# Decision Trees

Emre Ugur, BM 33

emre.ugur@boun.edu.tr

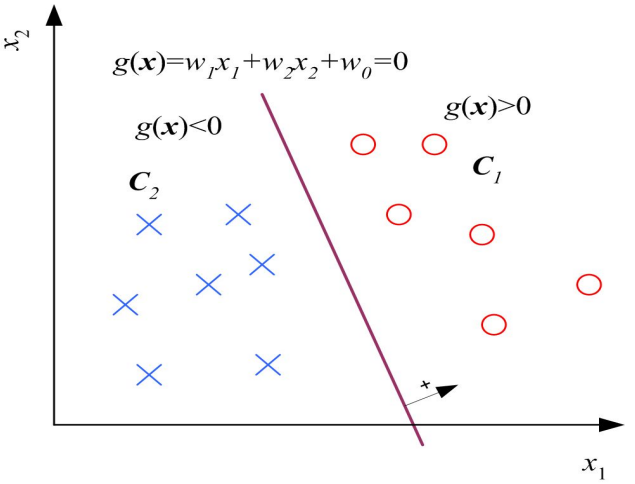
http://www.cmpe.boun.edu.tr/~emre/courses/cmpe462

cmpe462@listeci.cmpe.boun.edu.tr

# Previously on CMPE462

- Linear discriminants

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$



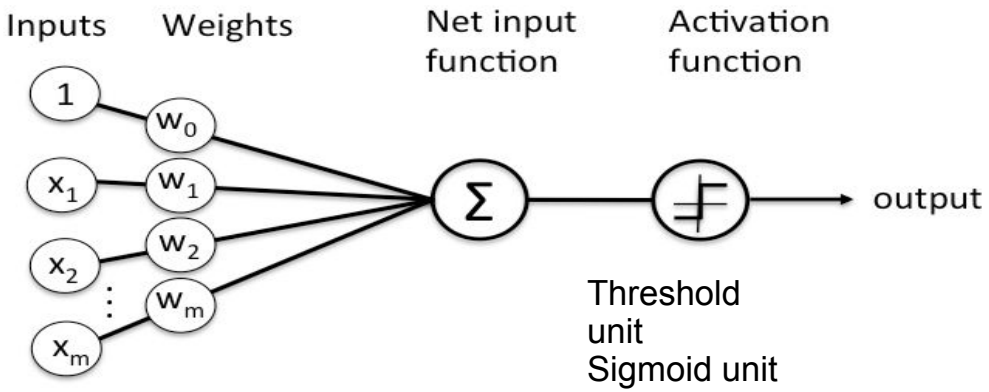

- Logistic discrimination

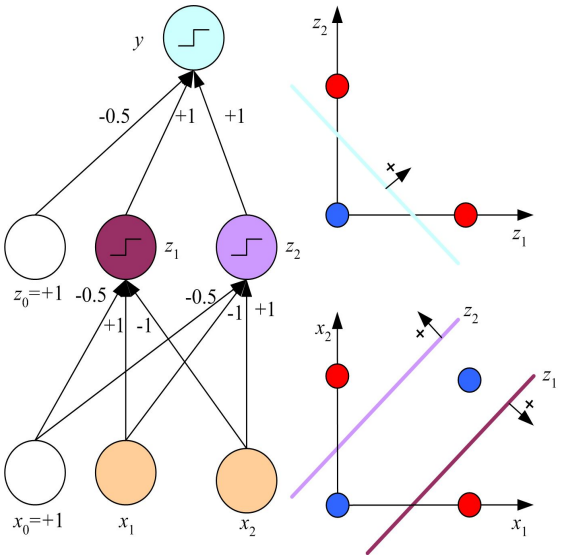
$$y = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x} + w_0) \text{ and choose} C_1 \text{ if } y > 0.5$$

- Gradient descent

$$\Delta w_j = \eta \sum_t (r^t - y^t) x_j^t$$

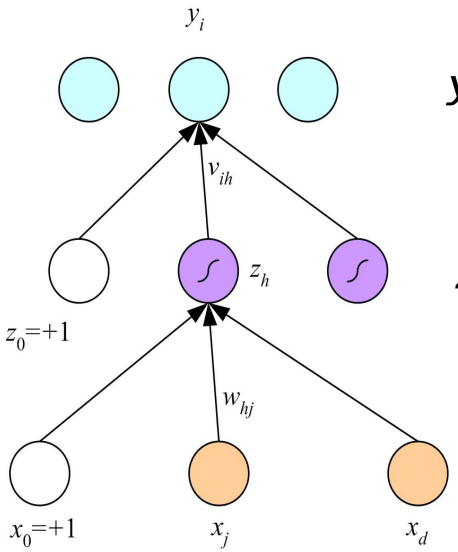
- Perceptrons



# Previously on CMPE462

- Multilayer perceptron




$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_o$$

$$z_h = \text{sigmoid}(w_h^T x)$$

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$-\eta \sum_t \frac{\partial E}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}}$$

$$-\eta \sum_t -(r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

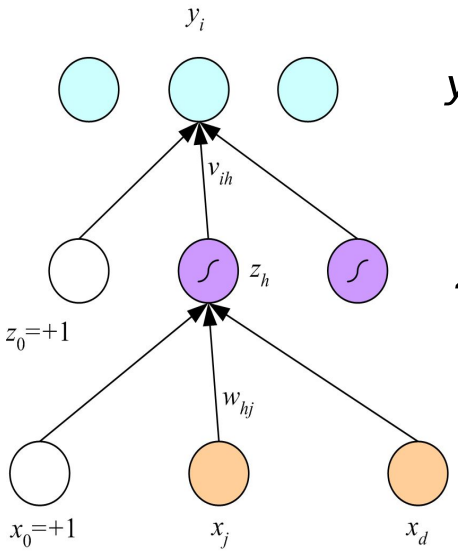
$$\eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

# Quiz

- Build a perceptron that calculates NOT of its one input.

# Previously on CMPE462: A paranthesis

- Multilayer perceptron




$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0 \qquad y^t = \text{sigmoid}\left( \sum_{h=1}^{H} v_h z_h^t + v_0 \right)$$

$$z_h = \text{sigmoid}(w_h^T x)$$

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$-\eta \sum_t \frac{\partial E}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}}$$

$$-\eta \sum_t -(r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

$$\eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

- Logistic discrimination

$$y = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x} + w_0) \text{ and choose } C_1 \text{ if } y > 0.5$$

$$\Delta w_j = \eta \sum_t (r^t - y^t) x_j^t$$

# Logistic Discrimination
# Logistic Regression: Target is categorical

- Do not model class-conditional densities, but their ratio
- Two classes: Assume log likelihood ratio is linear

$$\log\frac{p(\boldsymbol{x}\,|\,C_1)}{p(\boldsymbol{x}\,|\,C_2)} = \boldsymbol{w}^T\boldsymbol{x} + w_0^o$$

$$\begin{aligned}\text{logit}(P(C_1\,|\,\boldsymbol{x})) &= \log\frac{P(C_1\,|\,\boldsymbol{x})}{1-P(C_1\,|\,\boldsymbol{x})} = \log\frac{p(\boldsymbol{x}\,|\,C_1)}{p(\boldsymbol{x}\,|\,C_2)} + \log\frac{P(C_1)}{P(C_2)}\\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0\end{aligned}$$

$$\text{where} w_0 = w_0^o + \log\frac{P(C_1)}{P(C_2)}$$

$$y = \hat{P}(C_1\,|\,\boldsymbol{x}) = \frac{1}{1+\exp\left[-\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right)\right]}$$

# Logistic Discrimination
# Logistic Regression: Target is categorical

- Do not model class-conditional densities, but their ratio
- Two classes: Assume log likelihood ratio is linear

# Training: Two Classes

$$\mathrm{X} = \left\{\boldsymbol{x}^t, r^t\right\}_t \quad r^t \mid \boldsymbol{x}^t \sim \text{Bernoulli}\left(y^t\right)$$

$$y = P\left(C_1 \mid \boldsymbol{x}\right) = \frac{1}{1+\exp\left[-\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right)\right]}$$

$$l\left(\boldsymbol{w}, w_0 \mid \mathrm{X}\right) = \prod_t \left(y^t\right)^{\left(r^t\right)}\left(1-y^t\right)^{\left(1-r^t\right)}$$

$$E = -\log l$$

$$E\left(\boldsymbol{w}, w_0 \mid \mathrm{X}\right) = -\sum_t r^t \log y^t + \left(1-r^t\right)\log\left(1-y^t\right)$$

# Training: Gradient-Descent

$$E(\boldsymbol{w}, w_0 \mid X) = -\sum_t r^t \log y^t + (1 - r^t) \log (1 - y^t)$$

$$\text{If } y = \text{sigmoid}(a) \quad \frac{dy}{da} = y(1 - y)$$

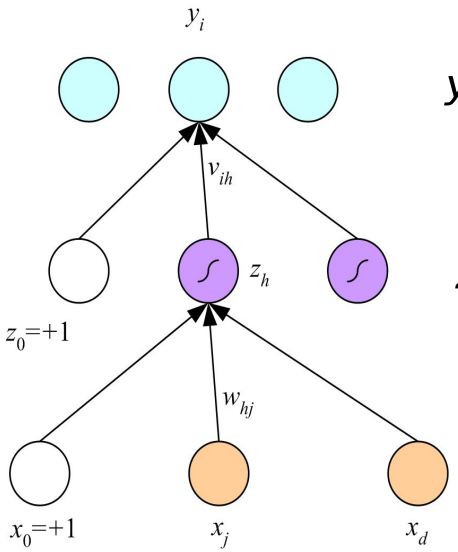
$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left( \frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t (1 - y^t) x_j^t$$

$$= \eta \sum_t (r^t - y^t) x_j^t, j = 1, \ldots d$$

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t)$$

# Previously on CMPE462: A paranthesis

- Multilayer perceptron




$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0 \qquad y^t = \text{sigmoid}\left( \sum_{h=1}^{H} v_h z_h^t + v_0 \right)$$

$$z_h = \text{sigmoid}(w_h^T x)$$

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$-\eta \sum_t \frac{\partial E}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}}$$

$$-\eta \sum_t -(r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

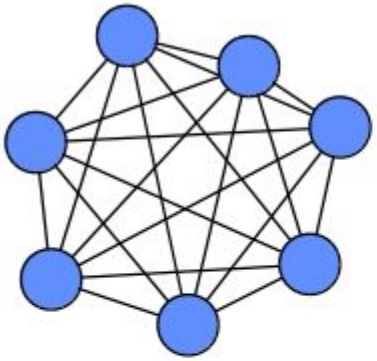
$$\eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

- Logistic discrimination

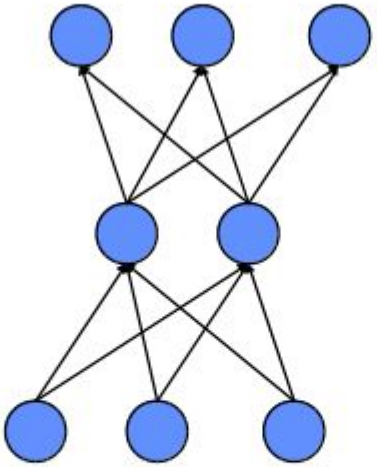
$$y = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x} + w_0) \text{ and choose } C_1 \text{ if } y > 0.5$$

$$\Delta w_j = \eta \sum_t (r^t - y^t) x_j^t$$

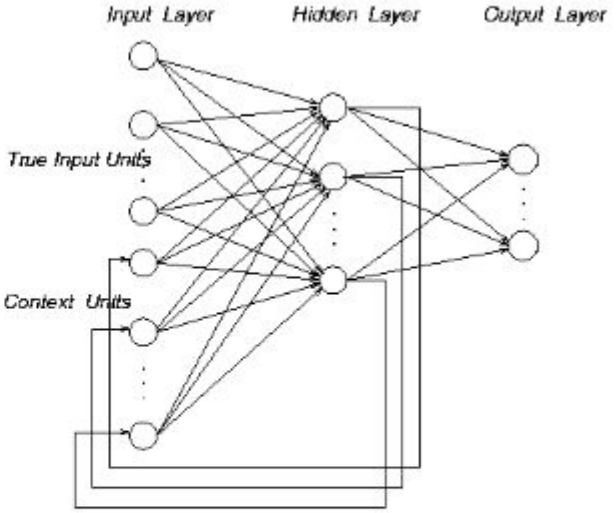
# Topologies of Neural Networks



*completely connected*



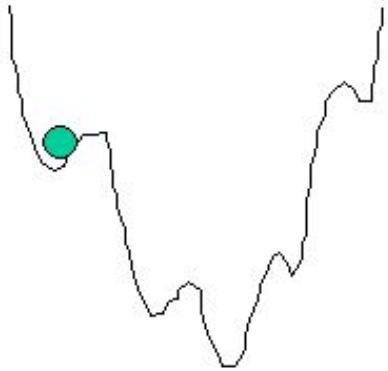
*feedforward (directed, a-cyclic)*

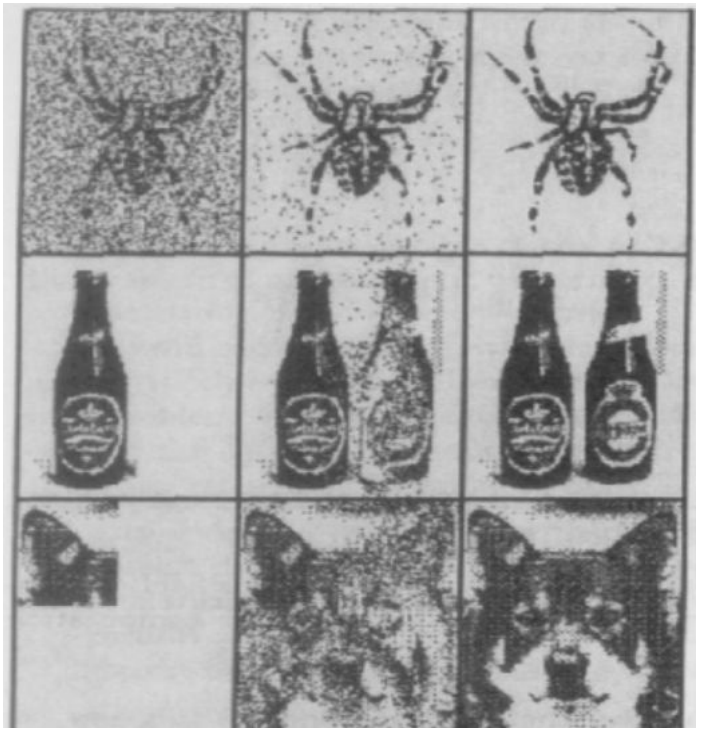


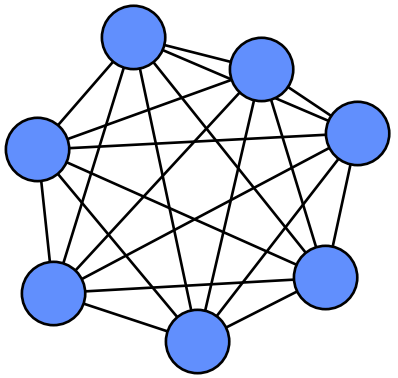

*recurrent (feedback connections)*

# Hopfield Networks

- Act as “autoassociative” memories to store patterns
- Network converges to local minima which store different patterns.

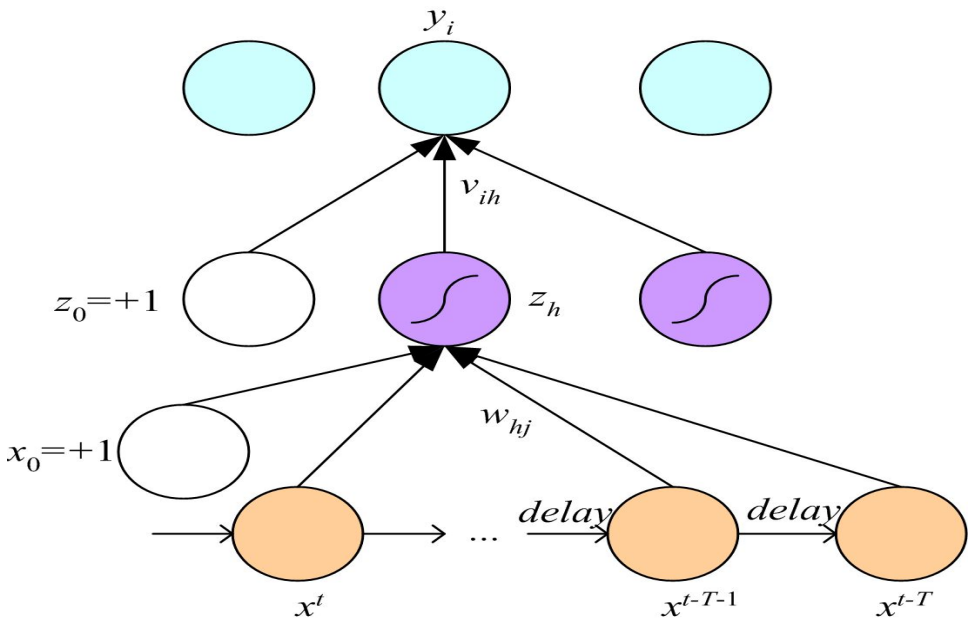


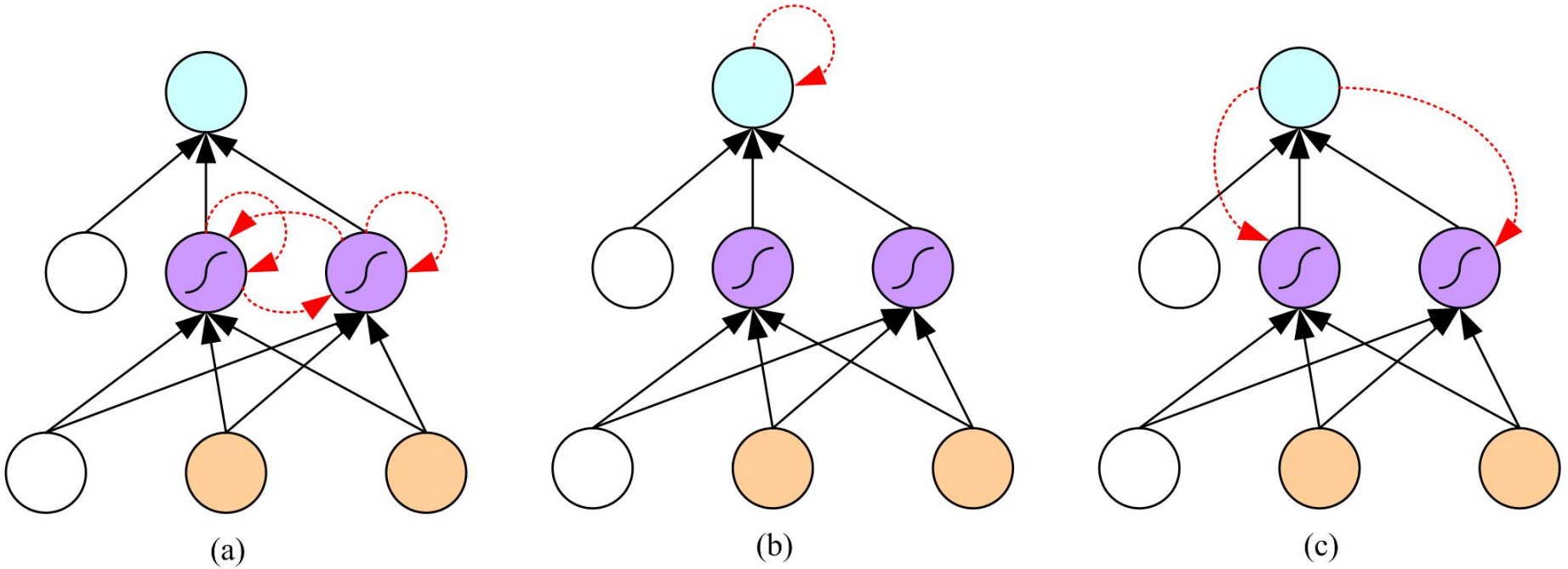




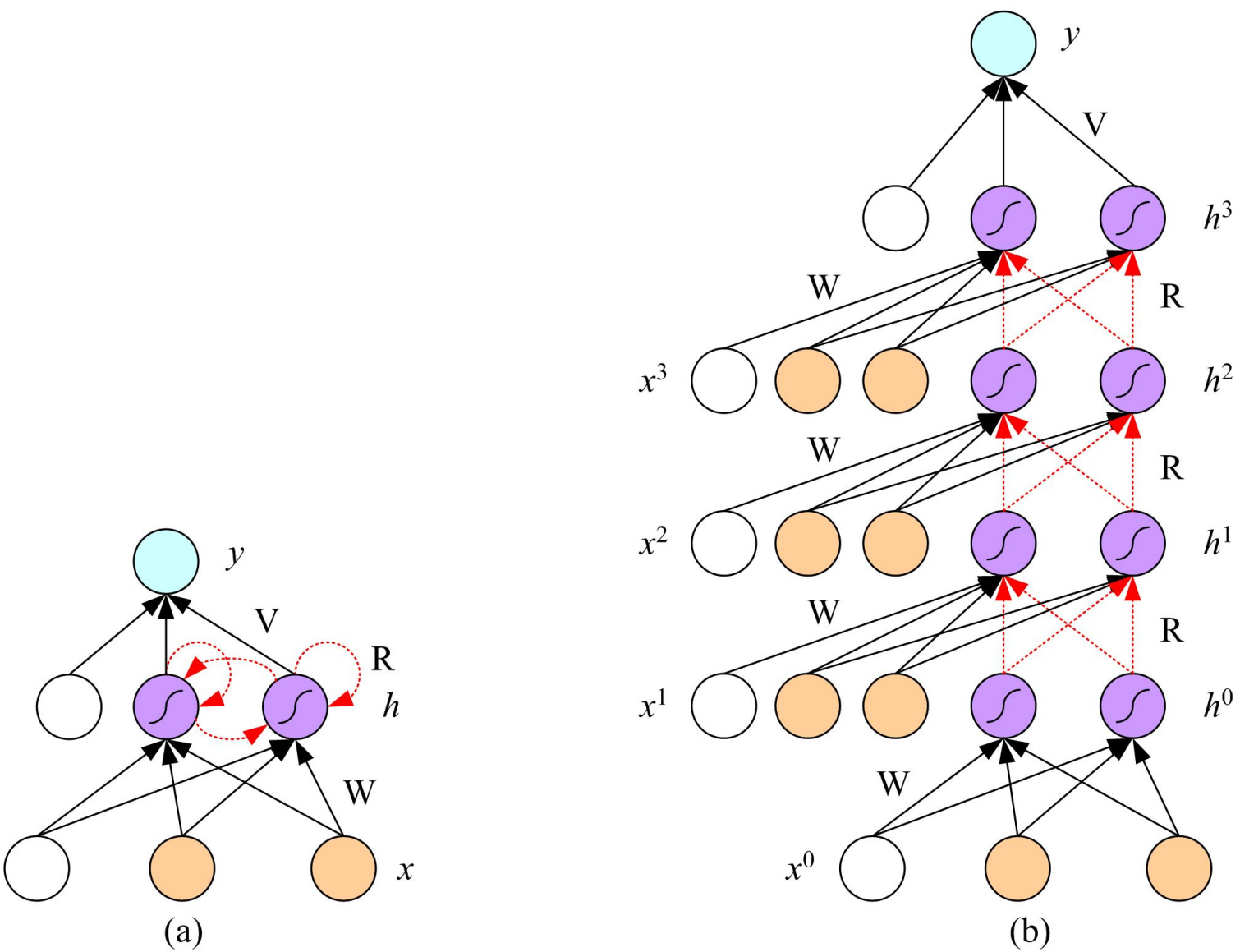
*completely connected*

# Time-delay Neural Networks



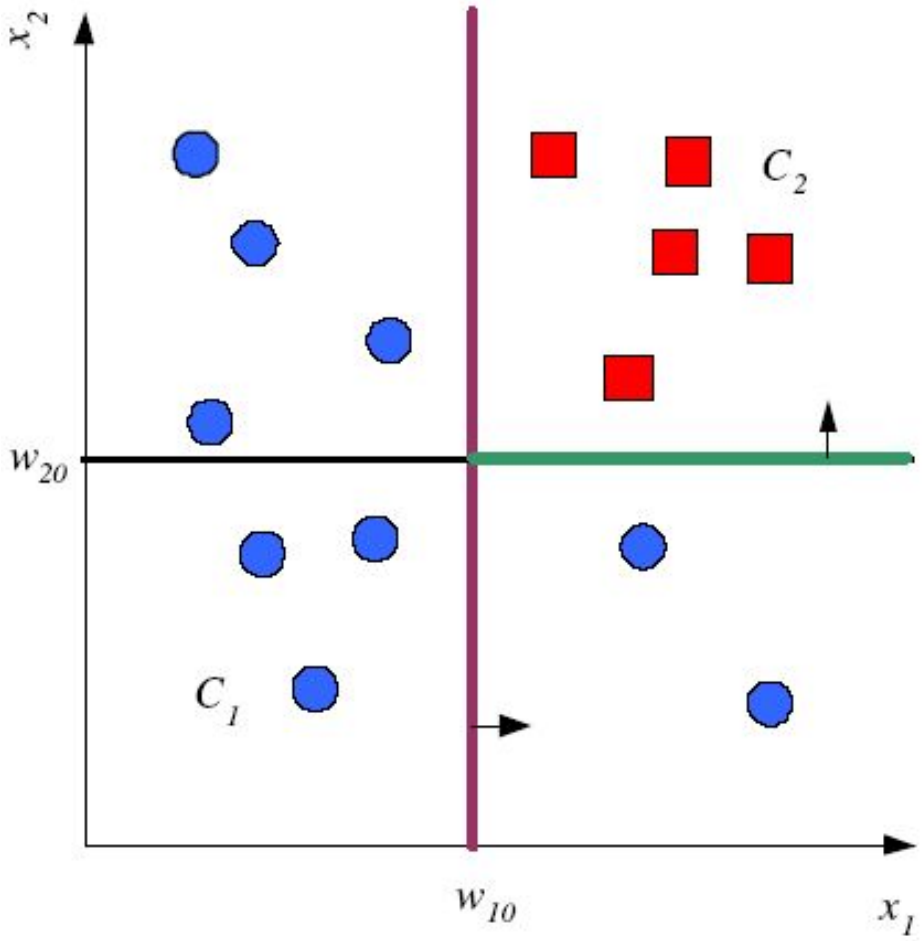



# Recurrent Networks

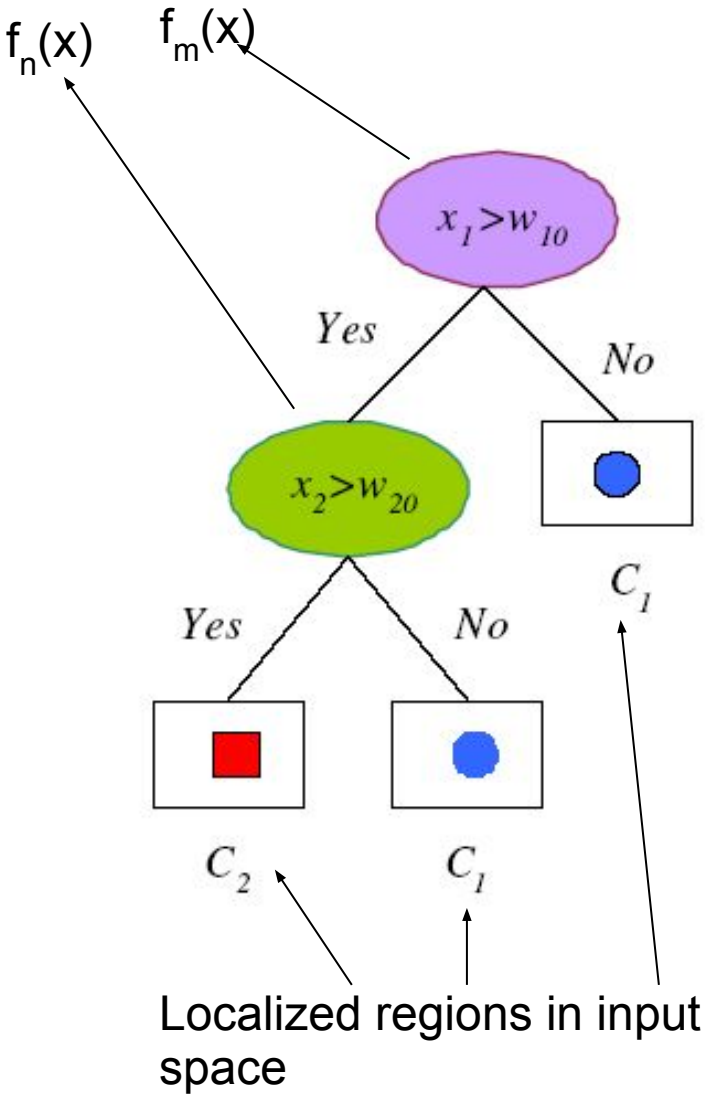


# Decision Trees

# Tree Uses Nodes, and Leaves



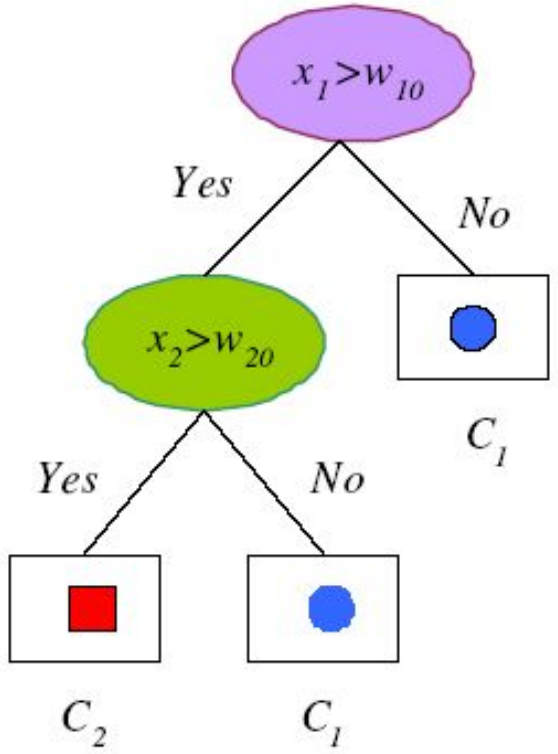

Internal decision nodes




Both for classication and regression

# Properties

- Structure not fixed
- Fast localization
- Interpretability
  - IF-THEN rules
- Many correct trees
  - Which one to choose?
  - Learning is greedy
    - find the best split recursively



# Training: Two Classes

$$\mathrm{X} = \left\{\boldsymbol{x}^t, r^t\right\}_t \quad r^t \mid \boldsymbol{x}^t \sim \text{Bernoulli}\left(y^t\right)$$

$$y = P\left(C_1 \mid \boldsymbol{x}\right) = \frac{1}{1+\exp\left[-\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right)\right]}$$

$$l\left(\boldsymbol{w}, w_0 \mid \mathrm{X}\right) = \prod_t \left(y^t\right)^{\left(r^t\right)}\left(1-y^t\right)^{\left(1-r^t\right)}$$

$$E = -\log l$$

$$E\left(\boldsymbol{w}, w_0 \mid \mathrm{X}\right) = -\sum_t r^t \log y^t + \left(1-r^t\right)\log\left(1-y^t\right)$$

# Likelihood, Bernoulli

- Assume each $x_i$ is a different variable from Bernoulli distribution with same p
  - Joint probability distribution

$$\prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i}$$

- Same variable, y times out of n
  - Binomial distribution

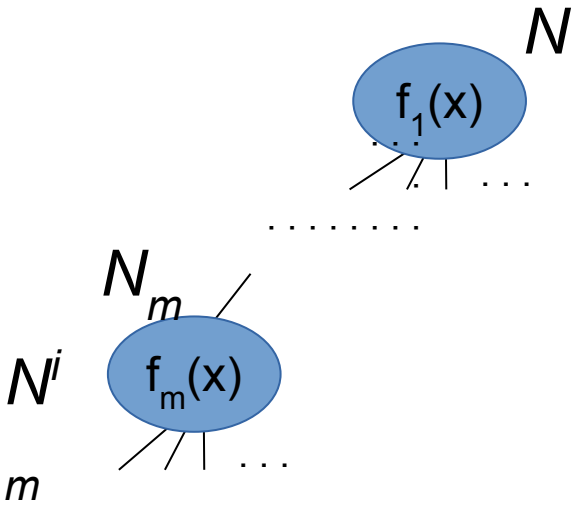
$$\binom{n}{y} p^{y}(1-p)^{n-y}$$

- For finding parameters, maximum likelihood
  - Coefficient drops
- For making inference, take ratio
  - Coefficient drops

# Classification Trees – Impurity (ID3, CART, C4.5)

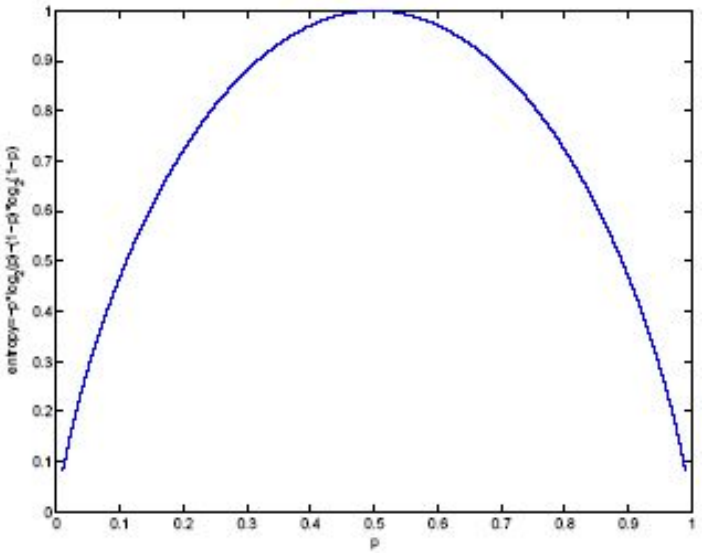
- For node $m$, $N_m$ instances reach $m$, $N^i_m$ belong to $C_i$

$$\hat{P}(C_i|\mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$




- Node $m$ is pure if all instances same class
- Node $m$ is pure if $p^i_m$ is 0 or 1
- Measure of impurity is entropy

$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$




- = uncertainty

# Simple Example

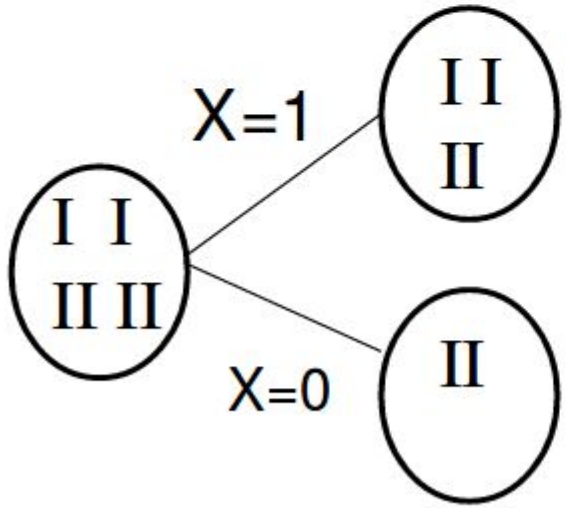
Training Set: 3 features and 2 classes

| X | Y | Z | C |
|---|---|---|---|
| 1 | 1 | 1 | I |
| 1 | 1 | 0 | I |
| 0 | 0 | 1 | II |
| 1 | 0 | 0 | II |

How would you distinguish class I from class II?

| X | Y | Z | C |
|---|---|---|---|
| 1 | 1 | 1 | I |
| 1 | 1 | 0 | I |
| 0 | 0 | 1 | II |
| 1 | 0 | 0 | II |

Split on attribute X




If X is the best attribute,
this node would be further split.

$E_{child1} = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= .5284 + .39$
$= .9184$

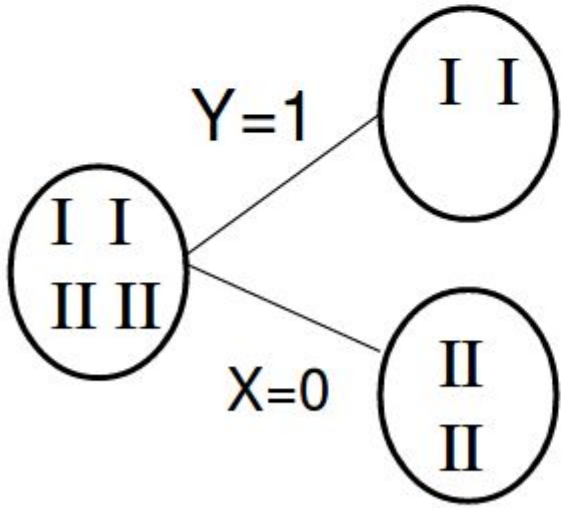
$E_{child2} = 0$

$E_{parent} = 1$

GAIN = 1 – ( 3/4)(.9184) – (1/4)(0) = .3112

| X | Y | Z | C |
|---|---|---|---|
| 1 | 1 | 1 | I |
| 1 | 1 | 0 | I |
| 0 | 0 | 1 | II |
| 1 | 0 | 0 | II |

Split on attribute Y




$E_{child1} = 0$

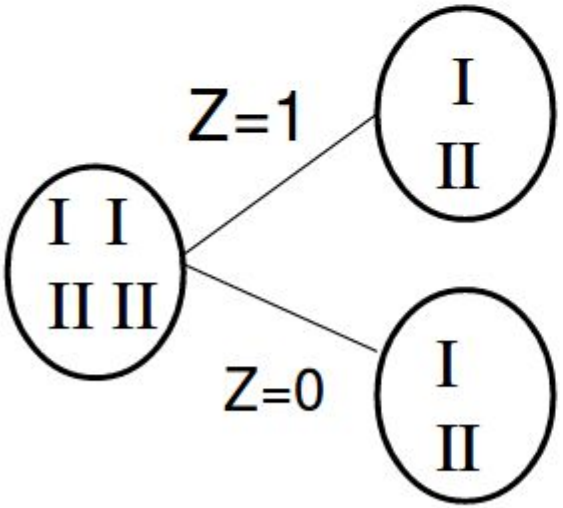
$E_{child2} = 0$

$E_{parent} = 1$

GAIN = 1 –(1/2) 0 – (1/2)0 = 1; BEST ONE

| X | Y | Z | C |
|---|---|---|---|
| 1 | 1 | 1 | I |
| 1 | 1 | 0 | I |
| 0 | 0 | 1 | II |
| 1 | 0 | 0 | II |

Split on attribute Z




$E_{child1} = 1$

$E_{child2} = 1$

$E_{parent} = 1$

GAIN = 1 – ( 1/2)(1) – (1/2)(1) = 0  ie. NO GAIN; WORST

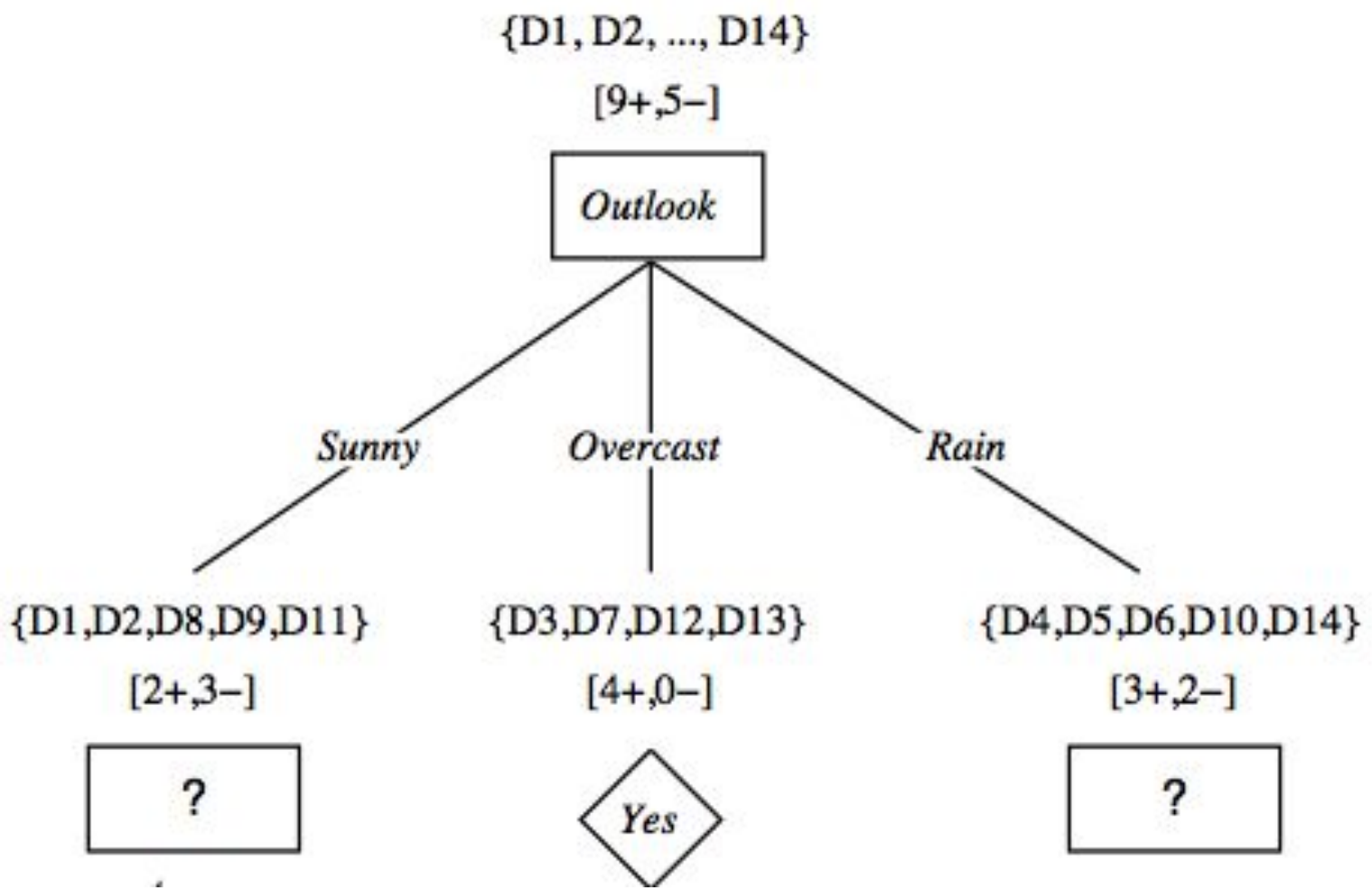
# Example

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

# First step: which attribute to test at the root?

- Which attribute should be tested at the root?
- $Gain(S, Outlook) = 0.246$
- $Gain(S, Humidity) = 0.151$
- $Gain(S, Wind) = 0.084$
- $Gain(S, Temperature) = 0.029$
- *Outlook* provides the best prediction for the target
- Lets grow the tree:
- add to the tree a successor for each possible value of *Outlook*
- partition the training samples according to the value of *Outlook*

# After first step



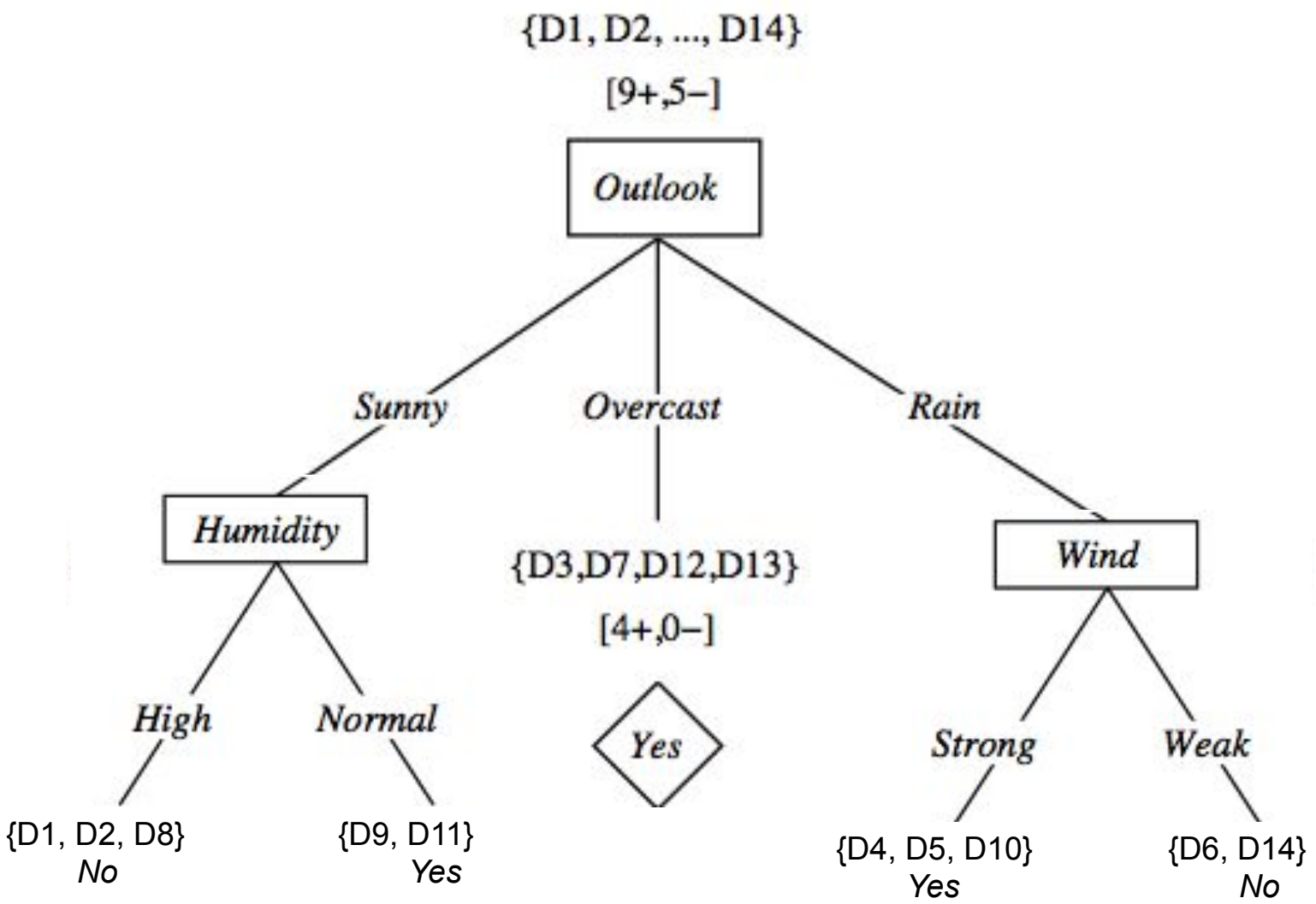
# Second step

- Working on *Outlook=Sunny* node:

$$Gain(S_{Sunny}, Humidity) = 0.970 - 3/5 \times 0.0 - 2/5 \times 0.0 = 0.970$$
$$Gain(S_{Sunny}, Wind) = 0.970 - 2/5 \times 1.0 - 3.5 \times 0.918 = 0\ .019$$
$$Gain(S_{Sunny}, Temp.) = 0.970 - 2/5 \times 0.0 - 2/5 \times 1.0 - 1/5 \times 0.0 = 0.570$$

- *Humidity* provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of *Humidity*
  - partition the training samples according to the value of *Humidity*

# Second and third steps



# Impurity metrics

- Impurity $\phi(p, 1-p)$
  - $\phi(1/2, 1/2) \geq \phi(p, 1-p)$, for any $p \in [0, 1]$.
  - $\phi(0, 1) = \phi(1, 0) = 0$.
  - $\phi(p, 1-p)$ is increasing in $p$ on $[0, 1/2]$ and decreasing in $p$ on $[1/2, 1]$.
  - Entropy $\phi(p, 1-p) = -p \log_2 p - (1-p) \log_2 (1-p)$
  - Gini-index $\phi(p, 1-p) = 2p(1-p)$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$

  - Misclassication error $\phi(p, 1-p) = 1 - \max(p, 1-p)$

# Best Split

- If node *m* is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: $N_{mj}$ of $N_m$ take branch *j*. $N^i_{mj}$ belong to $C_i$

$$\hat{P}(C_i \mid \boldsymbol{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

# Classification tree construction

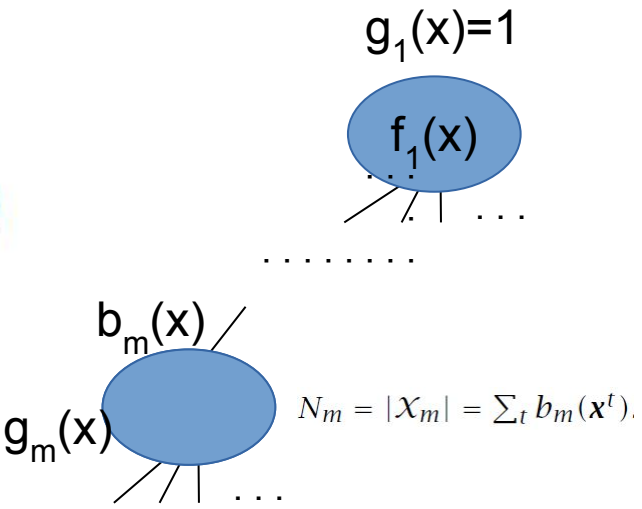
- When to stop the construction?

```
GenerateTree(X)
    If Pure (X)                    equation 9.3 */
        Create leaf labelled by majority class in X
        Return
    i ← SplitAttribute(X)
    For each branch of x_i
        Find X_i falling in branch
        GenerateTree(X_i)
```

# Regression Trees

- For each instance, define

$$b_m(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \mathcal{X}_m\text{: } \boldsymbol{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$




- Use mean to compute output

$$g_m = \frac{\sum_t b_m(\boldsymbol{x}^t) r^t}{\sum_t b_m(\boldsymbol{x}^t)}$$

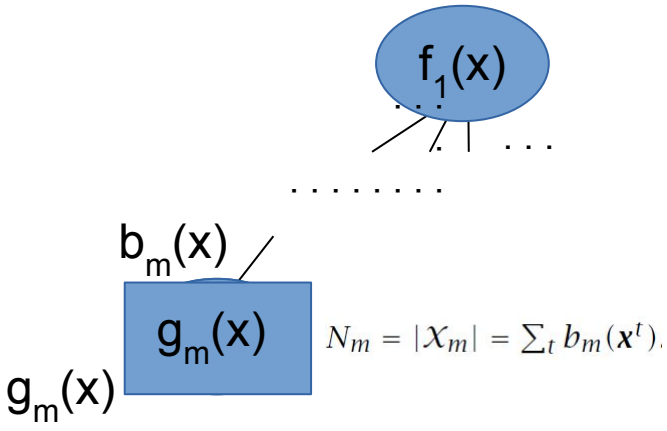
- Error at node m:

$$E_m = \frac{1}{N_m} \sum_t (r^t - g_m)^2 b_m(\boldsymbol{x}^t)$$

# Regression Trees

- Constructed in a similar way
- Goodness of split: mean square error $E_m = \frac{1}{N_m} \sum_t (r^t - g_m)^2 b_m(\mathbf{x}^t)$

- If at a node, the error is acceptable, create a leaf node

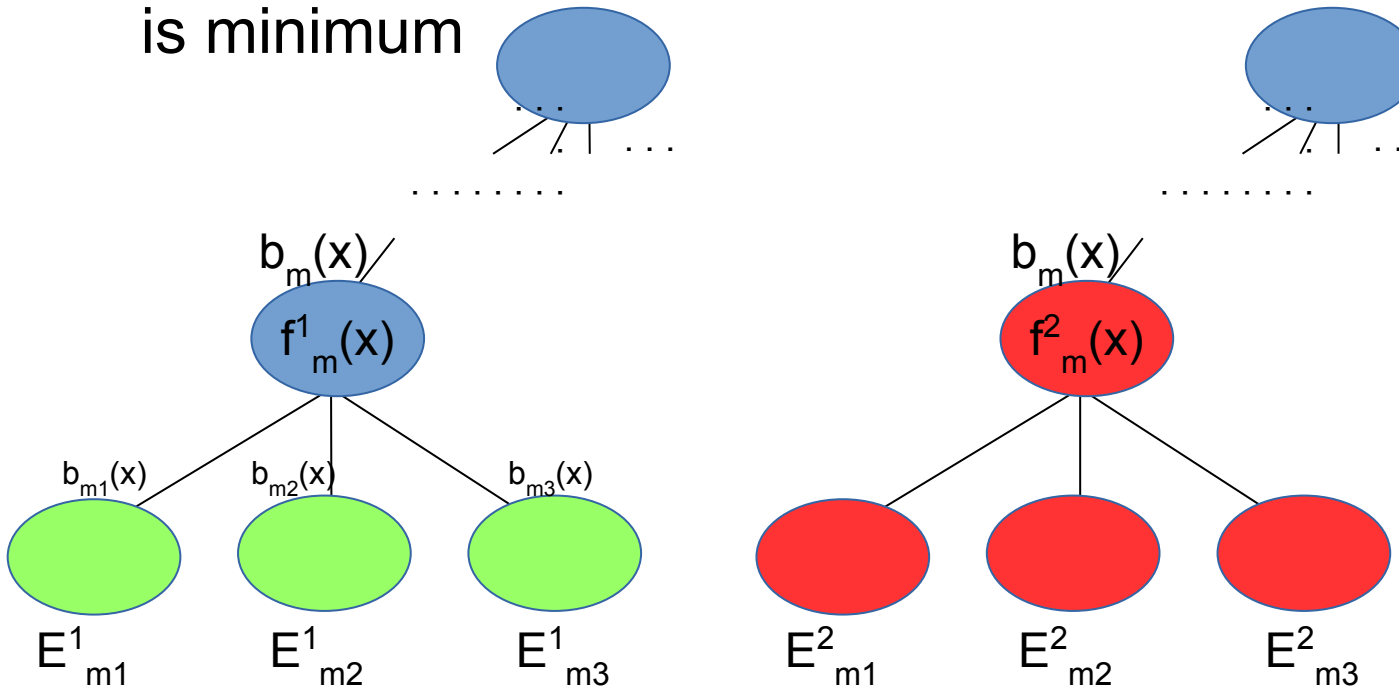



$$g_m = \frac{\sum_t b_m(\mathbf{x}^t) r^t}{\sum_t b_m(\mathbf{x}^t)}$$

# Regression Trees

- Constructed in a similar way
- Goodness of split: mean square error $E_m = \frac{1}{N_m} \sum_t (r^t - g_m)^2 b_m(\mathbf{x}^t)$

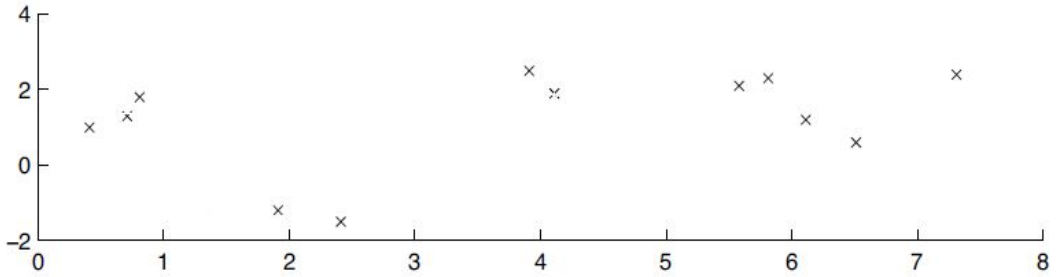
- If error is not acceptable, split m further such that sum of errors is minimum




$$E'_m = \frac{1}{N_m} \sum_j \sum_t (r^t - g_{mj})^2 b_{mj}(\mathbf{x}^t)$$

$$g_m = \frac{\sum_t b_m(\mathbf{x}^t) r^t}{\sum_t b_m(\mathbf{x}^t)}$$

# Model Selection in Trees:



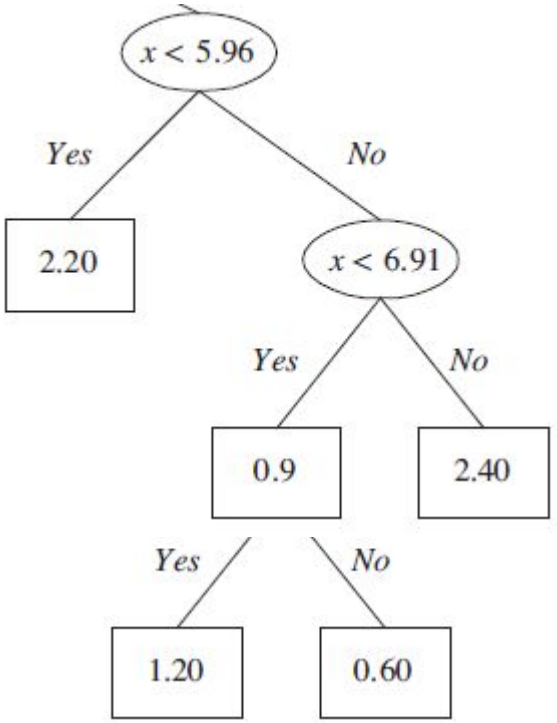

$$g_m = \frac{\sum_t b_m(\mathbf{x}^t) r^t}{\sum_t b_m(\mathbf{x}^t)}$$

What can you do instead of mean?

# Pruning Trees

- Remove subtrees for better generalization (decrease variance)

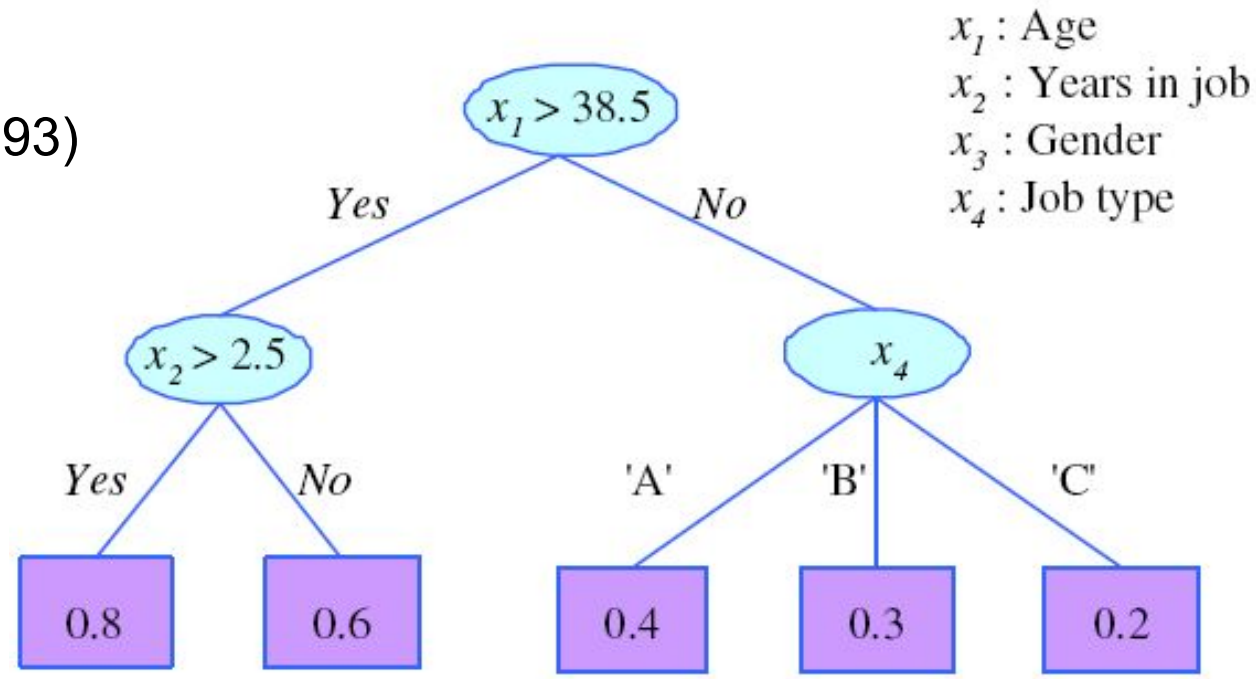
- Prepruning: Early stopping
    - If $N_m < 5\%$
- Postpruning:
    - Grow the whole tree (pure leaves)
    - Overfitting subtrees – pruning set
        - Each subtree replaced by a leaf node
- Prepruning is faster,
- Postpruning is more accurate (requires a separate pruning set)



# Rule Extraction from Trees

Feature extraction – interpretability

C4.5Rules
(Quinlan, 1993)




R1: IF (age>38.5) AND (years-in-job>2.5) THEN $y$ =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN $y$ =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN $y$ =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN $y$ =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN $y$ =0.2

# Learning Rules

- Rule induction is similar to tree induction but
- tree induction is breadth-first,
- rule induction is depth-first; one rule at a time
- Rule set contains rules; rules are conjunctions of terms
- Rule covers an example if all terms of the rule evaluate to true for the example
- Sequential covering: Generate rules one at a time until all positive examples are covered
- IREP (Fürnkrantz and Widmer, 1994), Ripper (Cohen, 1995)

```
Ripper(Pos,Neg,k)
   RuleSet ← LearnRuleSet(Pos,Neg)
   For k times
      RuleSet ← OptimizeRuleSet(RuleSet,Pos,Neg)
LearnRuleSet(Pos,Neg)
   RuleSet ← ∅
   DL ← DescLen(RuleSet,Pos,Neg)
   Repeat
      Rule ← LearnRule(Pos,Neg)
      Add Rule to RuleSet
      DL' ← DescLen(RuleSet,Pos,Neg)
      If DL'>DL+64
         PruneRuleSet(RuleSet,Pos,Neg)
         Return RuleSet
      If DL'<DL DL ← DL'
         Delete instances covered from Pos and Neg
   Until Pos = ∅
   Return RuleSet
```

```
PruneRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet in reverse order
    DL ← DescLen(RuleSet,Pos,Neg)
    DL' ← DescLen(RuleSet-Rule,Pos,Neg)
    IF DL'<DL Delete Rule from RuleSet
  Return RuleSet
OptimizeRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet
      DL0 ← DescLen(RuleSet,Pos,Neg)
      DL1 ← DescLen(RuleSet-Rule+
       ReplaceRule(RuleSet,Pos,Neg),Pos,Neg)
      DL2 ← DescLen(RuleSet-Rule+
       ReviseRule(RuleSet,Rule,Pos,Neg),Pos,Neg)
      If DL1=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
          add ReplaceRule(RuleSet,Pos,Neg)
      Else If DL2=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
          add ReviseRule(RuleSet,Rule,Pos,Neg)
  Return RuleSet
```

# Multivariate Trees