training examples
- Offspring are generated by *crossover* and *mutation*
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable

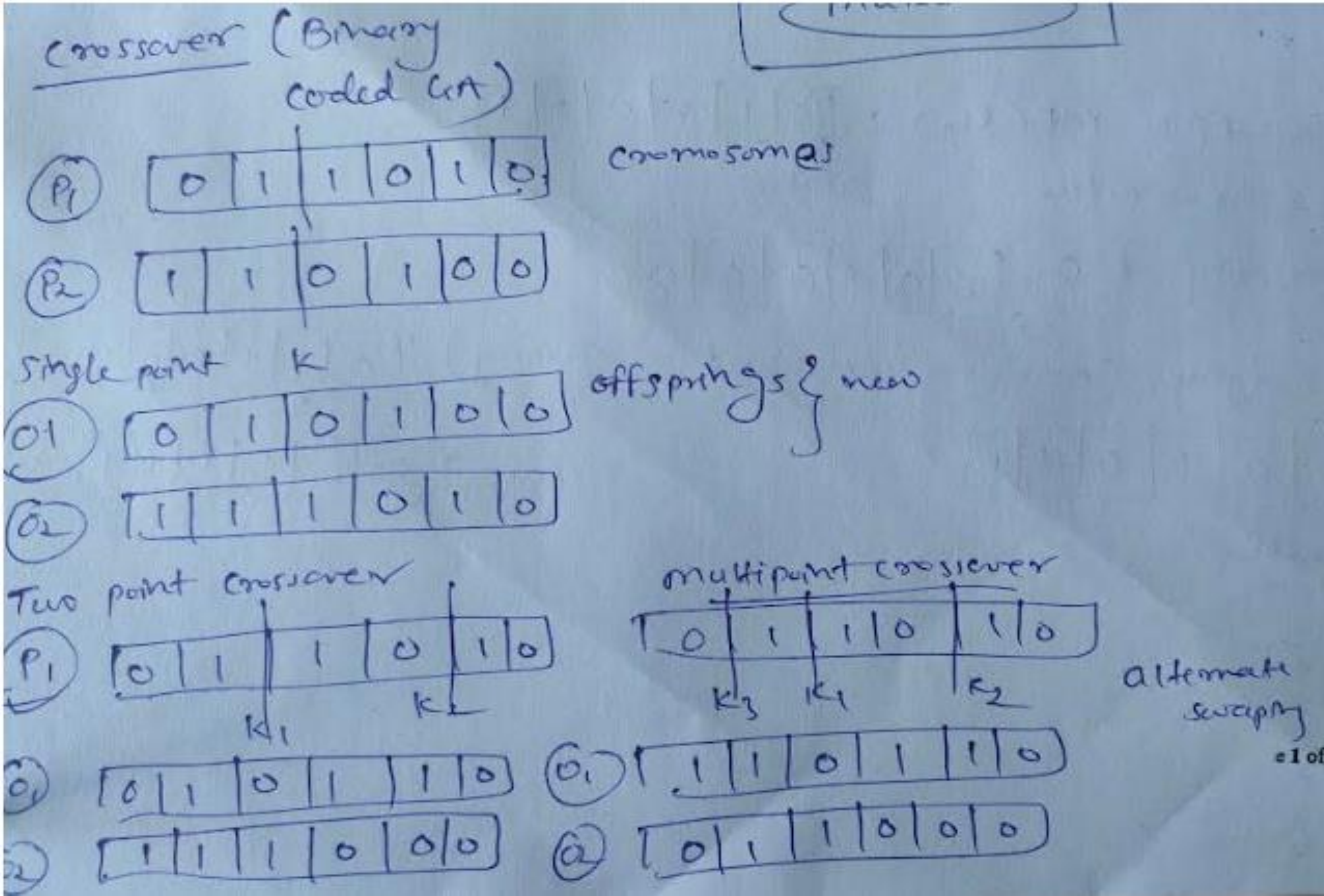
Genetic Algorithm

- Abstraction of real biological evolution
- Solve complex problems (NP-Hard)
- Focus on optimization
- Population of possible solutions for a given problem
- From a group of individuals, the best will survive





Genetic Algorithm



Genetic Algorithm

<u>mutation</u> (genetic diversity)

offspring

0	1	1	0	0	1

① <u>Flipping</u>

<u>mutation probability</u> mp(μ_p) :

0	1	0	0	0	1

as it is

↳ bit flip → yes / No

muted offspring mo :

0	0	1	0	0	0

② <u>Interchanging</u> (random selection from offspring)

0	1	1	0	0	1

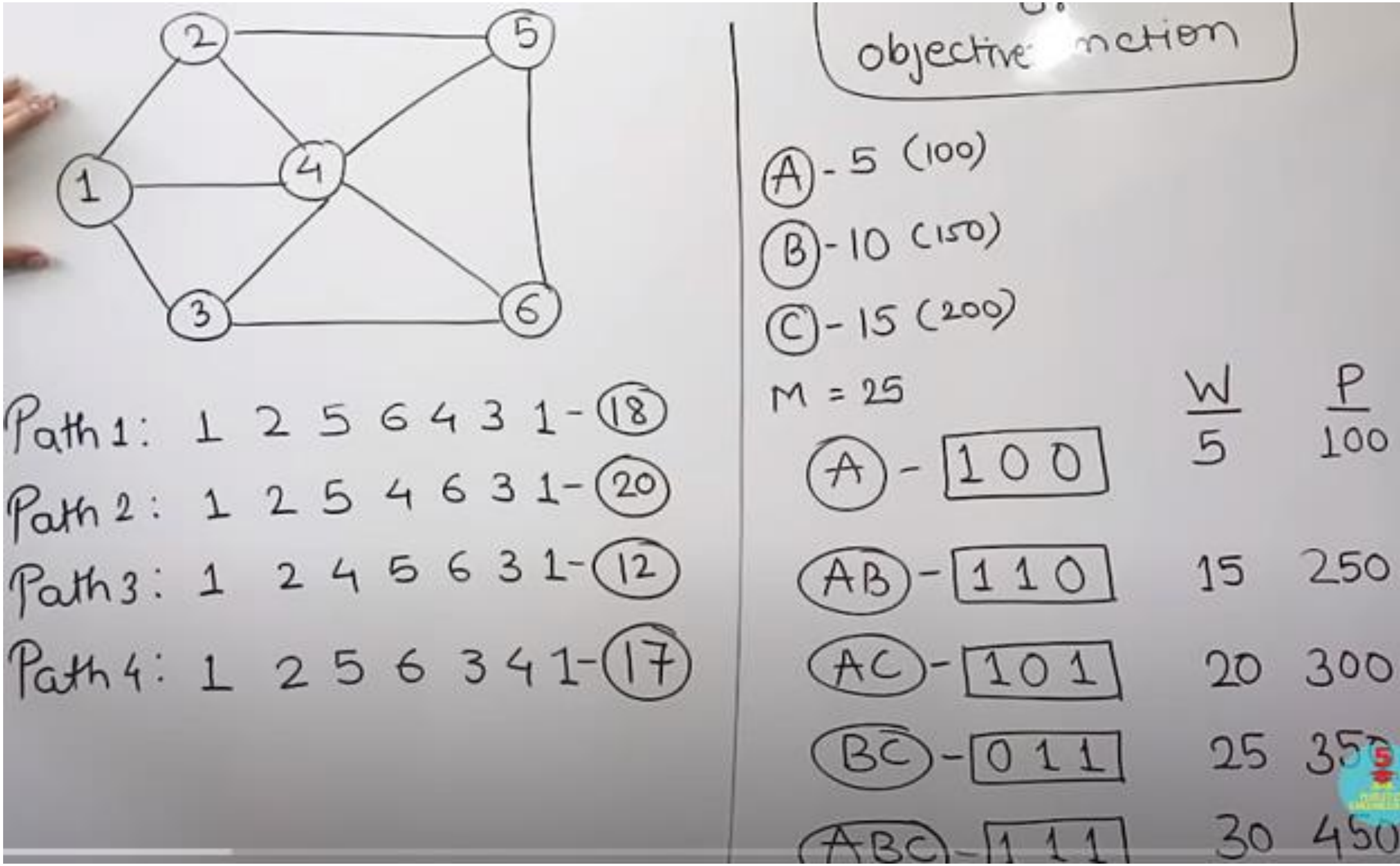
mo :

0	0	1	0	1	1

mo:

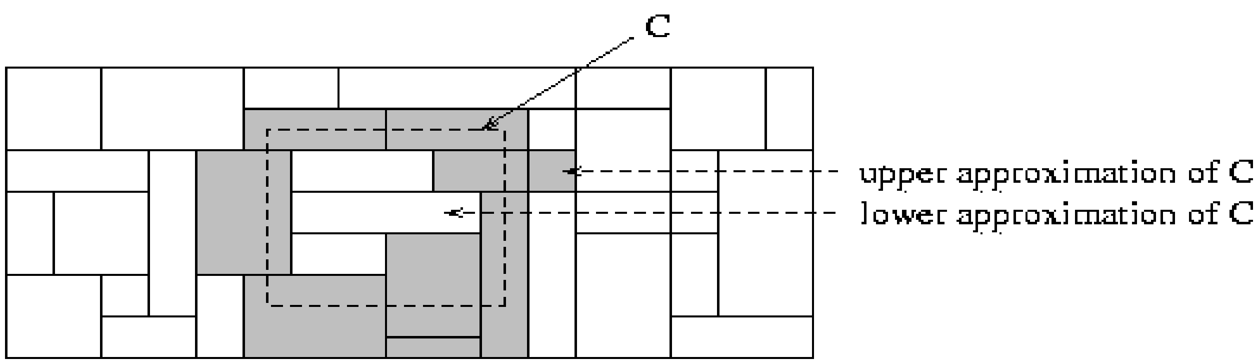
0	0	1	0	1	1

Genetic Algorithm Fitness function

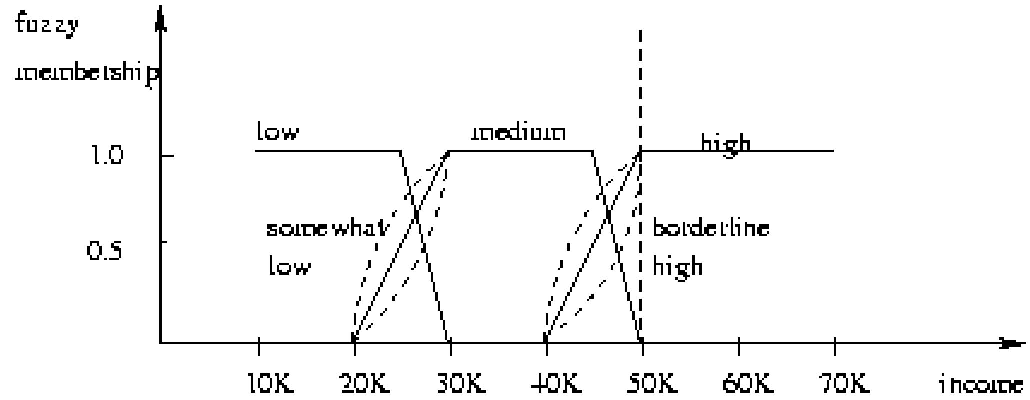


Rough Set Approach

- Rough sets are used to **approximately or “roughly” define equivalent classes**
- A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C)
- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity



Fuzzy Set Approaches



- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as in a *fuzzy membership graph*)
- Attribute values are converted to fuzzy values. Ex.:
  - Income, *x*, is assigned a fuzzy membership value to each of the discrete categories {low, medium, high}, e.g. $49K belongs to "medium income" with fuzzy value 0.15 but belongs to "high income" with fuzzy value 0.96
  - Fuzzy membership values do not have to sum to 1.
- 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

Classification by Backpropagation

Classification by Backpropagation

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or "structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs *and outputs*
  - Successful on an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Neural Network

Neural Networks:

Neural Networks mimics the human brain by learning from a training dataset and applying the learning to generalize patterns for classification and prediction.

↳ basic unit of ANN ⇒ Neuron in brain

$$\text{SUM} = X^{*} W$$

$$= x_1 {}^{*} w_1 + x_2 {}^{*} w_2 + \dots + x_n {}^{*} w_n$$

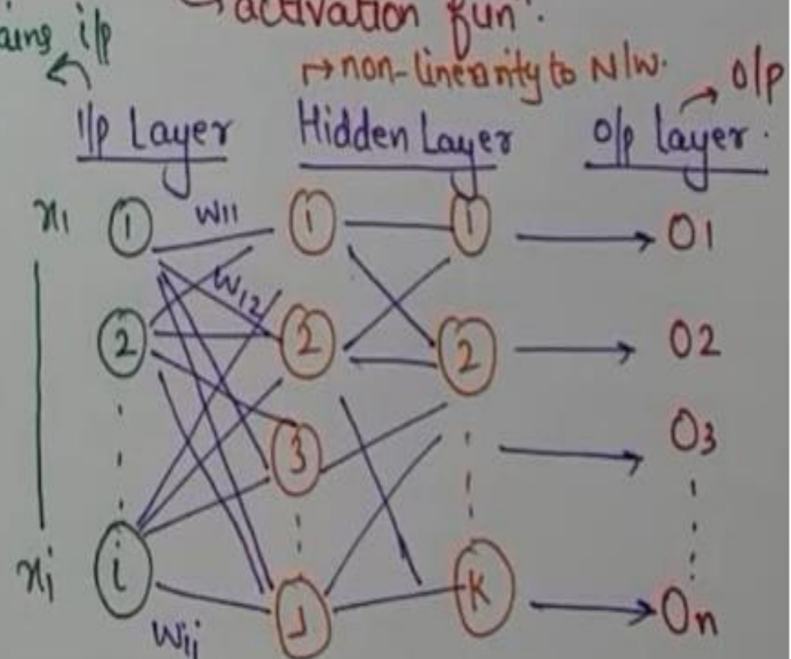
$$\boxed{\text{SUM} = \sum_{i=1}^{n} x_i w_i}$$

$$\boxed{Y(o/p) = \phi(\text{SUM})}$$

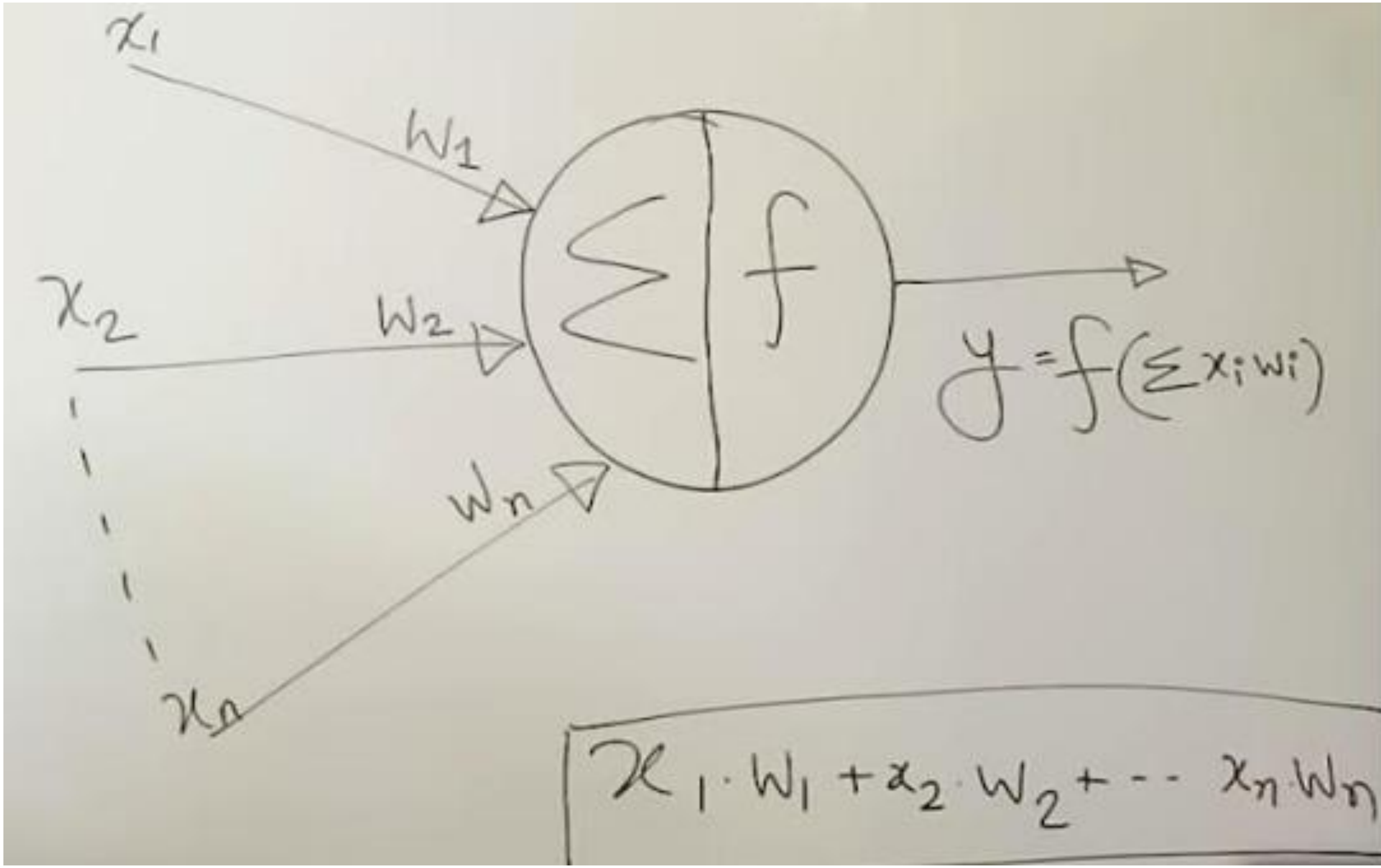
↳ activation funⁿ.

→ non-linearity to N/w.

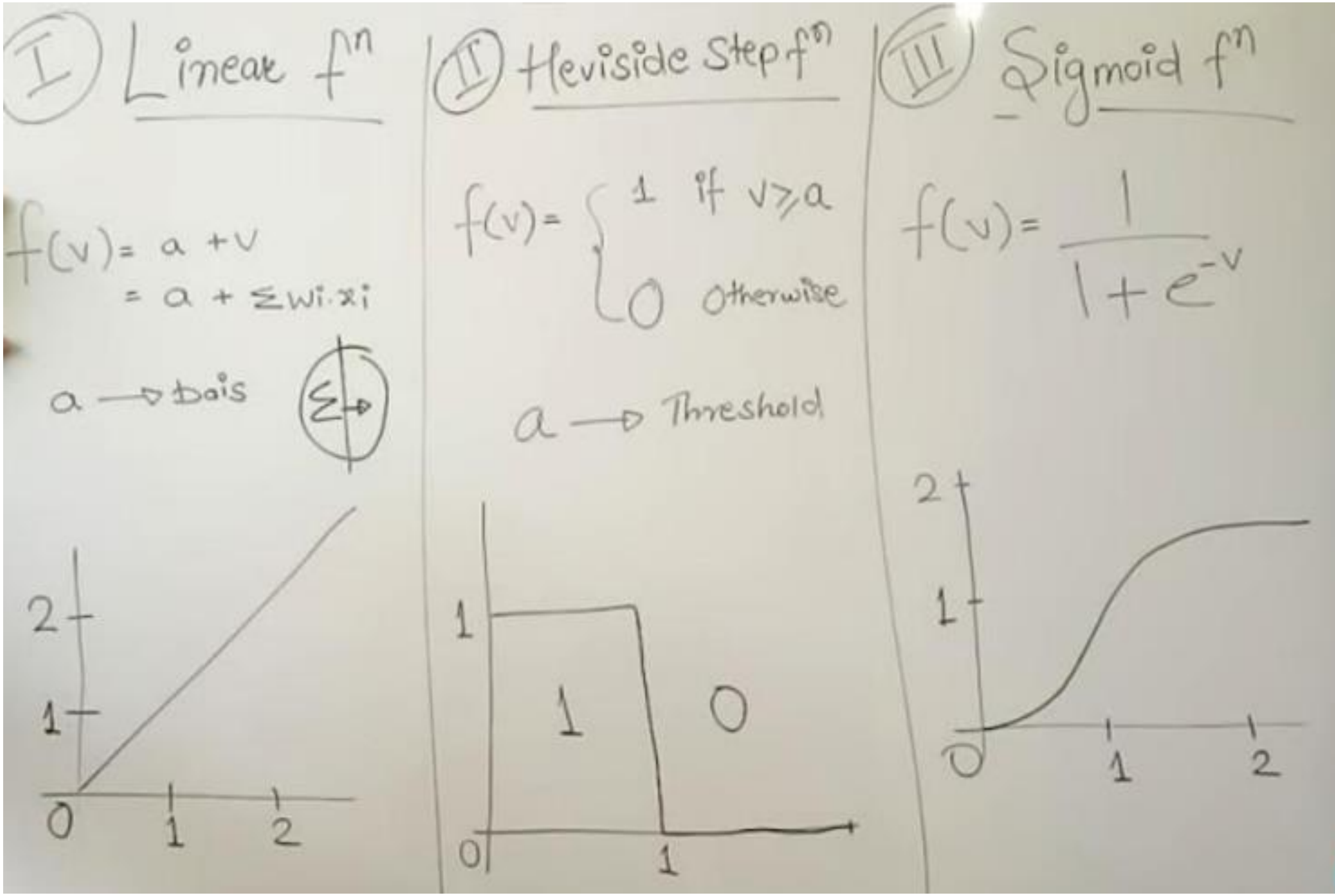
ANN is an N/w of interconnected neurons, inspired from the biological nervous system. units



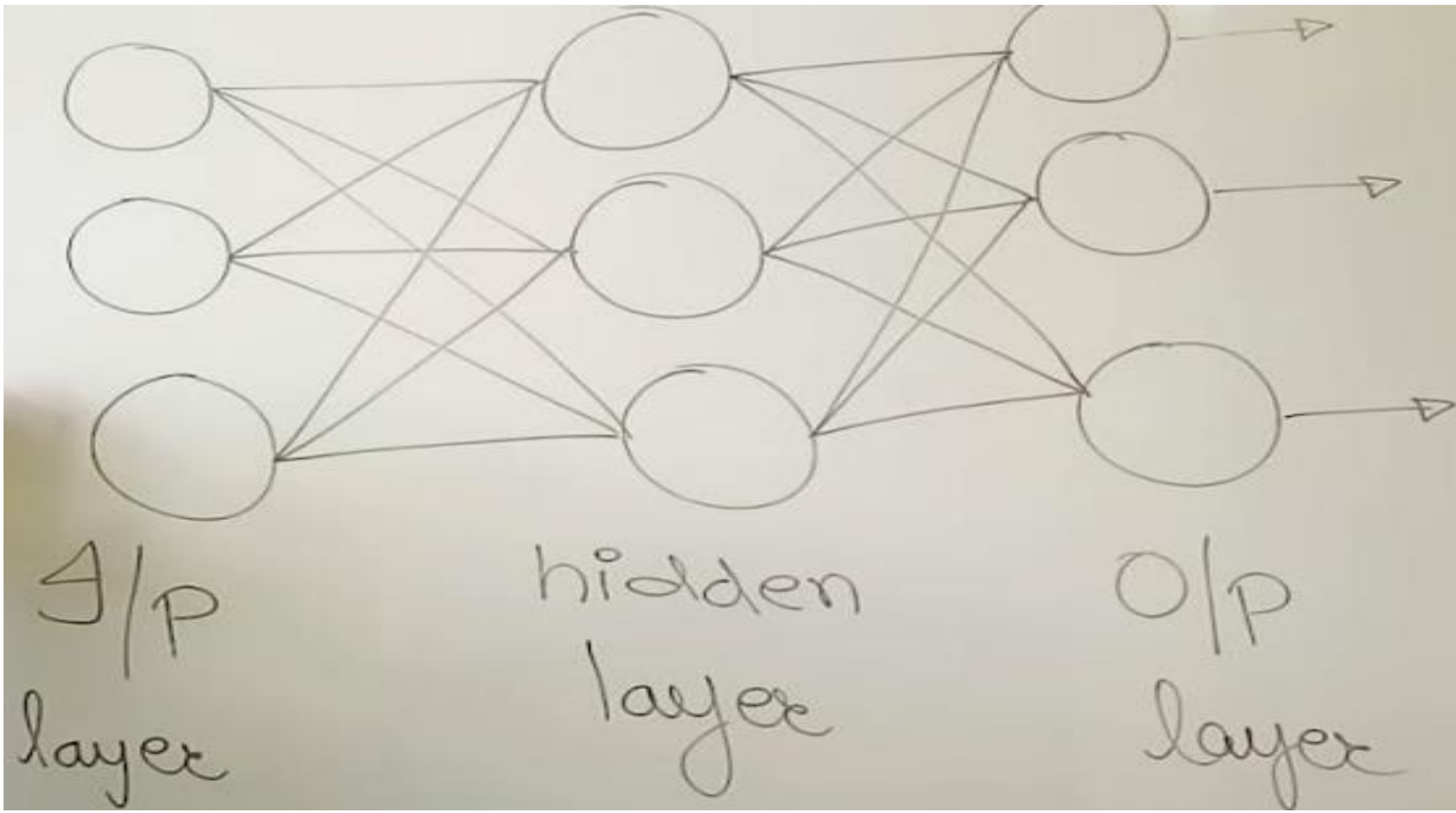
Artificial Neural Network



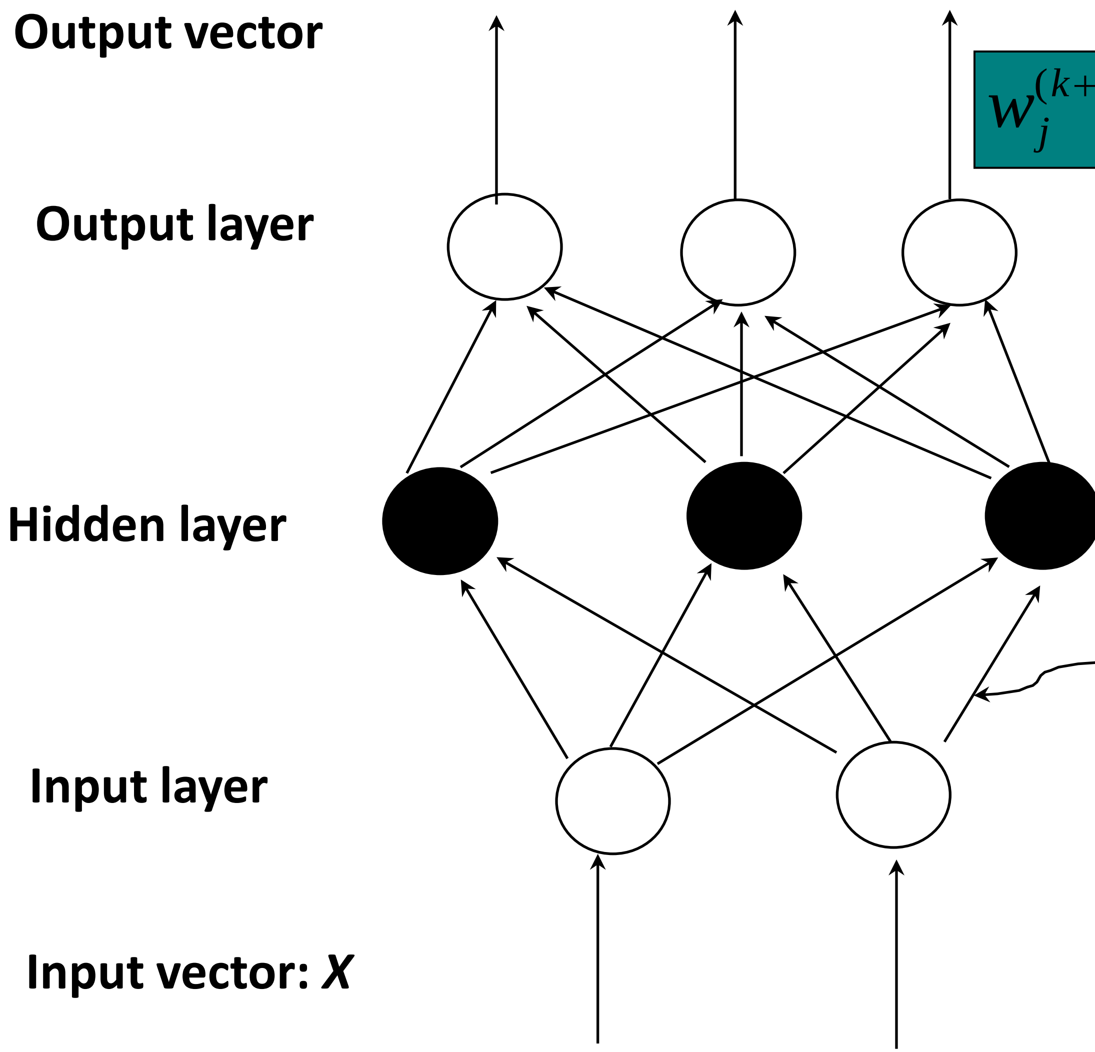
Activation Function



Feed Forward Network



A Multi-Layer Feed-Forward Neural Network



$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$

How A Multi-Layer Neural Network Works

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward**: None of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

Defining a Network Topology

- Decide the **network topology:** Specify # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalize the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value, each initialized to 0
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

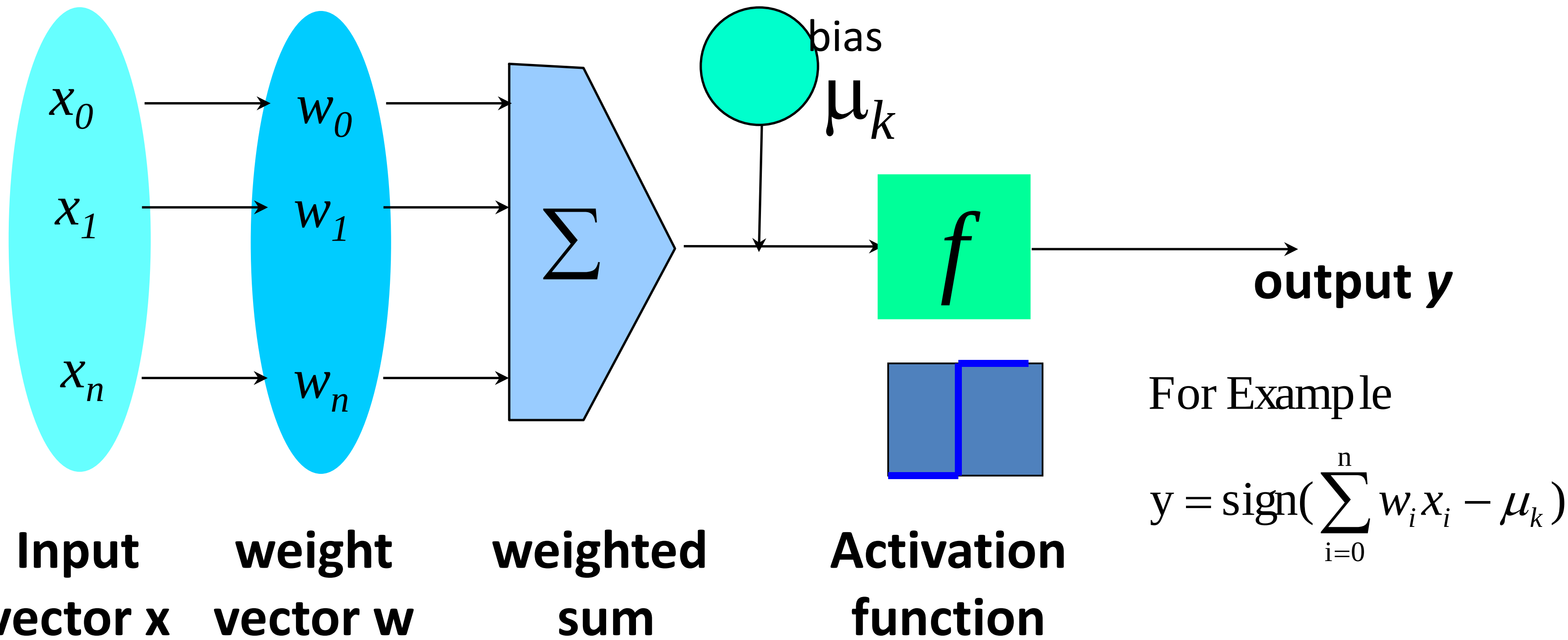
Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights to small random numbers, associated with biases
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Back propagation Network

- **Weights enable an artificial neural network to adjust the strength of connections between neurons, bias can be used to make adjustments within neurons**. Bias can be positive or negative, increasing or decreasing a neuron's output.

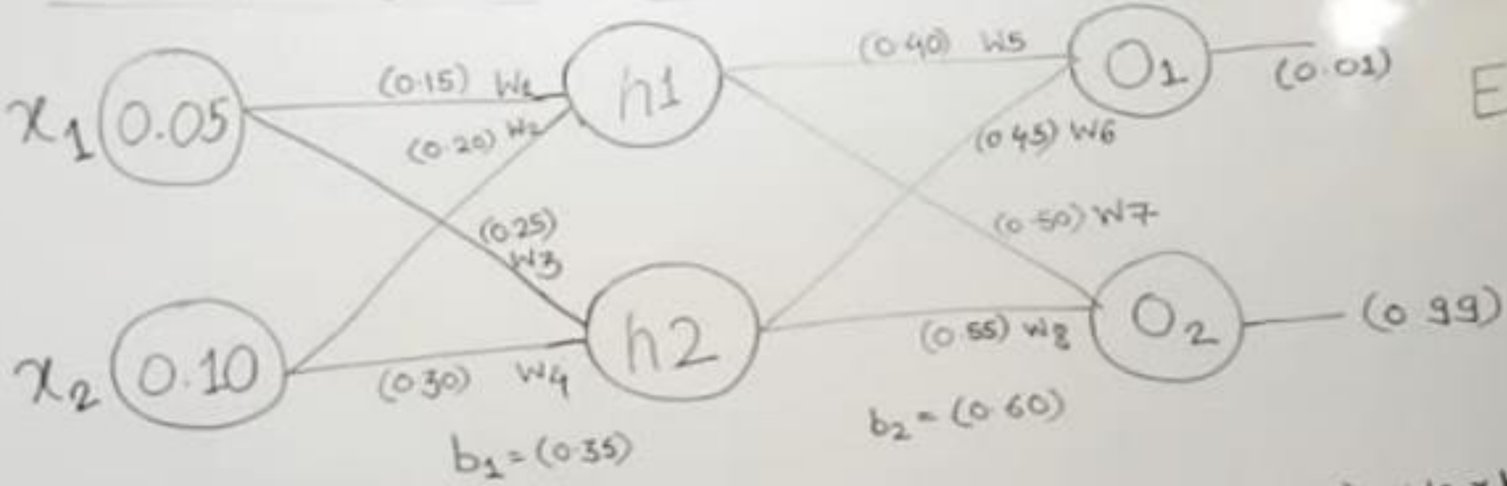
Neuron: A Hidden/Output Layer Unit



- An *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping
- The inputs to unit are outputs from the previous layer. They are multiplied by their corresponding weights to form a weighted sum, which is added to the bias associated with unit. Then a nonlinear activation function is applied to it.

Back propagation Network

Backpropagation / Backward Propagation of error



$E_{total} = E_{O_1} + E_{O_2}$
$= 0.2983$

$h_1(in) = W_1 \times x_1 + W_2 \times x_2 + b_1$
$= (0.15 \times 0.05 + 0.2 \times 0.1 + 0.35)$
$= 0.377$

$h_1(out) = \frac{1}{1+e^{-h_1(in)}} = 0.5932$

$h_2(out) = 0.5968$

$O_1(in) = W_5 \times h_1(out) + W_6 \times h_2(out) + b_2$
$= (0.4 \times 0.593 + 0.45 \times 0.596 + 0.6)$
$= 1.105$

$O_1(out) = \frac{1}{1+e^{-O_1(in)}} = 0.7513$

$O_2(out) = 0.7729$

$E_{Total} = \sum \frac{1}{2}(target - o/p)^2$

$E_{O_1} = 0.274$ $E_{O_2} = 0.0235$

Back propagation Network

$E_{total} = 0.298371109$

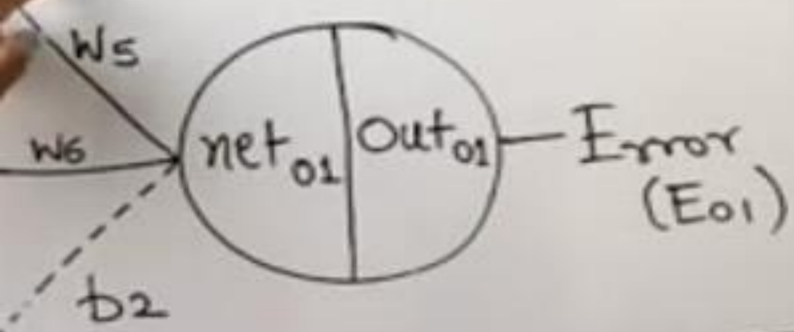
$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

- $$\frac{\partial E_{total}}{\partial out_{o1}} = Out_{o1} - Target_{o1}$$
  $$= 0.751365 - 0.01$$
  $$= 0.741365$$

- $$\frac{\partial out_{o1}}{\partial net_{o1}} = Out_{o1}(1 - Out_{o1})$$
  $$= 0.751365(1 - 0.751365)$$
  $$= 0.186815602$$

- $$\frac{\partial net_{o1}}{\partial w_5} = Out_{h1} = 0.593269992$$

- $$\frac{\partial E_{total}}{\partial w_5} = 0.08216704$$



$$W_5^* = W_5 - \alpha \times \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.6 * 0.08216704$$
$$= 0.350699776$$

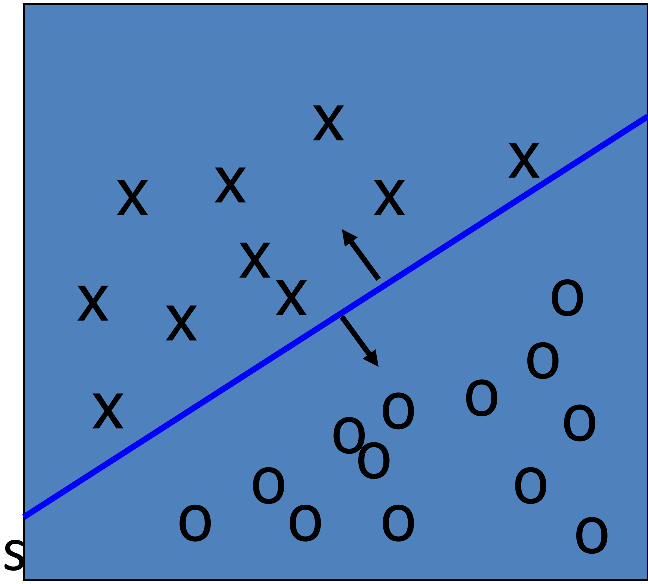
Efficiency and Interpretability

- **Efficiency** of backpropagation: Each epoch (one iteration through the training set) takes O(|D| * *w*), with |D| tuples and *w* weights, but # of epochs can be exponential to n, the number of inputs, in worst case
- For easier comprehension: **Rule extraction** by network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- **Sensitivity analysis**: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

Support Vector Machines

Classification: A Mathematical Mapping

- Classification: predicts categorical class labels
  - E.g., Personal homepage classification
    - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1
    - x_1 : # of word "homepage"
    - x_2 : # of word "welcome"
- Mathematically, $x \in X = \Re^n$, $y \in$
  - We want to derive a function f: X → Y
- Linear Classification
  - Binary Classification problem
  - Data above the red line belongs to class
  - Data below red line belongs to class 'o'
  - Examples: SVM, Perceptron, Probabilistic Classifiers



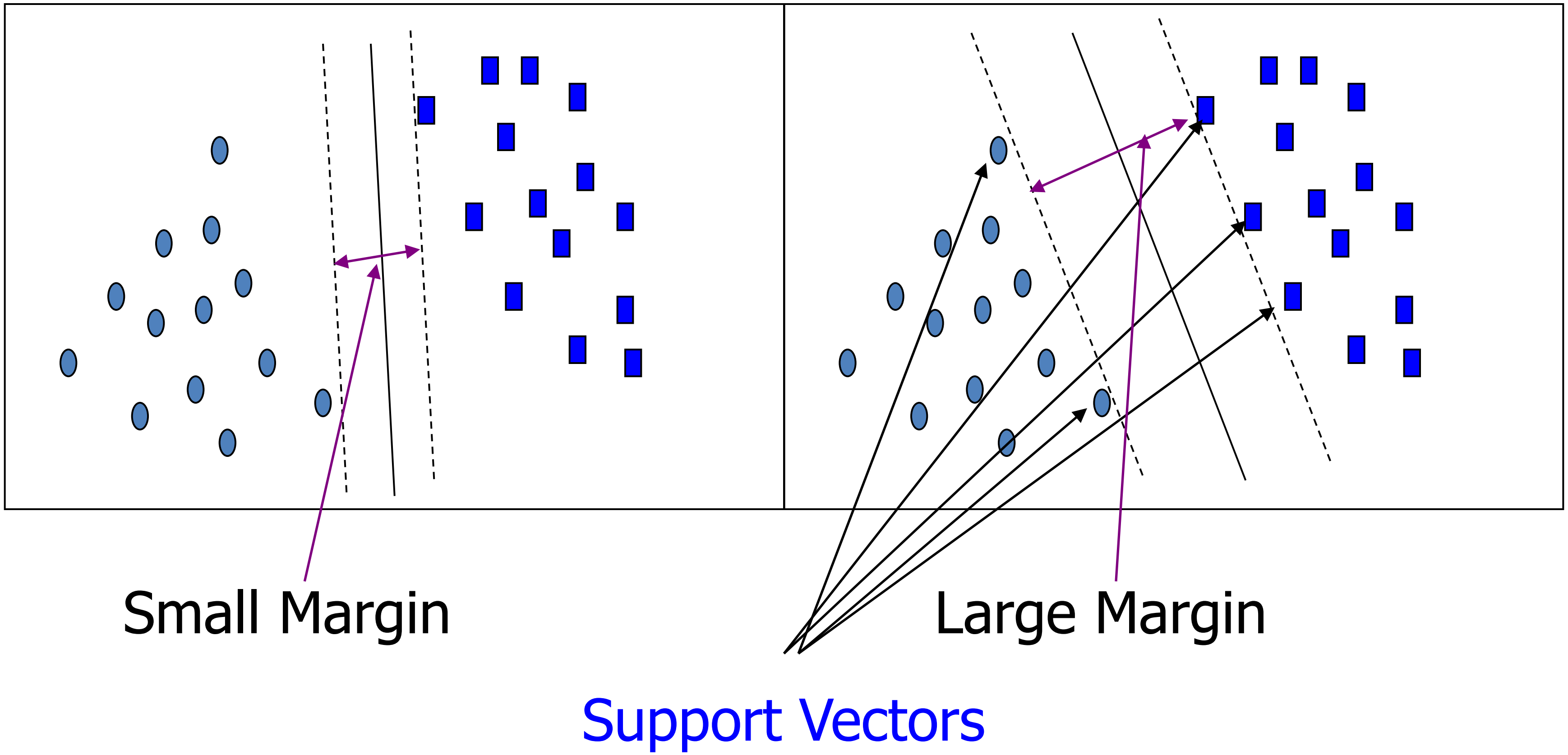
Discriminative Classifiers

- Advantages
  - Prediction accuracy is generally high
    - As compared to Bayesian methods – in general
  - Robust, works when training examples contain errors
  - Fast evaluation of the learned target function
    - Bayesian networks are normally slow
- Criticism
  - Long training time
  - Difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - Not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

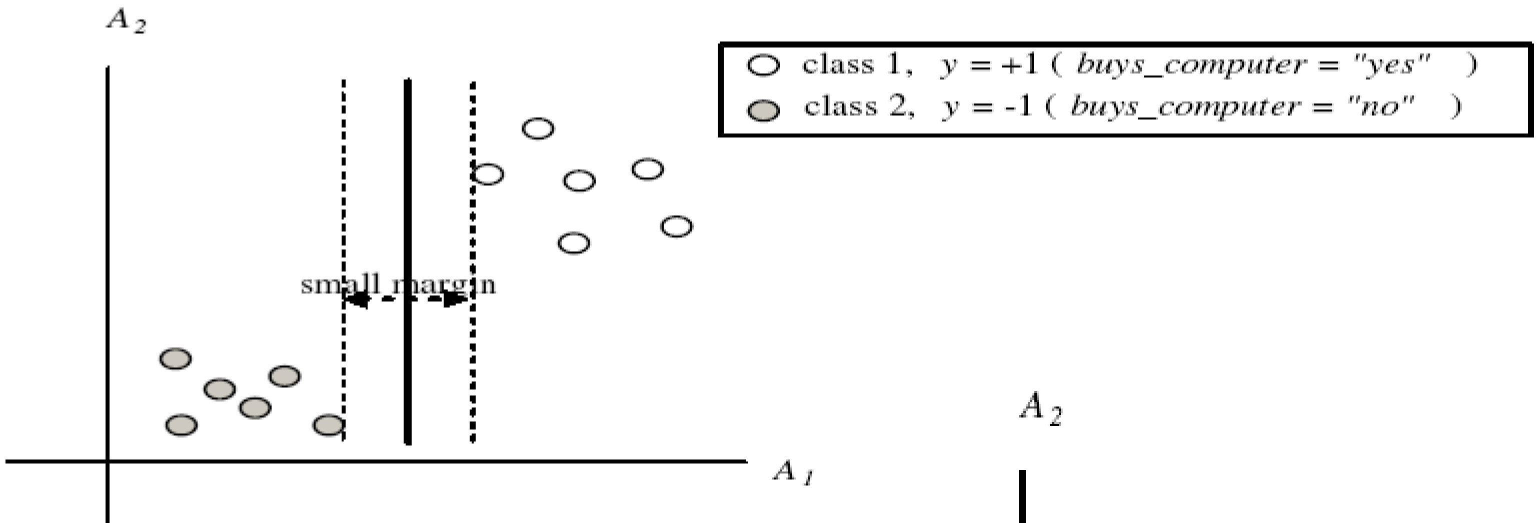
SVM—Support Vector Machines

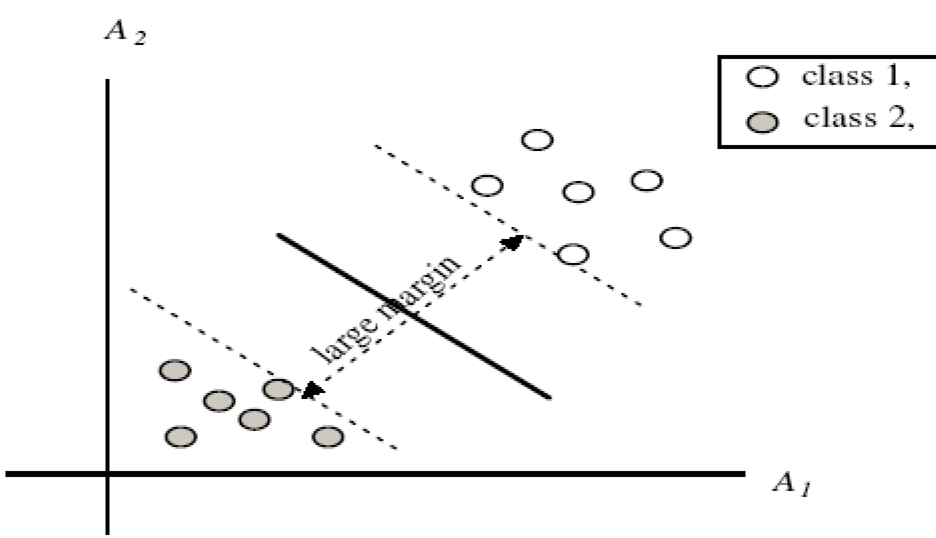
- A relatively new classification method for both <u>linear and nonlinear</u> data
- It uses a <u>nonlinear mapping</u> to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** ("essential" training tuples) and **margins** (defined by the support vectors)

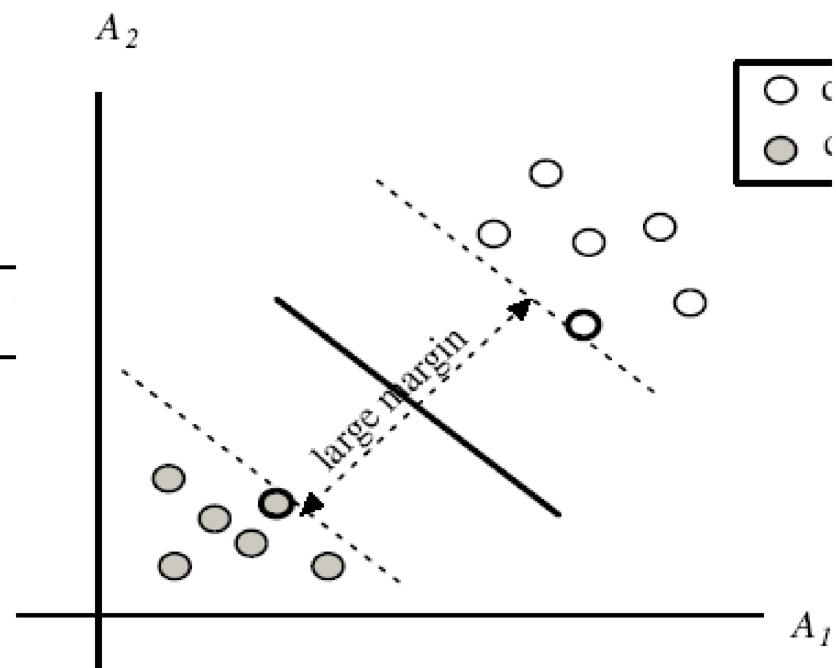
SVM—General Philosophy



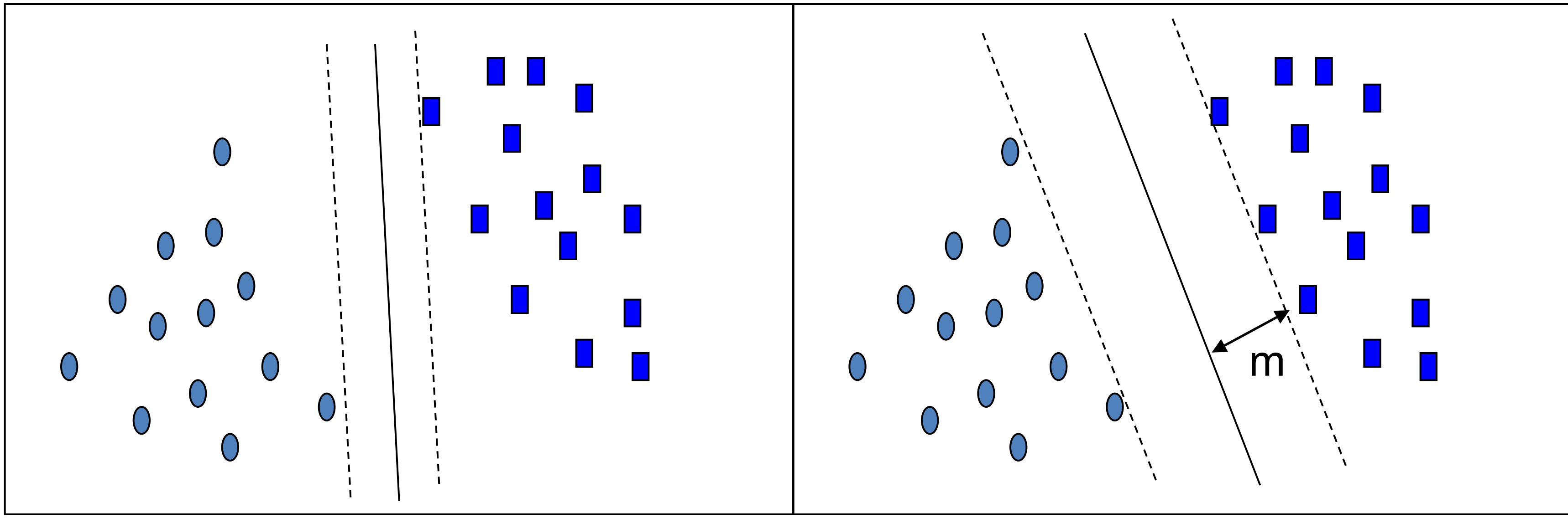
SVM—Margins and Support Vectors







SVM—When Data Is Linearly Separable



Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (<u>hyperplanes</u>) separating the two classes but we want to <u>find the best one</u> (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

SVM—Linearly Separable

- A separating hyperplane can be written as

  $\mathbf{W} \bullet \mathbf{X} + b = 0$

  where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

  $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:

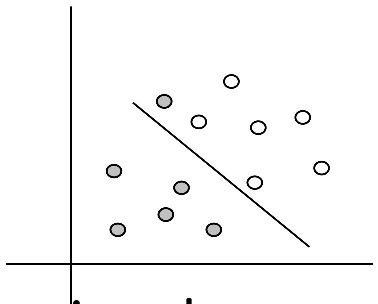
  $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

Why Is SVM Effective on High Dimensional Data?

- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

SVM—Linearly Inseparable



- Transform the original input data into a higher dimensional space

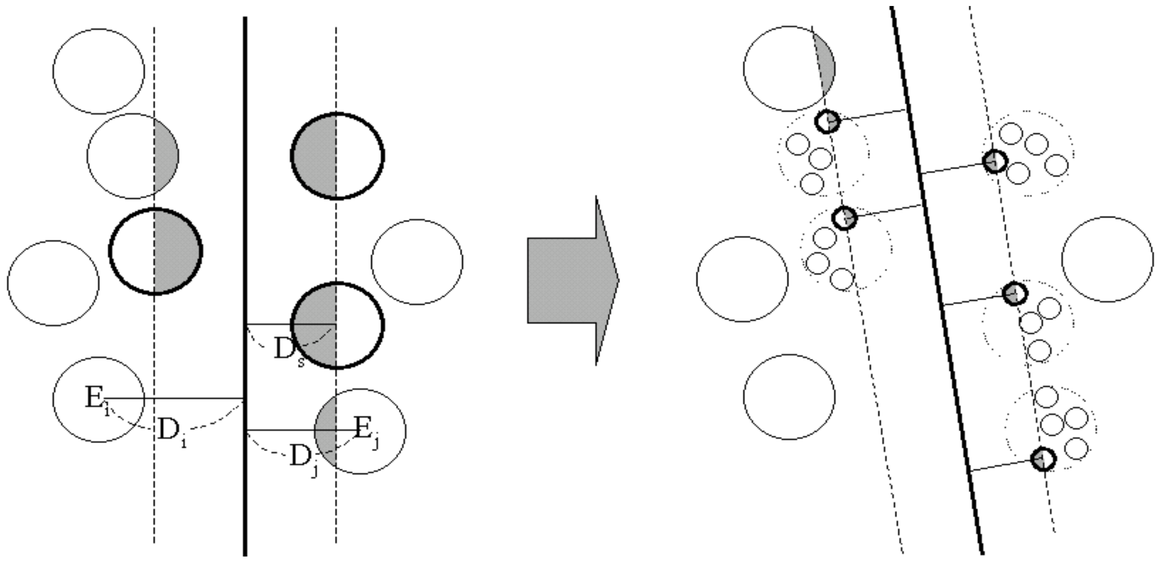
Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3$, $\phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and b and then substitute back so that we see that the linear decision hyperplane in the new ($\mathbf{Z}$) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned}$$ ∎

- Search for a linear separating hyperplane in the new space

Selective Declustering: Ensure High Accuracy

- CF tree is a suitable base structure for selective declustering
- De-cluster only the cluster E_i such that
  - $D_i - R_i < D_s$, where D_i is the distance from the boundary to the center point of E_i and R_i is the radius of E_i
  - Decluster only the cluster whose subclusters have possibilities to be the support cluster of the boundary
    - "Support cluster": The cluster whose centroid is a support vector



SVM vs. Neural Network

- **SVM**
  - Deterministic algorithm
  - Nice generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

- **Neural Network**
  - Nondeterministic algorithm
  - Generalizes well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (nontrivial)

Lazy Learners (or Learning from Your Neighbors)

Lazy vs. Eager Learning

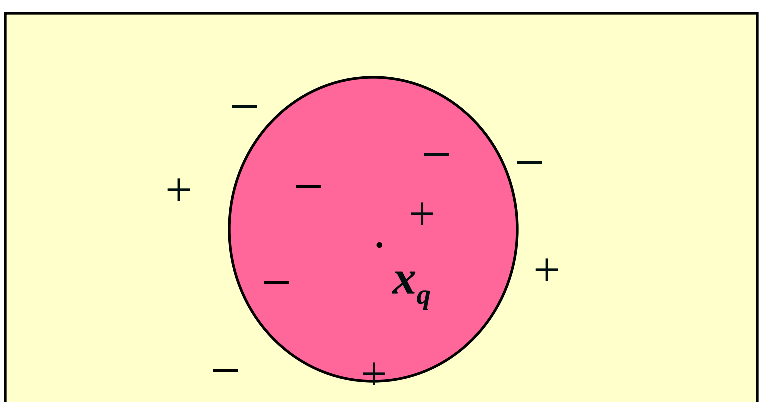
- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

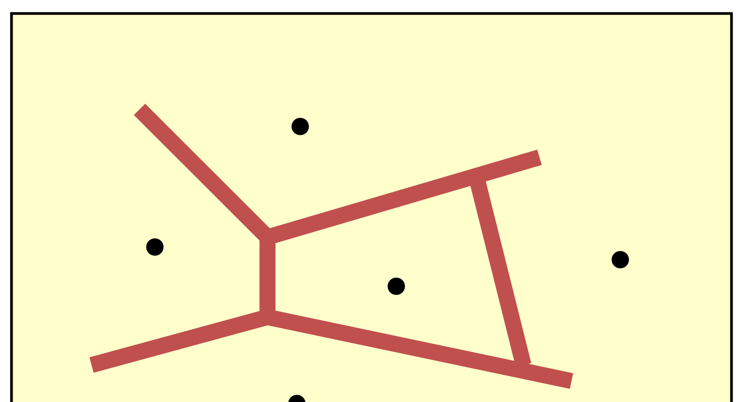
Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
  - <u>*k*-nearest neighbor approach</u>
    - Instances represented as points in a Euclidean space.
  - <u>Locally weighted regression</u>
    - Constructs local approximation
  - <u>Case-based reasoning</u>
    - Uses symbolic representations and knowledge-based inference

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist($\mathbf{X_1}$, $\mathbf{X_2}$)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples





Discussion on the *k*-NN Algorithm

- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

Case-Based Reasoning (CBR)

- **CBR**: Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

Additional Topics Regarding Classification

Multiclass Classification

- Classification involving more than two classes (i.e., > 2 Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
  - Given m classes, train m classifiers: one for each class
  - Classifier j: treat tuples in class j as *positive* & all others as *negative*
  - To classify a tuple **X**, the set of classifiers vote as an ensemble
- Method 2. **All-vs.-all** (AVA): Learn a classifier for each pair of classes
  - Given m classes, construct m(m-1)/2 binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple **X**, each classifier votes. X is assigned to the class with maximal vote
- Comparison
  - All-vs.-all tends to be superior to one-vs.-all
  - Problem: Binary classifier is sensitive to errors, and errors affect vote count

Error-Correcting Codes for Multiclass Classification

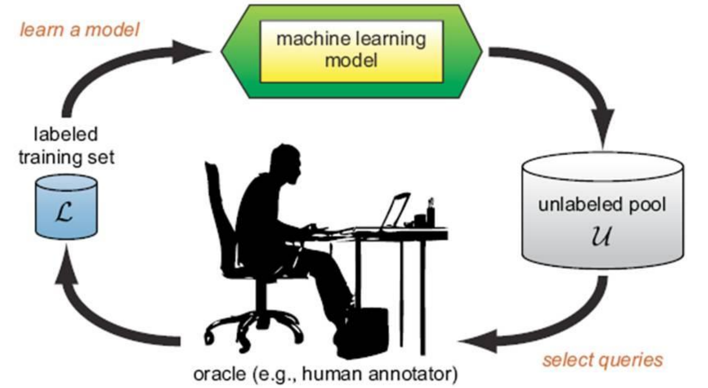
Class	Error-Corr. Codeword						
C_1	1	1	1	1	1	1	1
C_2	0	0	0	0	1	1	1
C_3	0	0	1	1	0	0	1
C_4	0	1	0	1	0	1	0

- Originally designed to correct errors during data transmission for communication tasks by exploring data redundancy
- Example
  - A 7-bit codeword associated with classes 1-4
    - Given a unknown tuple **X**, the 7-trained classifiers output: 0001010
    - Hamming distance: # of different bits between two codewords
    - H(**X**, C_1) = 5, by checking # of bits between [1111111] & [0001010]
    - H(**X**, C_2) = 3, H(**X**, C_3) = 3, H(**X**, C_4) = 1, thus C_4 as the label for **X**
- Error-correcting codes can correct up to (h-1)/h 1-bit error, where h is the minimum Hamming distance between any two codewords
- If we use 1-bit per class, it is equiv. to one-vs.-all approach, the code are insufficient to self-correct
- When selecting error-correcting codes, there should be good row-wise and col.-wise separation between the codewords

Semi-Supervised Classification

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training:
  - Build a classifier using the labeled data
  - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - Repeat the above process
  - Adv: easy to understand; disadv: may reinforce errors
- Co-training: Use two or more classifiers to teach each other
  - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say f_1
  - Then f_1 and f_2 are used to predict the class label for unlabeled data X
  - Teach each other: The tuple having the most confident prediction from f_1 is added to the set of labeled data for f_2, & vice versa
- Other methods, e.g., joint probability distribution of features and labels

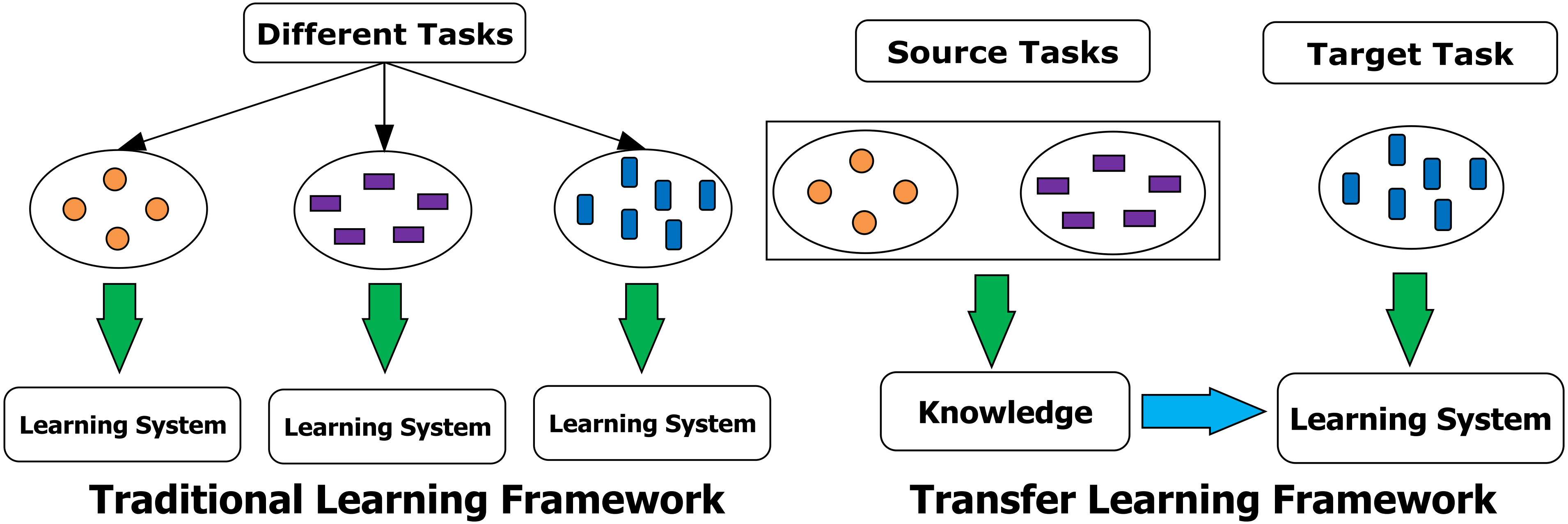
Active Learning



- Class labels are expensive to obtain
- Active learner: query human (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: Achieve high accuracy using as few labeled data as possible
- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- Research issue: How to choose the data tuples to be queried?
  - Uncertainty sampling: choose the least certain ones
  - Reduce *version space*, the subset of hypotheses consistent w. the training data
  - Reduce expected entropy over U: Find the greatest reduction in the total number of incorrect predictions

Transfer Learning: Conceptual Framework

- Transfer learning: Extract knowledge from one or more source tasks and apply the knowledge to a target task
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks



Transfer Learning: Methods and Applications

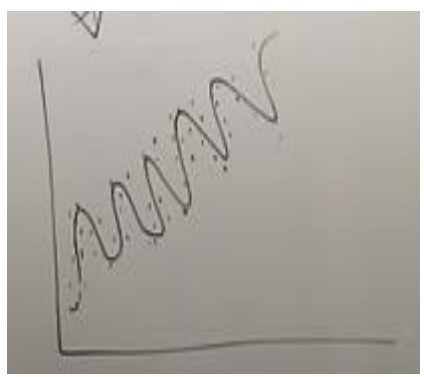
- Applications: Especially useful when data is outdated or distribution changes, e.g., Web document classification, e-mail spam filtering
- *Instance-based transfer learning*: Reweight some of the data from source tasks and use it to learn the target task
- TrAdaBoost (Transfer AdaBoost)
  - Assume source and target data each described by the same set of attributes (features) & class labels, but rather diff. distributions
  - Require only labeling a small amount of target data
  - Use source data in training: When a source tuple is misclassified, reduce the weight of such tupels so that they will have less effect on the subsequent classifier
- Research issues
  - Negative transfer: When it performs worse than no transfer at all
  - Heterogeneous transfer learning: Transfer knowledge from different feature space or multiple source domains
  - Large-scale transfer learning

Summary

- Effective and advanced classification methods
  - Bayesian belief network (probabilistic networks)
  - Backpropagation (Neural networks)
  - Support Vector Machine (SVM)
  - Pattern-based classification
  - Other classification methods: lazy learners (KNN, case-based reasoning), genetic algorithms, rough set and fuzzy set approaches
- Additional Topics on Classification
  - Multiclass classification
  - Semi-supervised classification
  - Active learning
  - Transfer learning

Regression Analysis

- Dependent and Independent variables
- Outliers
- Multi collinearty – Not to have this property as we can't decide which factor affect more / How these Independent variables are correlated with each other/ Independent variables when they are sharing non linear relationship with each other
- Under fitting / Over fitting



Regression Analysis

Linear Regression

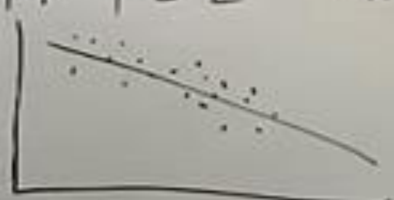
Dependent variable are continuous in nature

Linear Relationship

1 → I & 1 → D
↓
Simple Linear. R

1 – D & >1 → I
↓
Multiple Linear. R

$$y = \beta_1 + \beta_2 x_2 + \cdots \beta_k x_k + \varepsilon$$



Logistic Regression

→ Dependent variable is binary

→ 1 (True, success), 0 (False, failure)

→ Goal is to find the best fitting model for I & D variable relationship

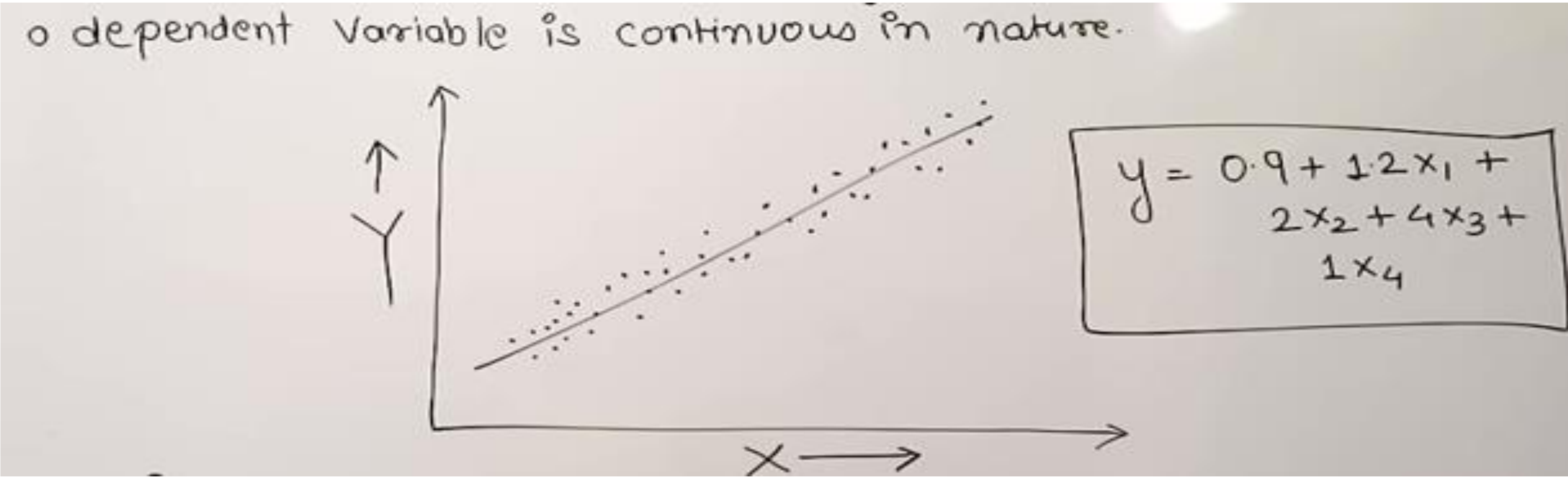
→ Independent variables can be continuous or binary.

→ Also called logit. R

→ Used in machine learning

→ Deals with probability to measure the relation betn dependent & independent variable.

Linear Regression



Simple linear equation

$$y = \alpha_0 + \alpha_1 x_1$$

$$y = c + mx$$

o **Multiple linear equation**

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 - \cdots + \alpha_m x_m$$

α_i = Reg. Coeff.
x_i = Independent var
y = Dependent var.

Linear Regression

Q:

X	Y	XY	x^2
1	3	3	1
2	4	8	4
3	5	15	9
4	7	28	16
10	19	54	30

$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$b = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$\boxed{Y = bx + a}$$

$$a = \frac{(19)(30) - (10)(54)}{(4)(30) - (100)}$$

$$= \frac{570 - 540}{120 - 100}$$

$$= 30/20 = \frac{3}{2} = 1.5$$

$$b = \frac{(4)(54) - (10)(19)}{(4)(30) - (100)}$$

$$= \frac{216 - 190}{120 - 100}$$

$$= 26/20 = 13/10 = 1.3$$

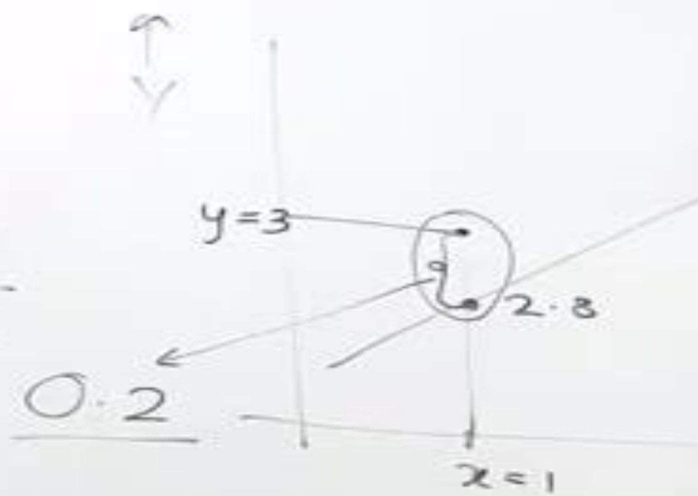
$$\boxed{Y = 1.3X + 1.5}$$

Linear Regression

Q:

X	Y	XY	x^2	P	Error
1	3	3	1	2.8	0.2
2	4	8	4	4.1	0.1
3	5	15	9	5.4	0.4
4	7	28	16	6.7	0.3
10	19	54	30		

$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)}{n(\Sigma x^2) - (\Sigma x)^2}$$



$$b = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$\boxed{Y = 1.3X + 1.5}$$

Logistic Regression