

# Foundations of Machine Learning

## Classifiers

Aug/Sep 2021

Vineeth N Balasubramanian

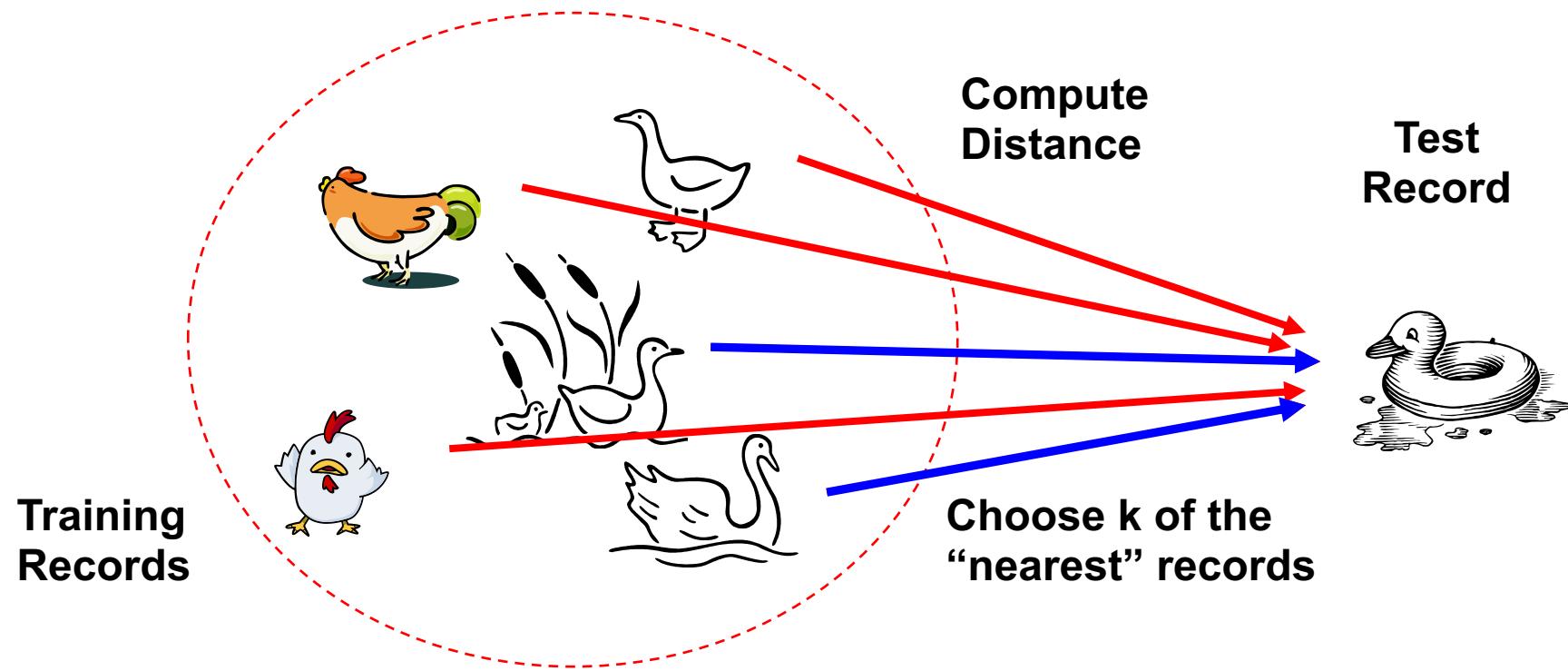


# Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

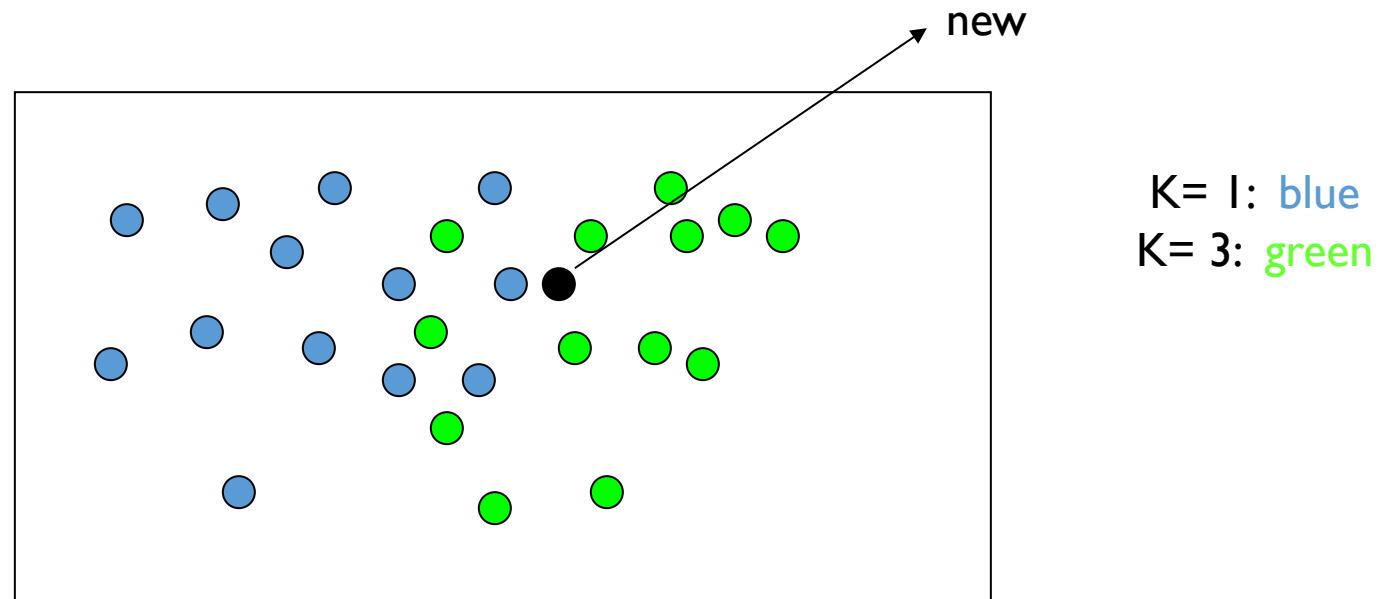
# k-Nearest Neighbors

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



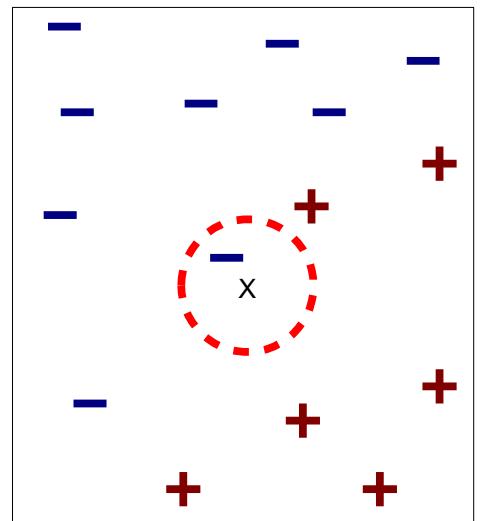
# k-Nearest Neighbors

- Majority vote within the k nearest neighbors

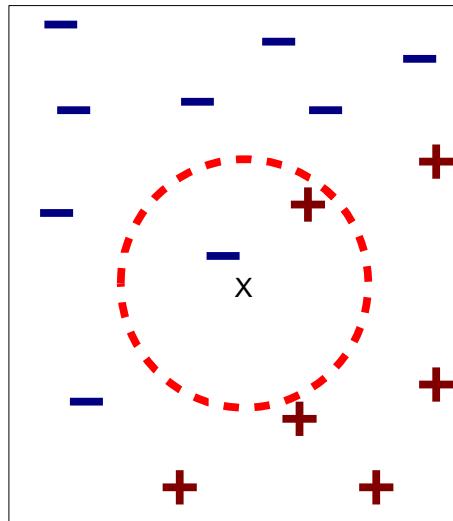


# k-Nearest Neighbors

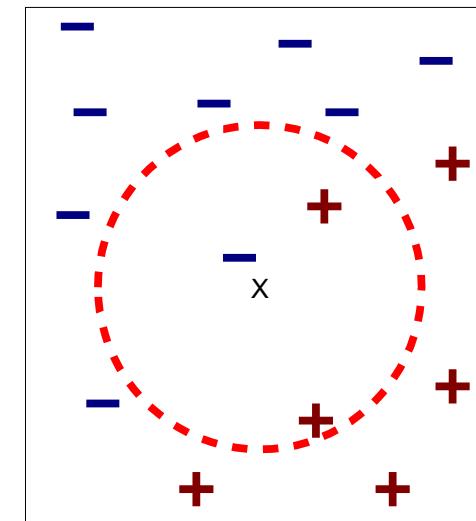
- Choosing k is important
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

# k-Nearest Neighbors

- An arbitrary instance is represented by  $(a_1(x), a_2(x), a_3(x), \dots, a_n(x))$ 
  - $a_i(x)$  denotes features
- Euclidean distance between two instances
  - $d(x_i, x_j) = \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$
- In case of continuous-valued target function
  - Mean value of k nearest training examples

# How to determine k

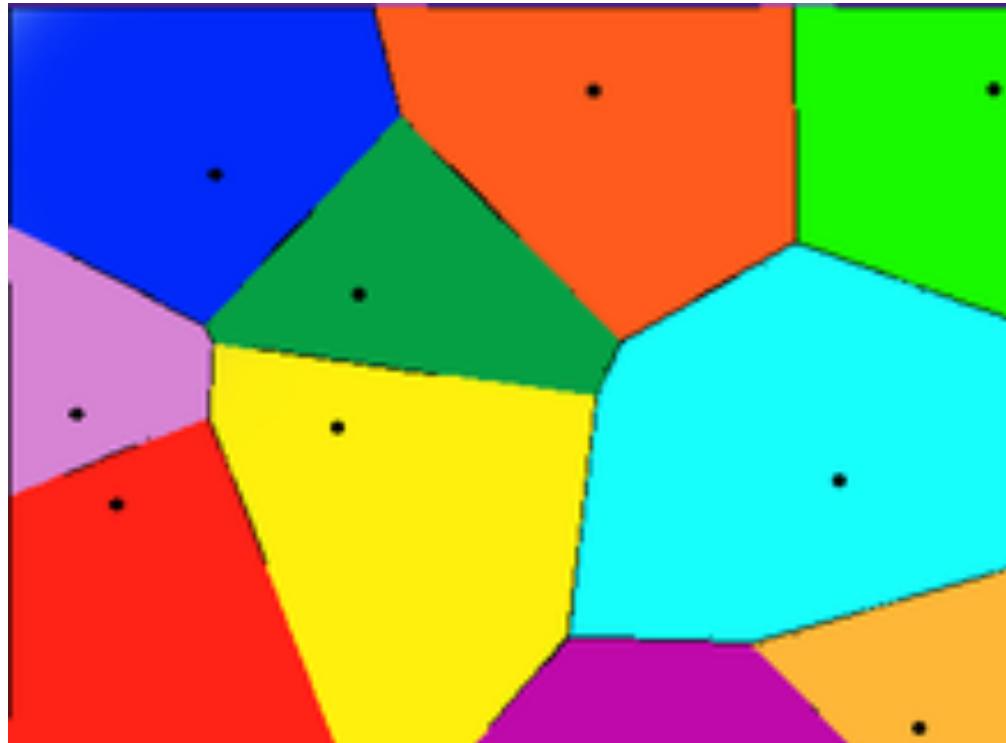
- Determined experimentally
  - Start with  $k=1$  and use a test set to validate the error rate of the classifier
  - Repeat with  $k=k+2$
  - Choose the value of  $k$  for which the error rate is minimum
  - Note:  $k$  typically an odd number to avoid ties in binary classification

# k-Nearest Neighbors

- Eager Learning (**Induction**)
  - Explicit description of target function on the whole training set
- Instance-based Learning (**Transduction**)
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
  - Referred to as “Lazy” learning

**Similar Keywords:** K-Nearest Neighbors, Memory-Based Reasoning, Example-Based Reasoning, Instance-Based Learning, Case-Based Reasoning, Lazy Learning

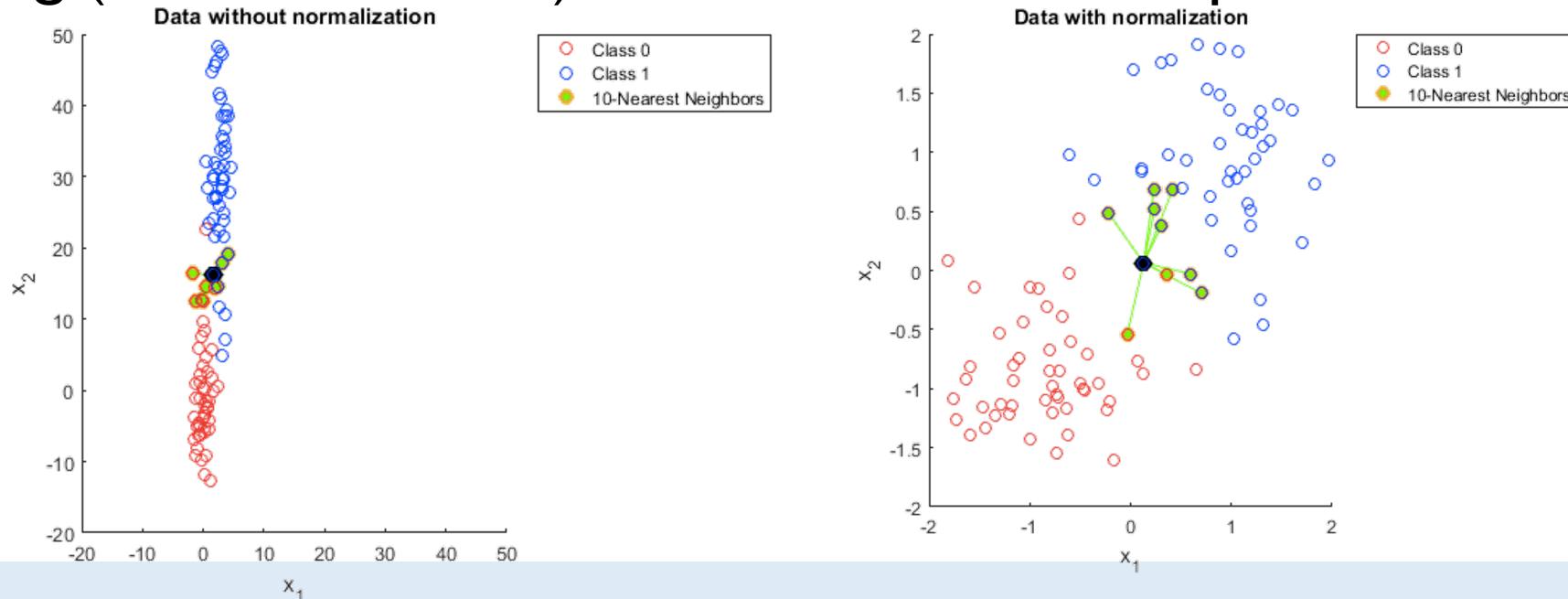
# Voronoi Diagram



Decision surface formed by the training examples!

# Improvements

- Distance-Weighted Nearest Neighbors
  - Assign weights to the neighbors based on their ‘distance’ from the query point (E.g., weight ‘may’ be inverse square of the distances)
- Scaling (**normalization**) attributes for fair computation of distances



# Improvements

- Distance-Weighted Nearest Neighbors
  - Assign weights to the neighbors based on their ‘distance’ from the query point (E.g., weight ‘may’ be inverse square of the distances)
- Scaling (**normalization**) attributes for fair computation of distances
- Measure “closeness” differently
- Finding “close” examples in a large training set quickly
  - E.g. Efficient memory indexing using kd-trees

# k-NN: Summary

- **Pros**

- Highly effective inductive inference method for noisy training data and complex target functions
- Target function for a whole space may be described as a combination of less complex local approximations
- Trains very fast (“Lazy” learner)

- **Cons**

- Curse of dimensionality
  - In higher dimensions, all the data points lie on the surface of the unit hypersphere (the inside is empty!) Check: <http://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/chap1-high-dim-space.pdf>
- Storage: all training examples are saved in memory
  - A decision tree or linear classifier is much smaller
- Slow at query time
  - Can be overcome and presorting and indexing training samples

# Convergence of 1-NN

$$P(\text{knnError})$$

$$= 1 - \Pr(y = y_1)$$

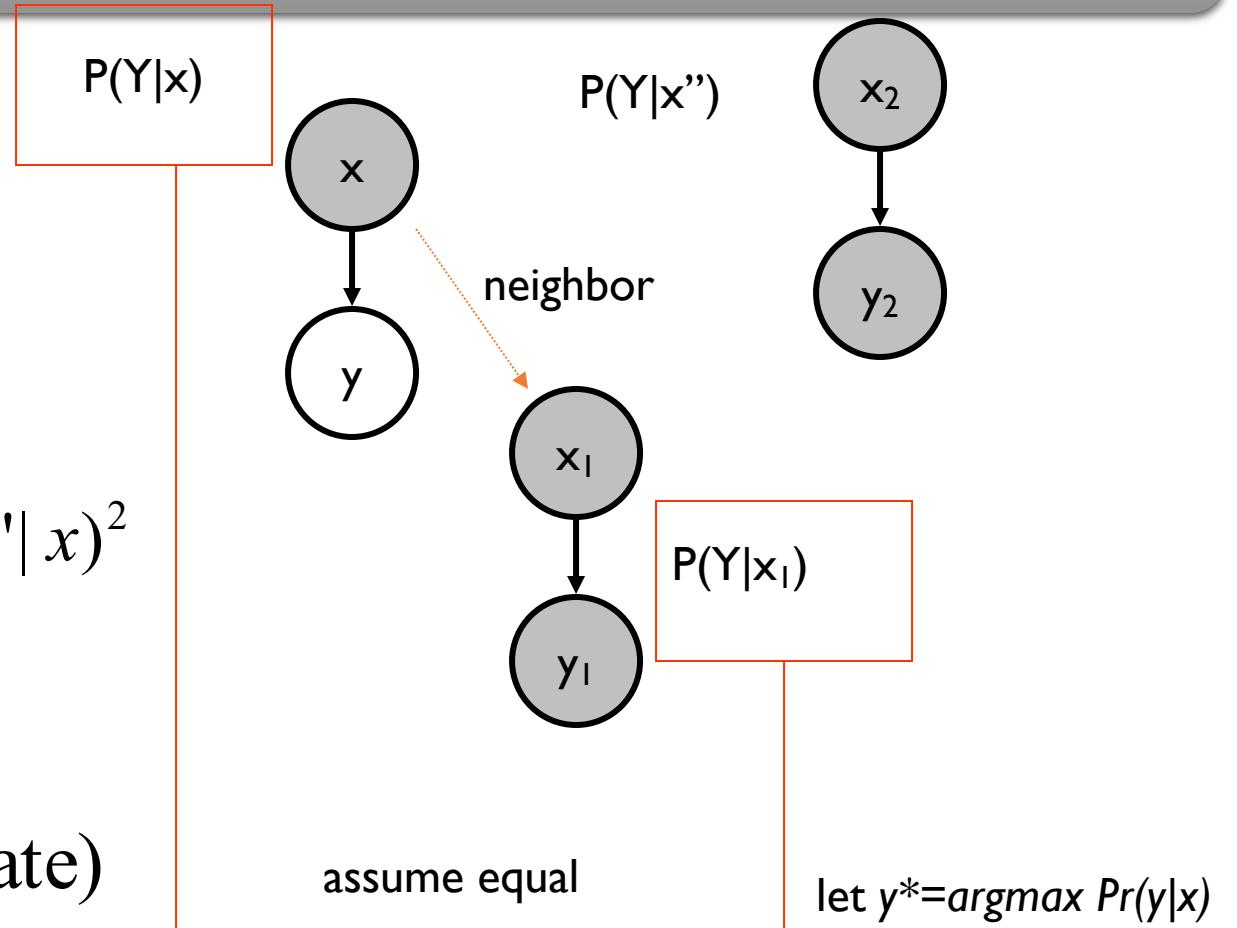
$$= 1 - \sum_{y'} \Pr(Y = y' | x)^2$$

$$= 1 - \Pr(y^* | x)^2 - \sum_{y' \neq y^*} \Pr(Y = y' | x)^2$$

$$\dots$$

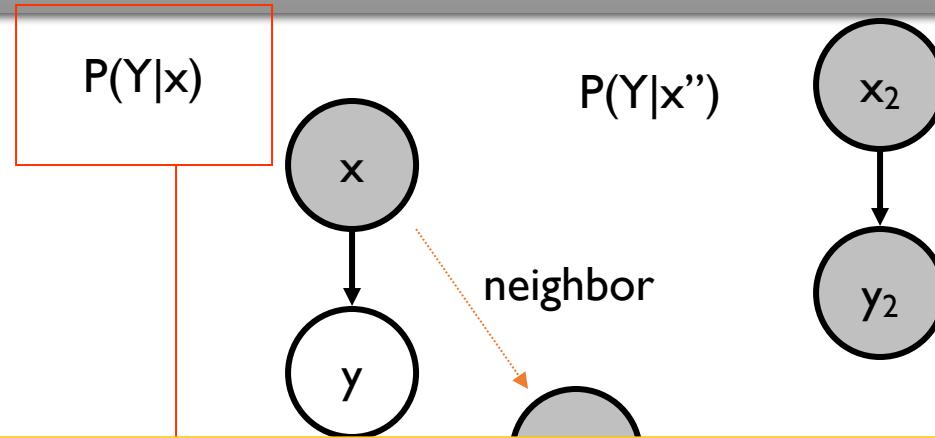
$$\leq 2(1 - \Pr(y^* | x))$$

$$= 2(\text{Bayes optimal error rate})$$



# Convergence of 1-NN

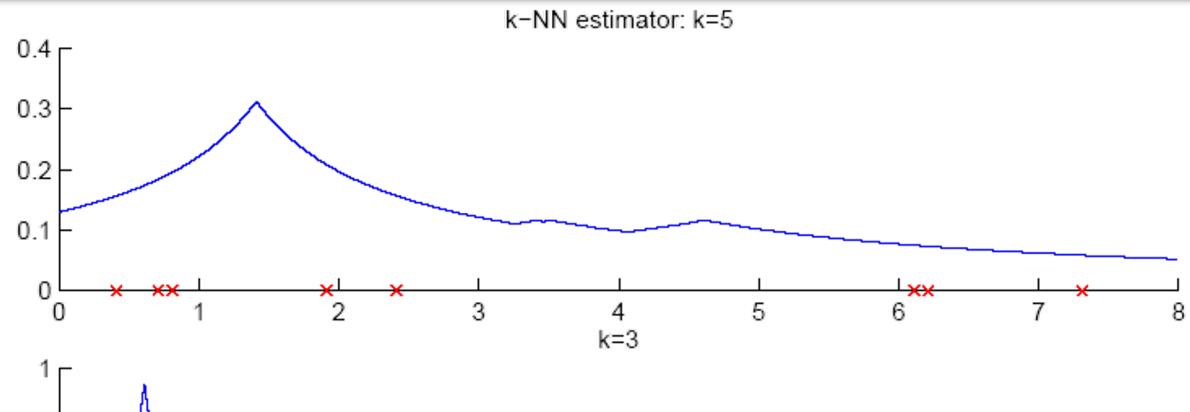
$$\begin{aligned} P(\text{knnError}) &= 1 - \Pr(y = y_1) \\ &= 1 - \sum_{y'} \Pr(Y = y' | x)^2 \end{aligned}$$



Possible to show that: as the size of training data set approaches infinity, the one nearest neighbor classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data). We will see this later.

# Non-parametric Density Estimation using kNNs

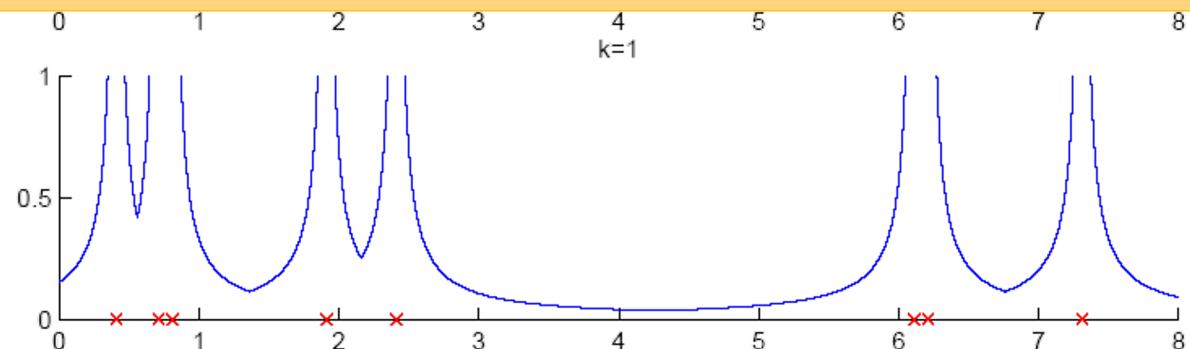
- **K-Nearest Neighbor estimator**
- Instead of fixing bin width  $h$  and counting the number of instances, fix



More later when we move to unsupervised learning

$$\hat{p}(x) = \frac{\kappa}{2N d_k(x)}$$

$d_k(x)$ , distance to  $k$ th closest instance to  $x$



Source: Ethem Alpaydin, *Introduction to Machine Learning*, 3<sup>rd</sup> Edition (Slides)

# Classification Methods

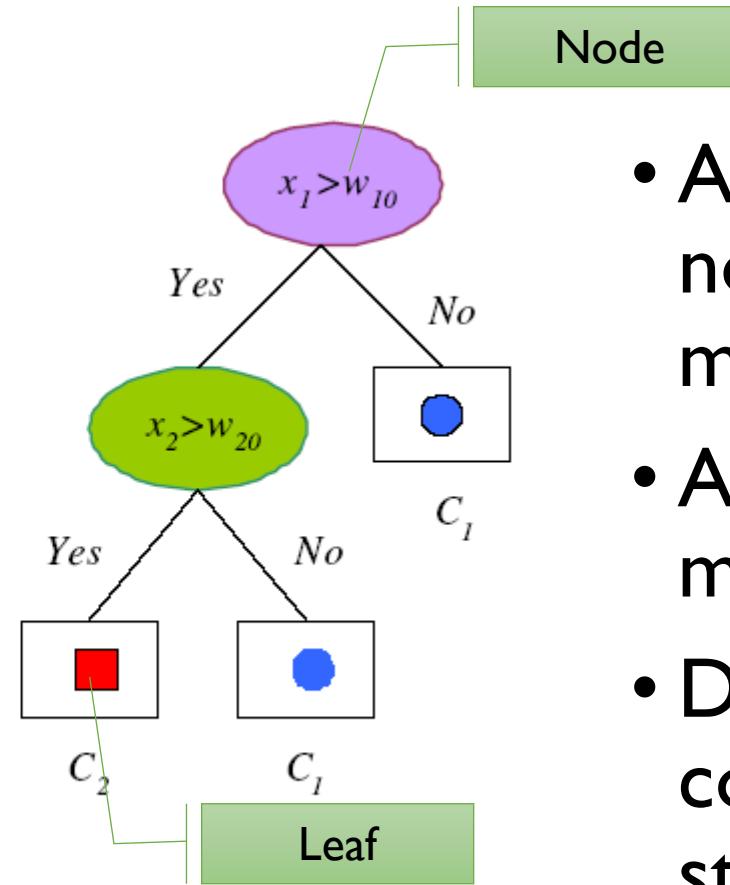
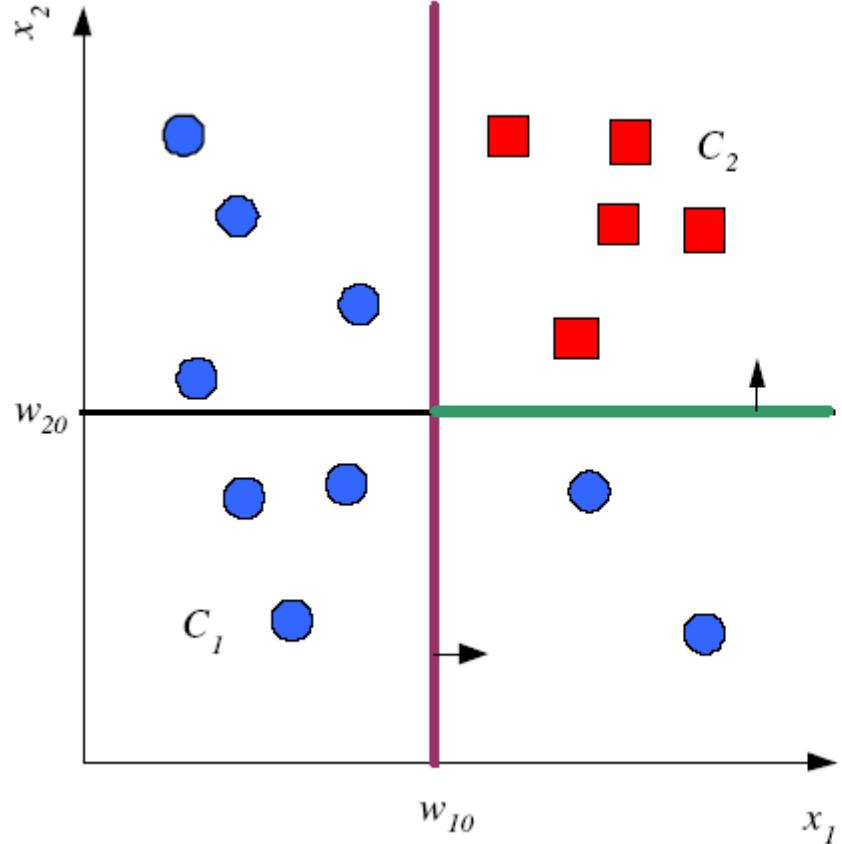
- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

# Example

*PlayTennis:* training examples

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

# Decision Trees



- An efficient nonparametric method
- A hierarchical model
- Divide-and-conquer strategy

Source: Ethem Alpaydin, *Introduction to Machine Learning*, 3<sup>rd</sup> Edition (Slides)

# Divide and Conquer

- Internal decision nodes
  - **Univariate:** Uses a single attribute,  $x_i$ 
    - Numeric  $x_i$  :
      - Binary split :  $x_i > w_m$
    - Discrete  $x_i$  :
      - $n$ -way split for  $n$  possible values
  - **Multivariate:** Uses more than one attributes,  $\mathbf{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric;  $r$  average, or local fit
- Learning is **greedy**; find the best split recursively

Source: Ethem Alpaydin, *Introduction to Machine Learning*, 3<sup>rd</sup> Edition (Slides)

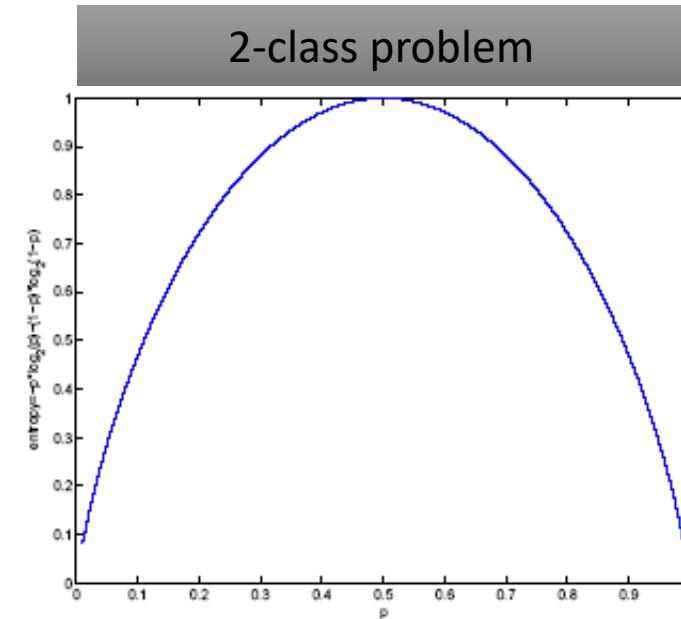
# Classification Trees (C4.5, J48)

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

$$\hat{P}(C_i | \mathbf{x}, m) \equiv p_m^i = \frac{N_m^i}{N_m}$$

- Node  $m$  is **pure** if  $p_m^i$  is 0 or 1
- Measure of **impurity** is **entropy**

$$I_m = - \sum_{i=1}^K p_m^i \log_2 p_m^i$$



Entropy in information theory specifies the **average (expected) amount of information derived from observing an event**

Source: Ethem Alpaydin, *Introduction to Machine Learning*, 3<sup>rd</sup> Edition (Slides)

# Classification Trees

- 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  $C_i$

$$\hat{P}(C_i | \mathbf{x}, m, j) \equiv p_{mj}^i = \frac{N_{mj}^i}{N_{mj}}$$

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

- **Information Gain:** Expected reduction in impurity measure after split
- Choose the best attribute(s) (**with maximum information gain**) to split the remaining instances and make that attribute a decision node
  - You can use same logic to find best splitting value too

Source: Ethem Alpaydin, *Introduction to Machine Learning*, 3<sup>rd</sup> Edition (Slides)

# Other Measures of Impurity

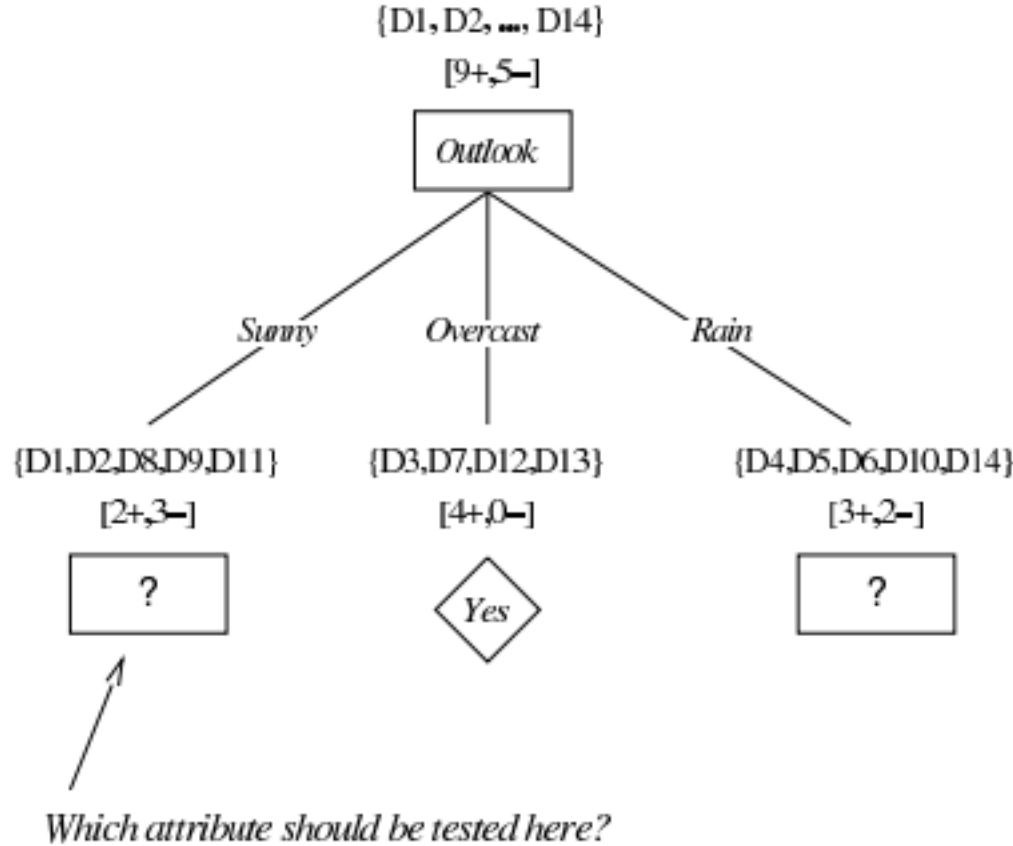
- The properties of functions measuring the **impurity** of a split:
  - $\phi(1/2, 1/2) \geq \phi(p, 1-p)$ , for any  $p \in [0, 1]$
  - $\phi(0, 1) = \phi(1, 0) = 0$
  - $\phi(p, 1-p)$  is increasing in  $p$  on  $[0, \frac{1}{2}]$   
and decreasing in  $p$  on  $[\frac{1}{2}, 1]$
- Examples (other than entropy)
  - **Gini impurity/index:**  $1 - \sum_{j=1}^c p_j^2$

# Decision Trees: Example

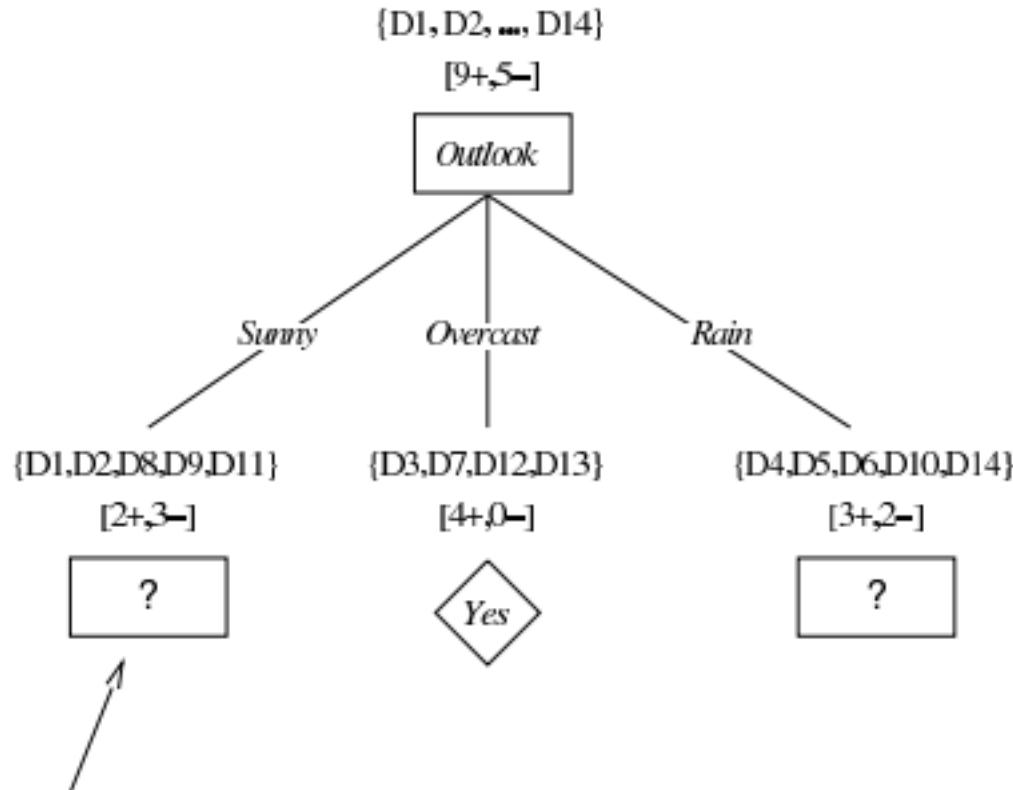
*PlayTennis:* training examples

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

# Decision Trees: Example



# Decision Trees: Example



Which attribute should be tested here?

$$S_{\text{Sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{Sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{Sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{Sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

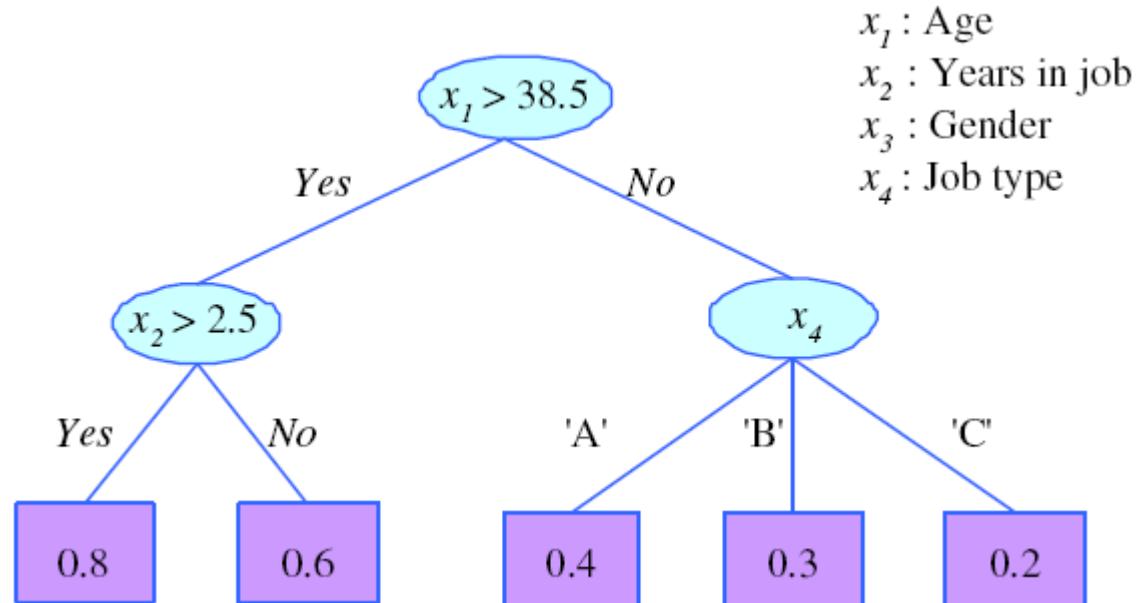
# Overfitting and Generalization

- Overfitting can occur with noisy training examples, also when small numbers of examples are associated with leaf nodes.  
How to handle?
- **Pruning:** Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
  - Prepruning is faster, postpruning is more accurate

# Overfitting and Generalization

- **Occam's Razor principle:** when multiple hypotheses can solve a problem, choose the simplest one
  - a short hypothesis that fits data unlikely to be coincidence
  - a long hypothesis that fits data might be coincidence
- How to select “best” tree:
  - Measure performance over training data
  - Measure performance over separate validation data set
  - **Minimum Description Length:** Minimize  $\text{size}(\text{tree}) + \text{size}(\text{misclassifications}(\text{tree}))$

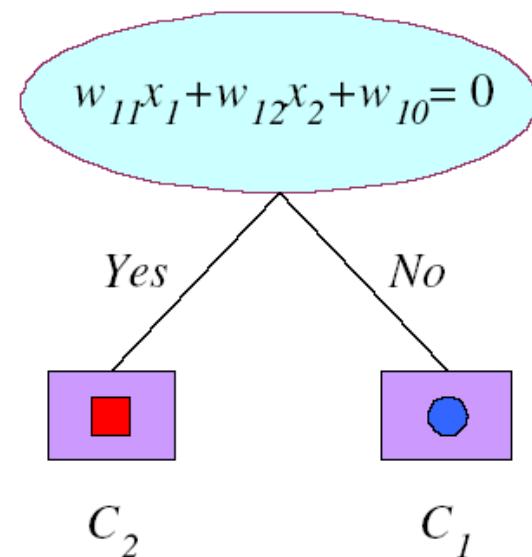
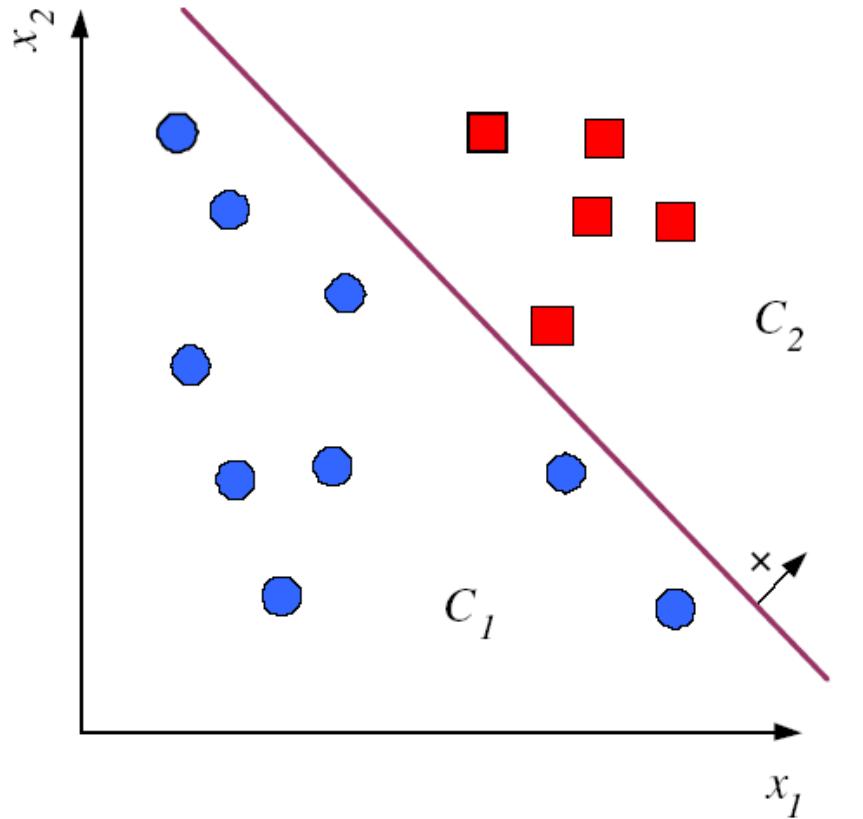
# Rule Extraction from Trees



- 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  $y = 0.2$

- Convert tree to equivalent set of rules
- Prune each rule independently of others, by removing any preconditions that result in improving its estimated accuracy
- Sort final rules into desired sequence for use

# Multivariate Trees



# Readings

- Chapters 8, 9, EA Introduction to ML 2<sup>nd</sup> Edn