BI 2014

Classification

Today

- What is classification
- Overview of classification methods
- Decision trees
- Forests

Predictive data mining

Our task

- Input: data representing objects that have been assigned labels
- Goal: accurately predict labels for new (previously unseen) objects

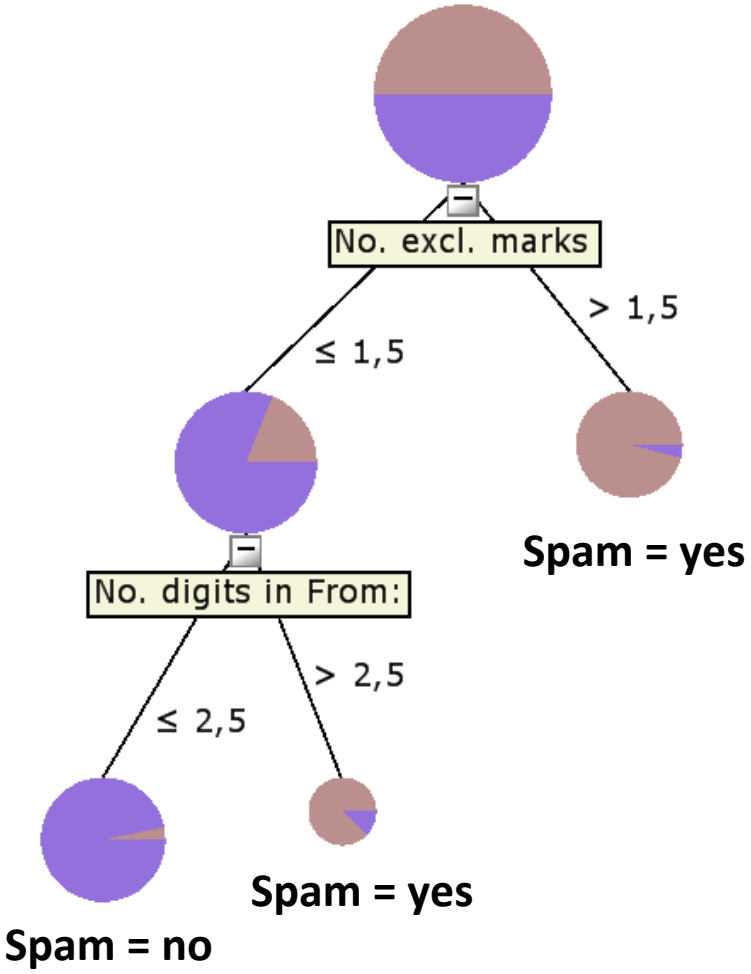
An example: email classification

Features (attributes)

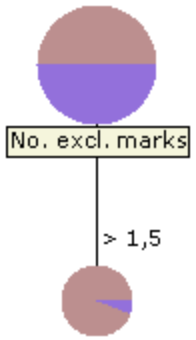
Examples (observations)

Ex.	All caps	No. excl. marks	Missing date	No. digits in From:	Image fraction	Spam
e1	yes	0	no	3	0	yes
e2	yes	3	no	0	0.2	yes
e3	no	0	no	0	1	no
e4	no	4	yes	4	0.5	yes
e5	yes	0	yes	2	0	no
e6	no	0	no	0	0	no

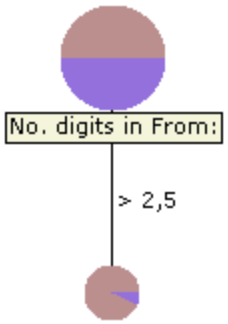
Decision tree



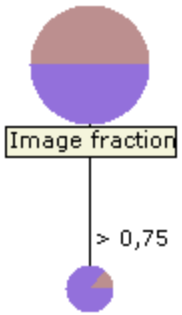
Rules



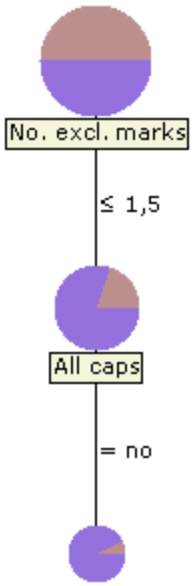
Spam = yes



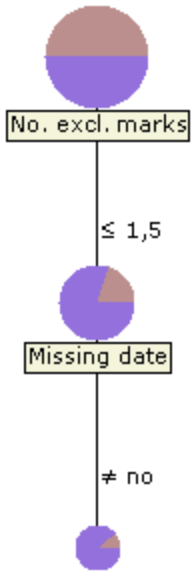
Spam = yes



Spam = no

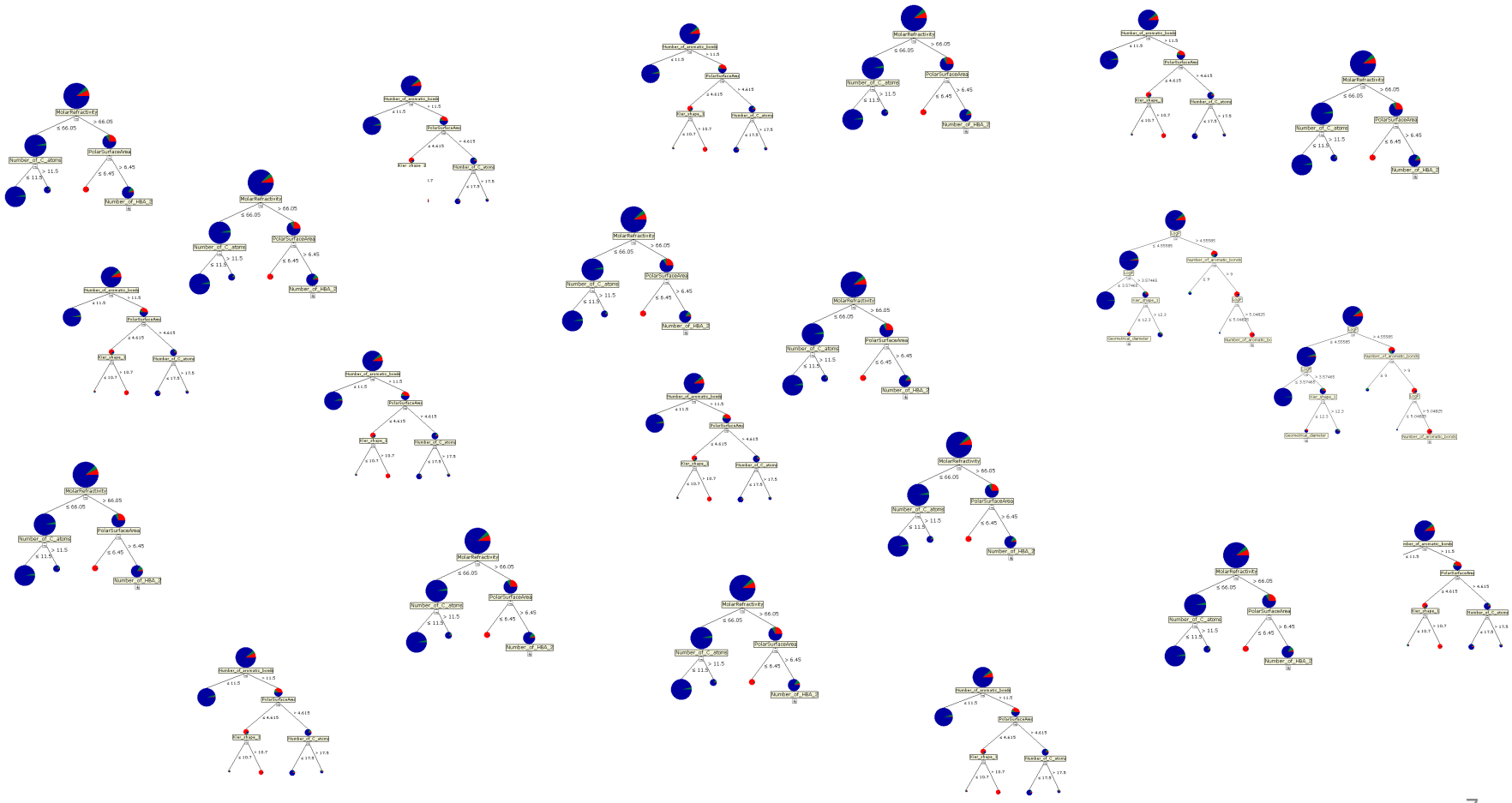


Spam = no



Spam = no

Forests



Classification

- **What is the class of the following patient?**
  - Glucose: 7.5
  - Cholesterol: 280
  - Age: 35
  - Family history: YES

Classification

- What is classification?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Classification by Nearest Neighbor
- Support Vector Machines
- Handling missing values

Classification

- Predicts categorical class labels
- Classifies data (constructs a model) based on
  - the training set
  - the values (class labels) in a classifying attribute
- Uses the model for classifying new data

Why Classification? A motivating application

- Patient classification
  - A hospital wants to build a patient screening system for heart diseases
  - 14 parameters are taken into account
  - The history of past patients is used to train the classifier
  - The classifier provides rules, which identify potential heart risks for new/unseen patients

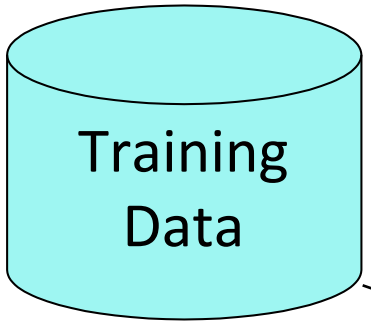
Classification—A Two-Step Process

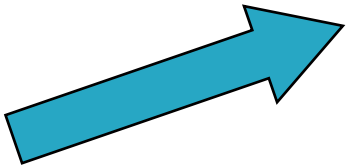
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulas

Classification—A Two-Step Process

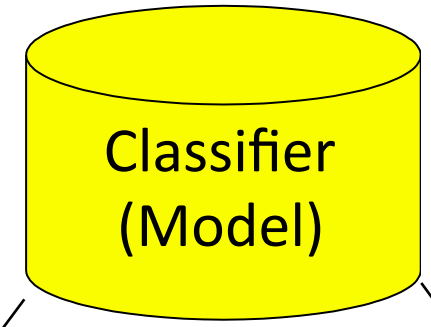
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test samples is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

Classification Process (1): Model Construction





Classification Algorithms

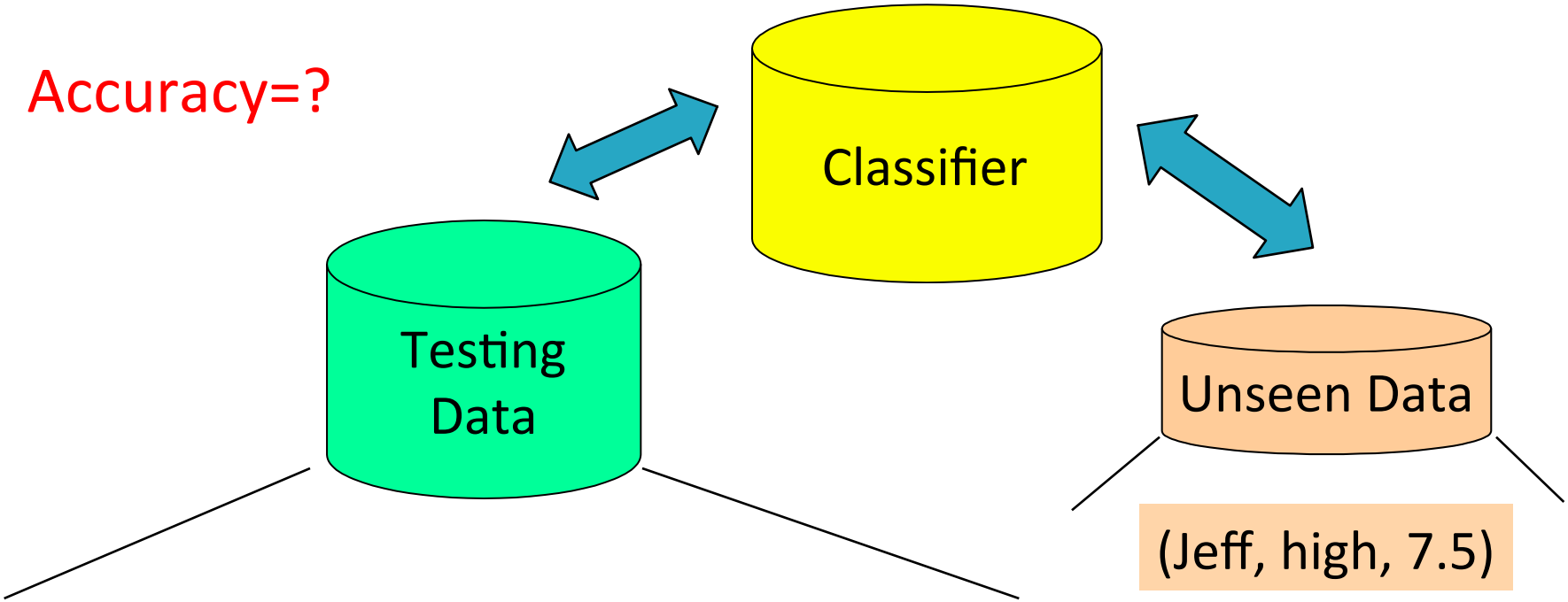


NAME	LDL	Glu	H/ATTACK
Mike	normal	3	no
Mary	high	8	yes
Bill	normal	7	yes
Jim	high	5	yes
Dave	normal	4	no

IF LDL = 'high'
OR Gluc > 6 mmol/lit
THEN Heart attack = 'yes'

Classification Process (2): Use the Model in Prediction

Accuracy=?



NAME	LDL	Glu	H/ATTACK
Tom	normal	5.5	no
Mellisa	low	5	no
George	high	7.5	yes
Joseph	high	5	yes

Heart attack?

Yes

Evaluation of classification models

- Counts of test records that are correctly (or incorrectly) predicted by the classification model
- **Confusion matrix**

Actual Class \ Predicted Class	Class = 1	Class = 0
Class = 1	f_{11}	f_{10}
Class = 0	f_{01}	f_{00}

$$\text{Accuracy} = \frac{\#\text{ correct predictions}}{\text{total }\#\text{ of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\#\text{ wrong predictions}}{\text{total }\#\text{ of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Classification by Decision Tree Induction

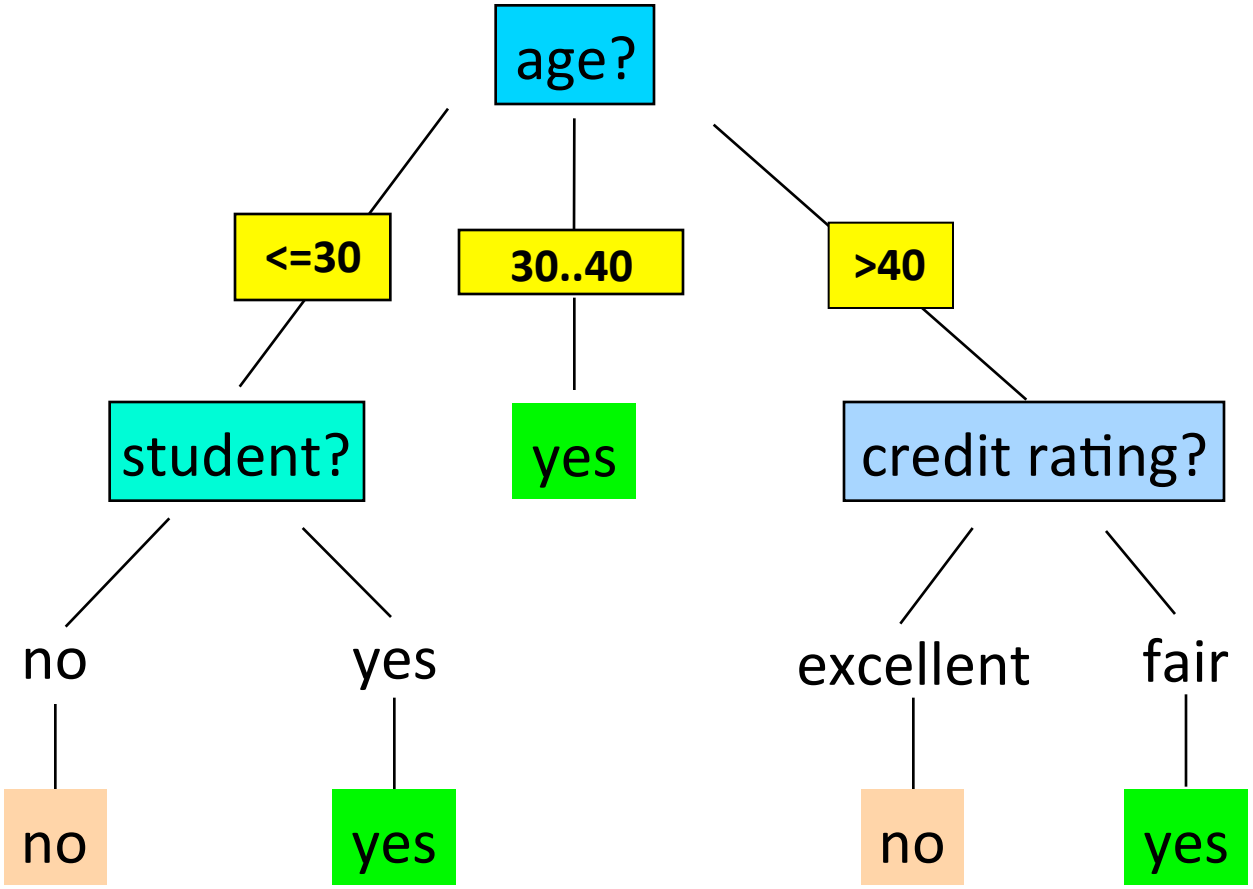
- Decision tree generation consists of two phases:
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree:
  - Classifying an unknown sample
    - Test the attribute values of the sample against the decision tree

Training Dataset

Example

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

Output: A Decision Tree for "*buys_computer*"



Design issues

- How should the training records be split?
- How should the splitting procedure stop?

Splitting methods

- Binary attributes

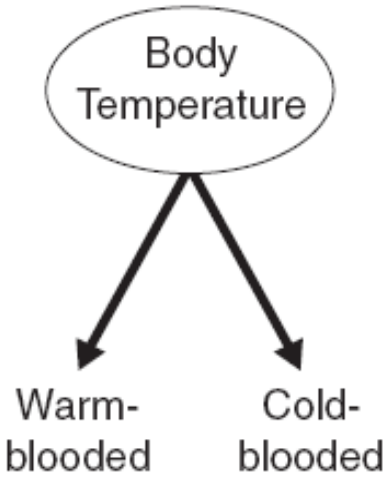


Figure 4.8. Test condition for binary attributes.

Splitting methods

- Nominal attributes

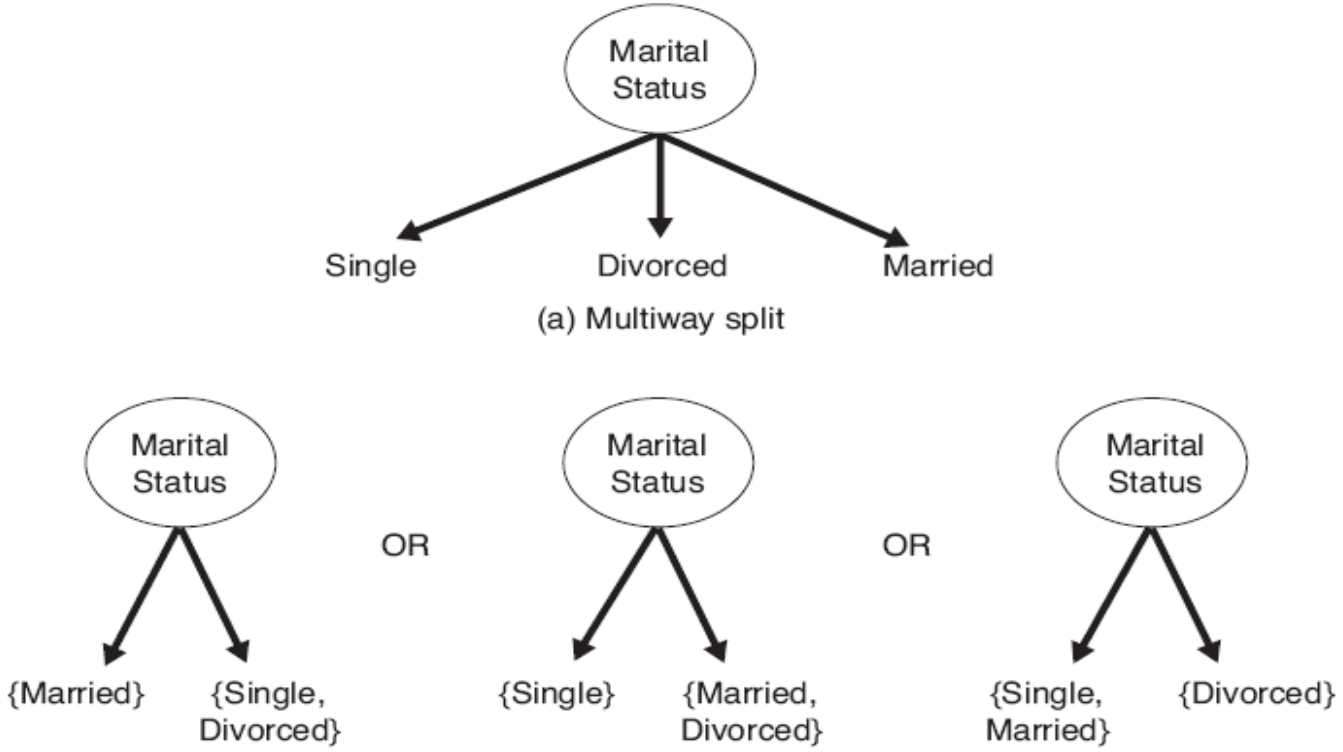


Figure 4.9. Test conditions for nominal attributes.

Splitting methods

- Ordinal attributes

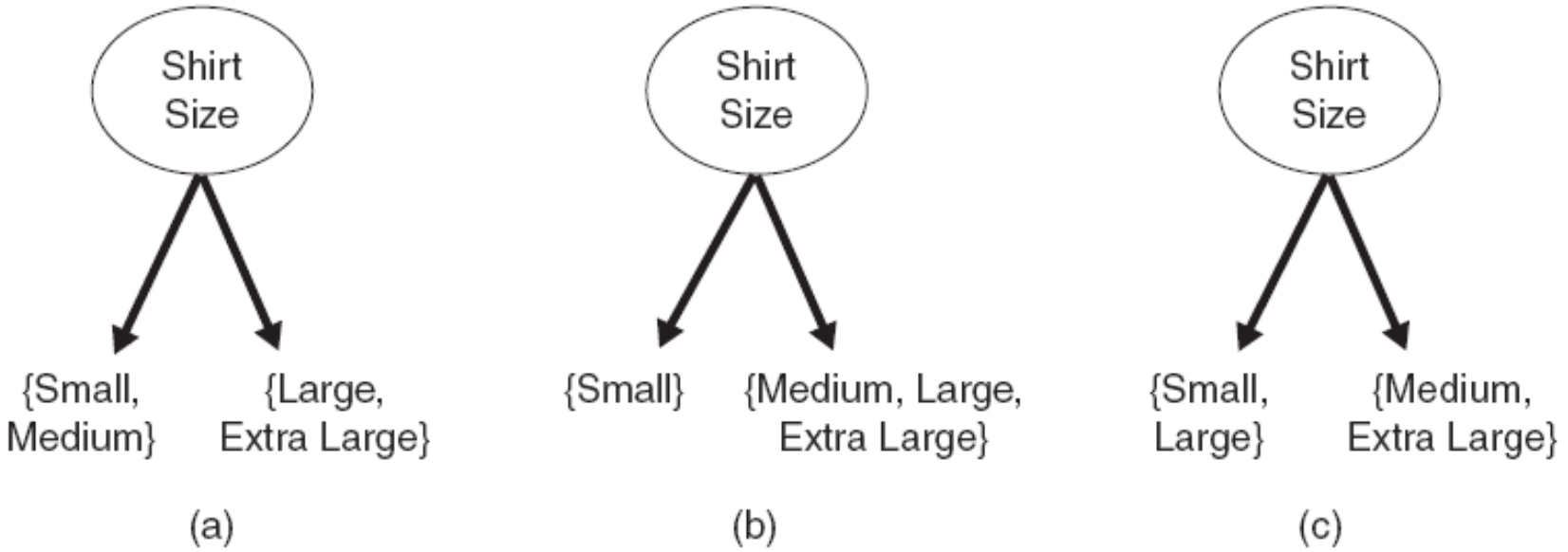


Figure 4.10. Different ways of grouping ordinal attribute values.

Splitting methods

- Continuous attributes

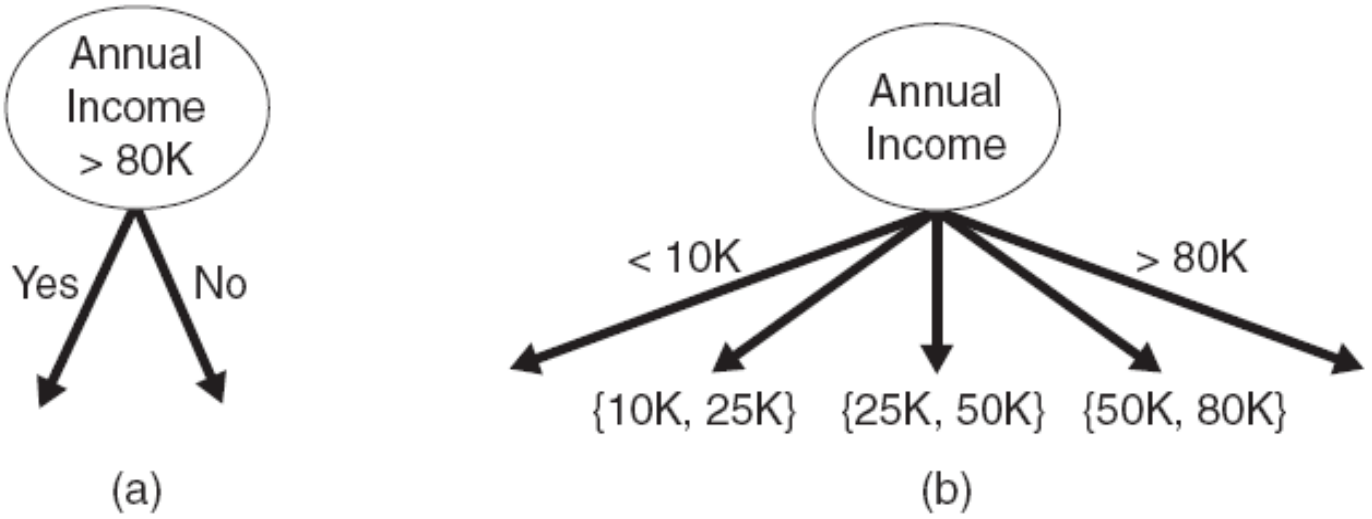
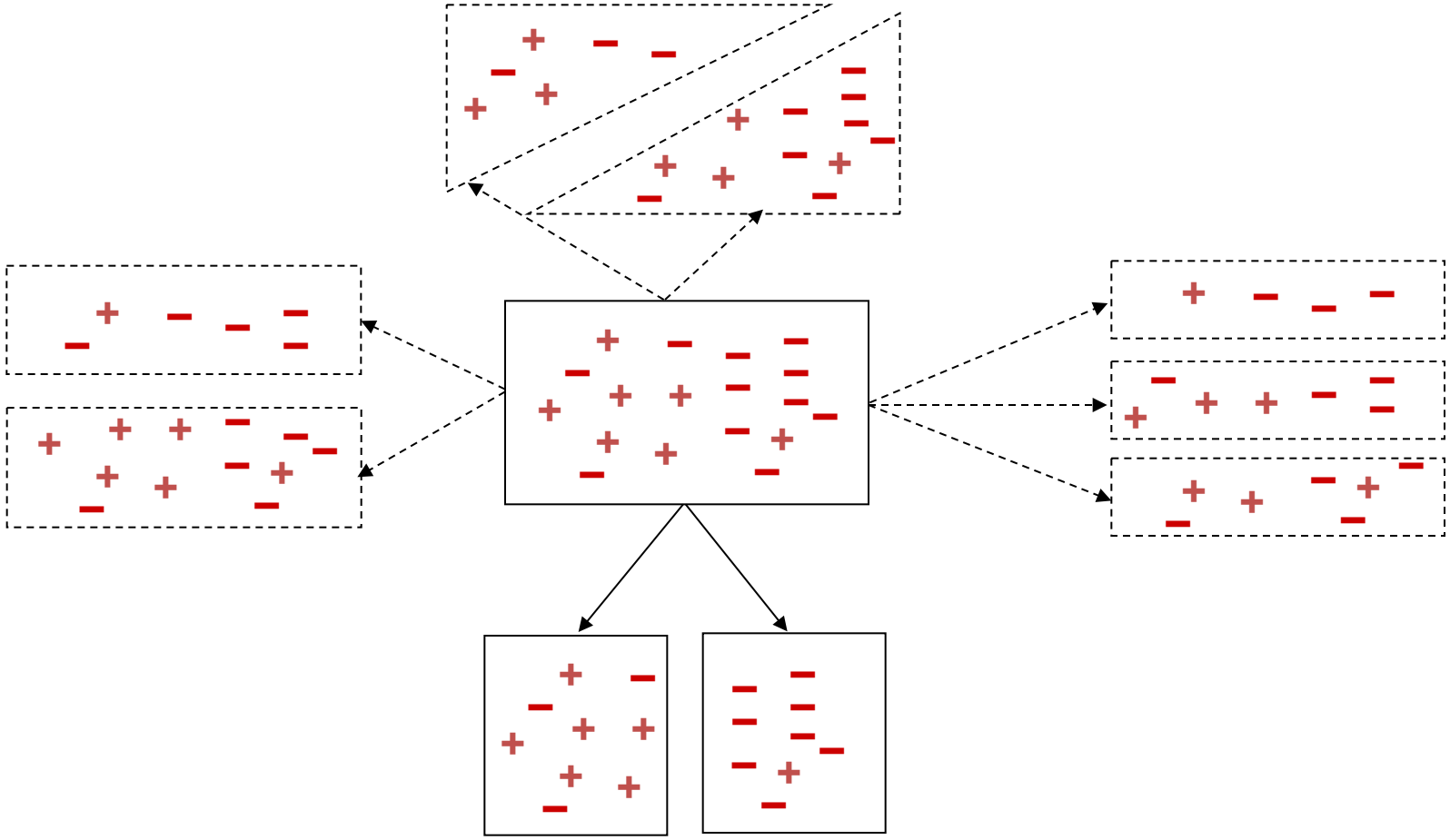


Figure 4.11. Test condition for continuous attributes.

Choice of attribute



Algorithm for Decision Tree Induction Hunt's algorithm

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Samples are partitioned recursively based on selected attributes
  - Test (split) attributes are selected on the basis of a heuristic or *impurity measure* (e.g., information gain)

Algorithm for Decision Tree Induction

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Attribute Selection Measure: Information Gain

- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$

  - where $C_{i,D}$ denotes the set of tuples that belong to class C_i

- **Expected information** (entropy) needed to classify a tuple in D:

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

  - where m is the number of classes

Attribute Selection Measure: Information Gain

- **Information** needed (after using A to split D into v partitions) to classify D:

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

- **Information gained** by branching on attribute A

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

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

samples: yes no

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

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

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

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

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

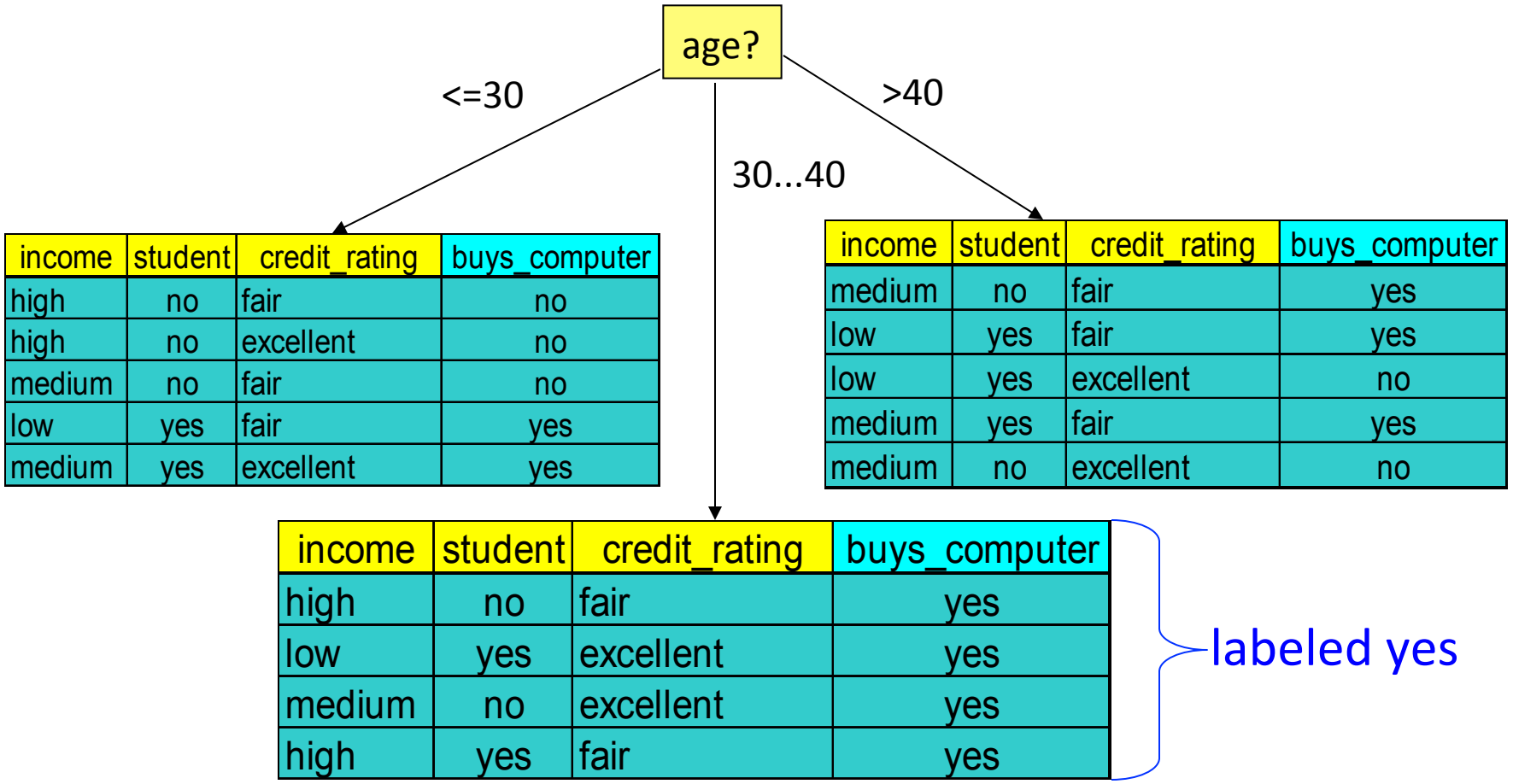
$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Splitting the samples using *age*



Is minimizing impurity/ maximizing Δ enough?

- Information gain favors attributes with large number of values

- A test condition with large number of outcomes may not be desirable
  - # of records in each partition is too small to make predictions

Gain ratio

- A modification that reduces its bias on high-branch attributes
- Gain ratio should be
  - Large when data is evenly spread
  - Small when all data belongs to one branch

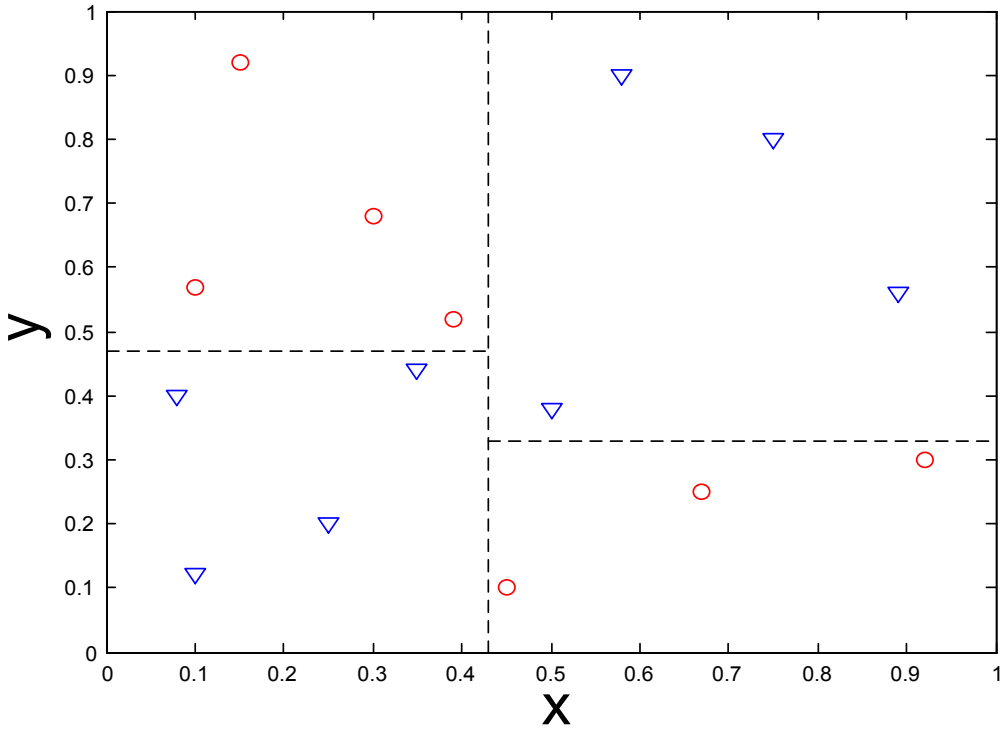
Gain ratio

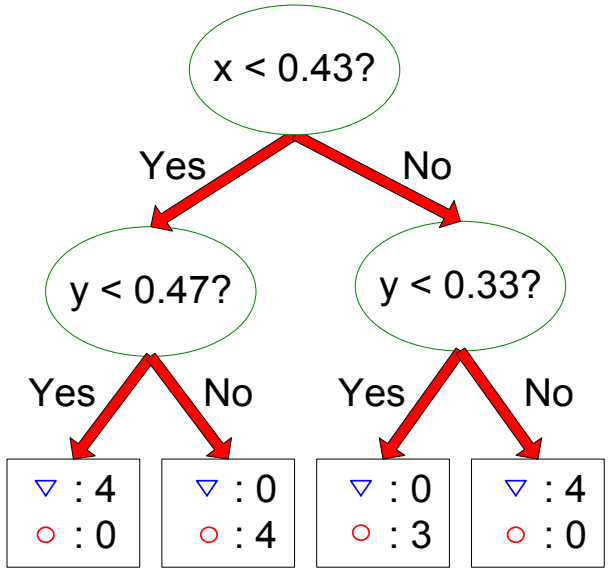
- Takes number and size of branches into account when choosing an attribute
- Corrects the gain Δ by:
  - taking the *intrinsic information* of a split into account
  - i.e., how much info do we need to tell which branch an instance belongs to

Gain ratio

- **Gain ratio = Δ/Splitinfo**
- **SplitInfo = $-\Sigma_{i=1\ldots k} P(v_i)\log(P(v_i))$**
- **k:** total number of branches
- **$P(v_i)$:** probability that an instance belongs to a branch
- If each branch has the same number of records,
  - **$P(v_i) = 1/k$ and SplitInfo = logk**
- Large number of branches → large **SplitInfo** → small gain ratio

Decision boundary for decision trees

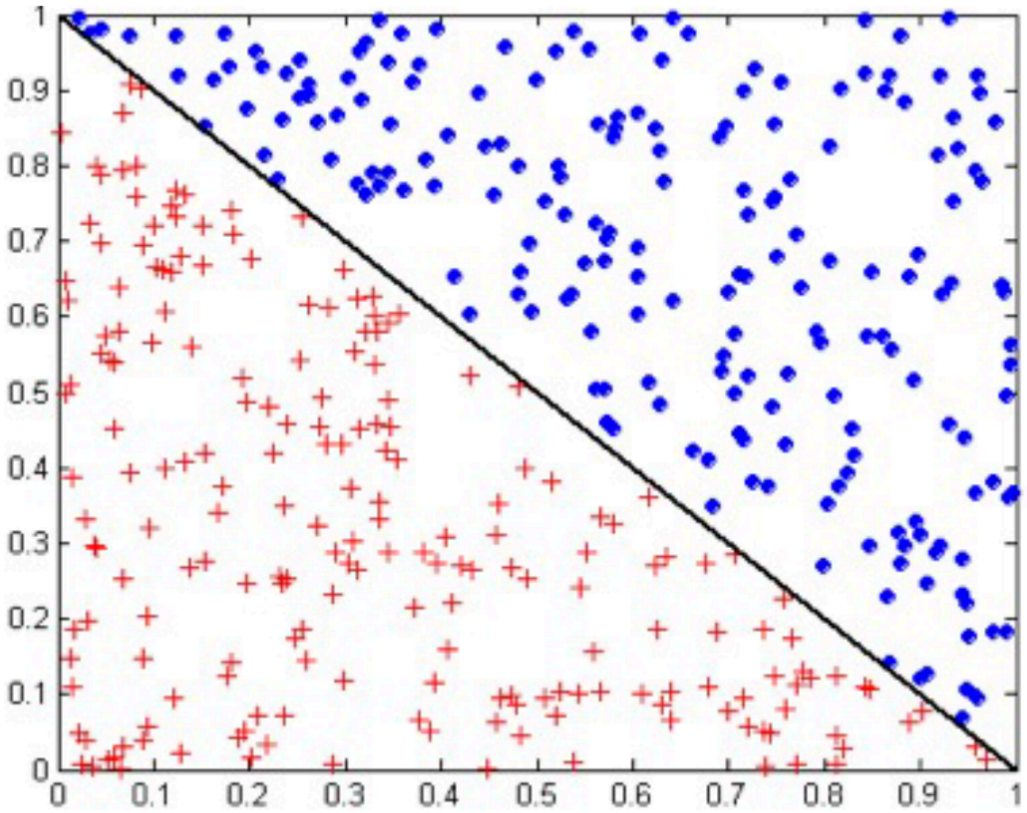




• Border line between two neighboring regions of different classes is known as **decision boundary**

• **Decision boundary in decision trees** is parallel to axes because test condition involves a single attribute at-a-time

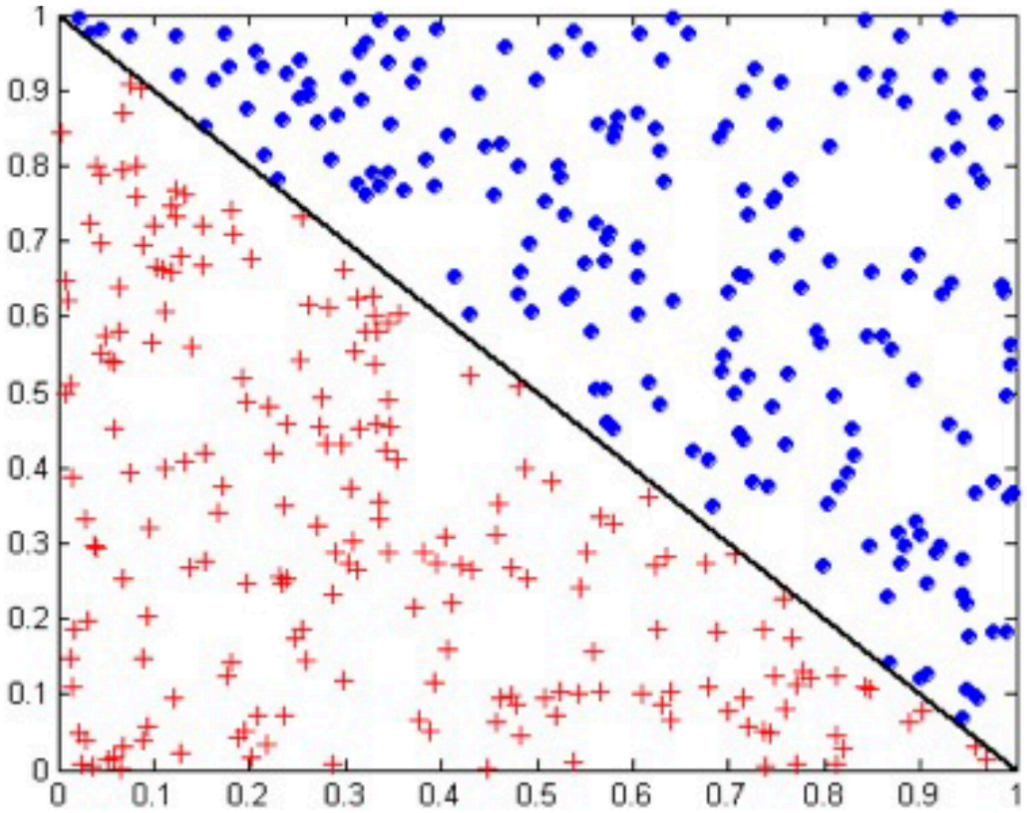
Oblique Decision Trees



?

Not all datasets can be partitioned optimally using test conditions using single attributes!

Oblique Decision Trees

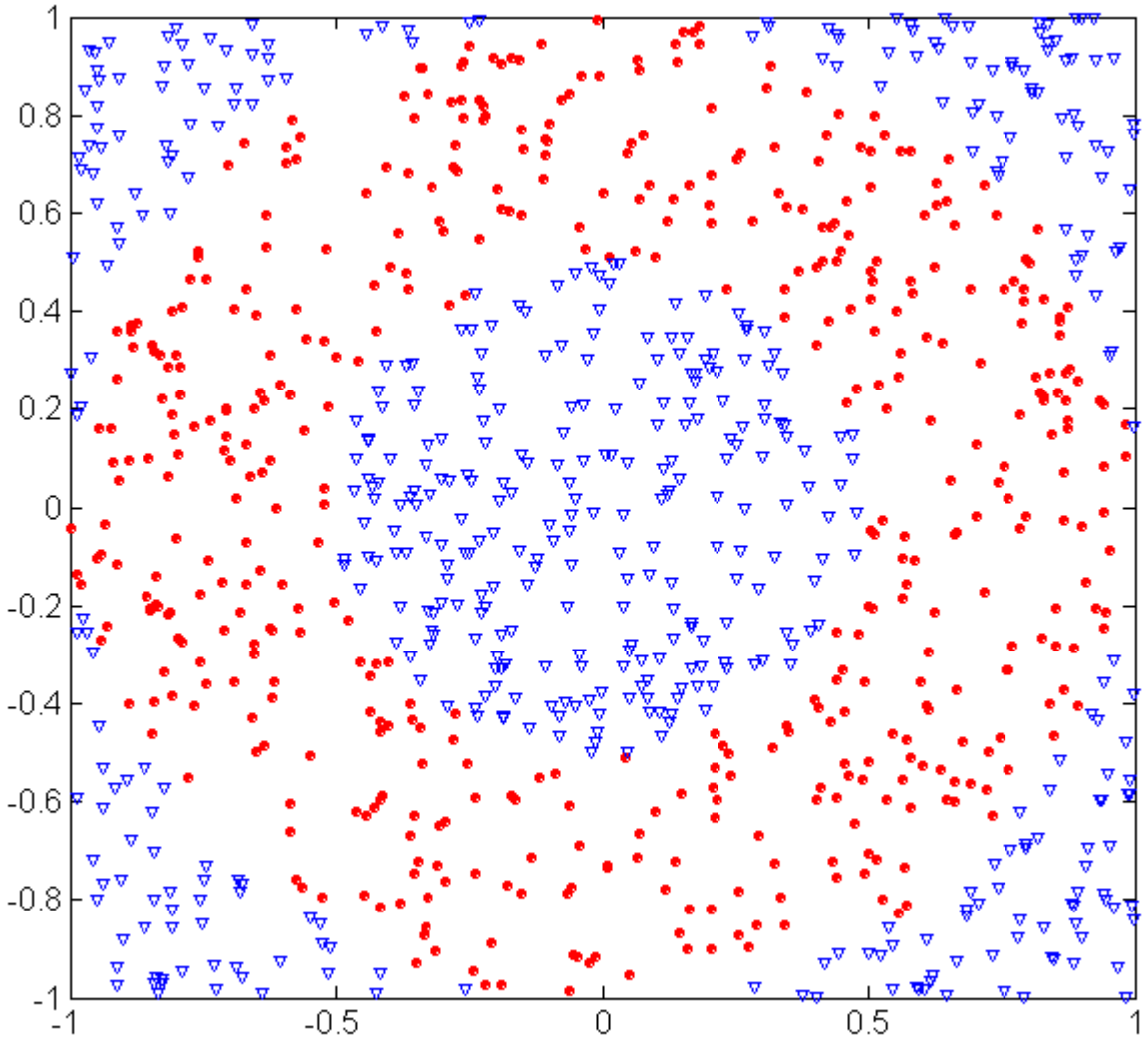


Test on multiple attributes

If **x+y < 1**
then
red class

Not all datasets can be partitioned optimally using test conditions using single attributes!

Oblique Decision Trees



500 circular and 500 triangular data points.

Circular points:

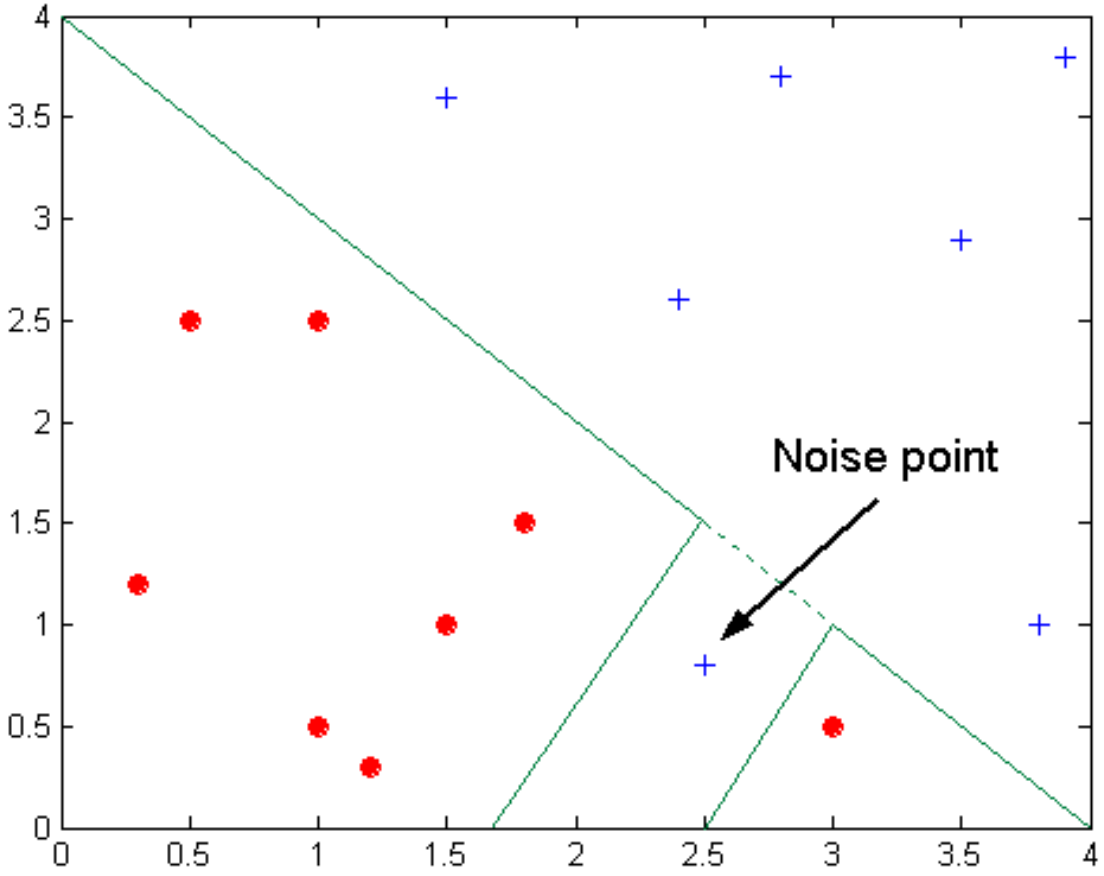
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 1$ or

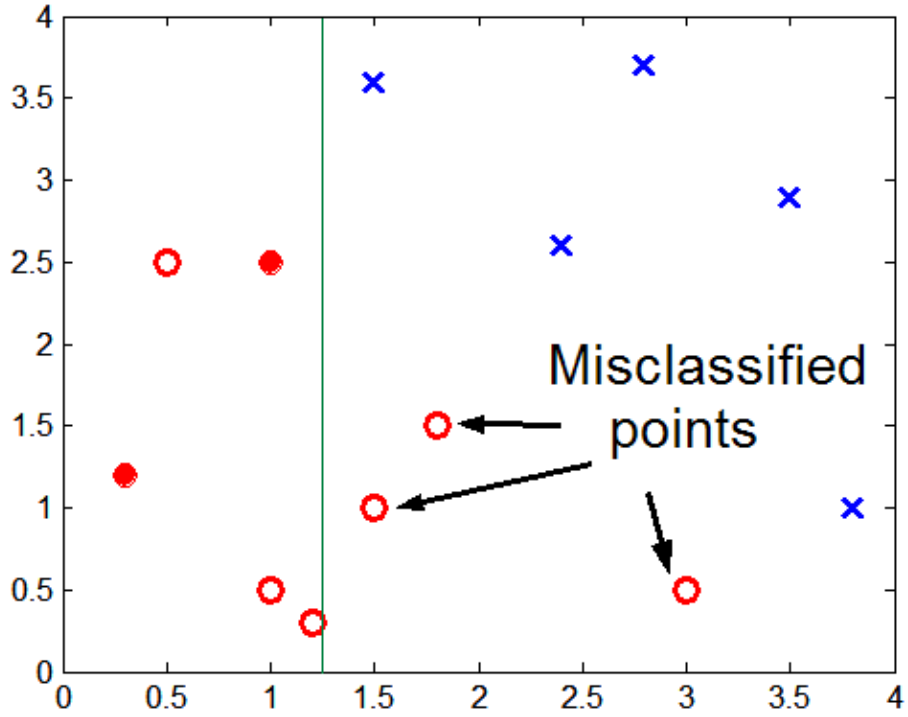
$\text{sqrt}(x_1^2+x_2^2) < 0.5$

Overfitting due to noise



Decision boundary is distorted by noise point

Overfitting due to insufficient samples



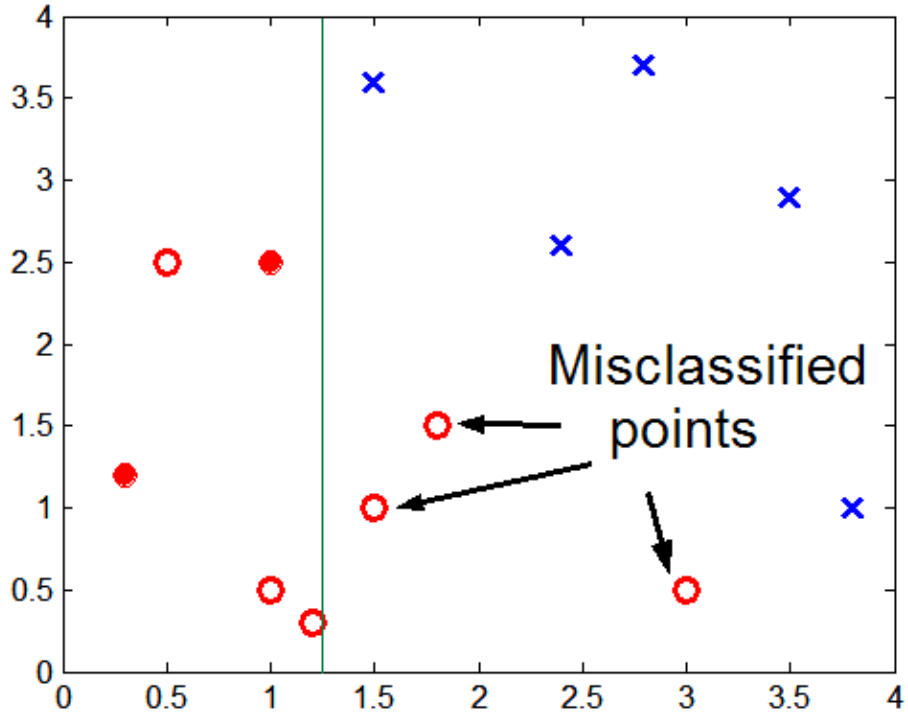
x : class 1

● : class 2

o: test samples

Why?

Overfitting due to insufficient samples



x : class 1

● : class 2

o: test samples

Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting

Overfitting and Tree Pruning

- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early
    - do not split a node if this would result in the goodness measure falling below a threshold
    - ***Difficult to choose an appropriate threshold***
  - Postpruning: Remove branches from a "fully grown" tree
    - get a sequence of progressively pruned trees

Pros and Cons of decision trees

- **Pros**
  - **+ Reasonable training time**
  - **+ Fast application**
  - **+ Easy to interpret**
  - **+ Easy to implement**
  - **+ Can handle large number of features**
- **Cons**
  - **– Cannot handle complicated relationships between features**
  - **– simple decision boundaries**
  - **– problems with lots of missing data**
  - **– NP-complete**

Some well-known decision tree learning implementations

CART Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and Regression Trees. Wadsworth

ID3 Quinlan JR (1986) Induction of decision trees. Machine Learning 1:81–106

C4.5 Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann

J48 Implementation of C4.5 in WEKA

Handling missing values

- Remove attributes with missing values
- Remove examples with missing values
- Assume most frequent value
- Assume most frequent value given a class
- Learn the distribution of a given attribute
- Induce relationships between the available attribute values and the missing feature

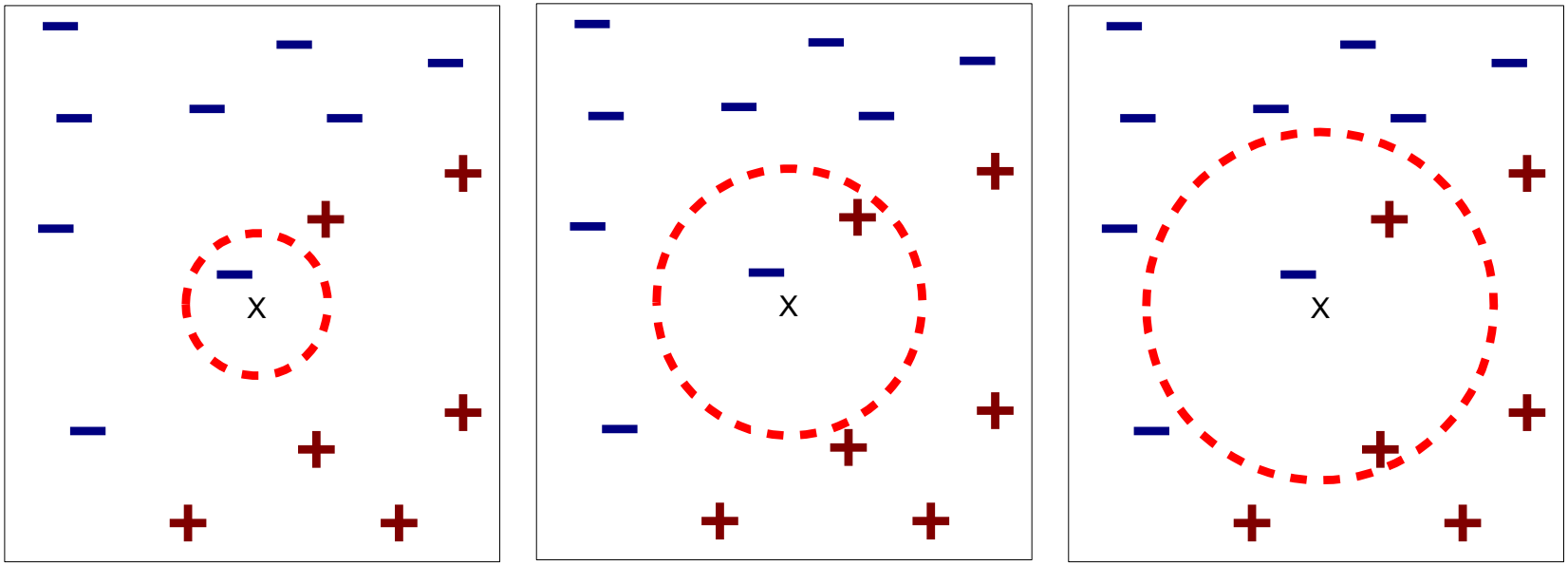
Imputing Missing Values

- Expectation Minimization (EM):
  - Build model of data values (ignore missing vals)
  - Use model to estimate missing values
  - Build new model of data values (including estimated values from previous step)
  - Use new model to re-estimate missing values
  - Re-estimate model
  - Repeat until convergence

Potential Problems

- Imputed values may be inappropriate:
  - in medical databases, if missing values not imputed separately for male and female patients, may end up with male patients with 1.3 prior pregnancies, and female patients suffering from prostate infection
  - many of these situations will not be so obvious
- If some attributes are difficult to predict, filled-in values may be random (or worse)

k-nearest neighbor classifiers



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

k-nearest neighbors of a record **x** are data points that have the **k** smallest distance to **x**

k-nearest neighbor classification

- Given a data record **x** find its **k** closest points
  - Closeness: ?

- Determine the class of **x** based on the classes in the neighbor list
  - Majority vote
  - Weigh the vote according to distance
    - e.g., weight factor, $w = 1/d^2$

Characteristics of NN classifiers

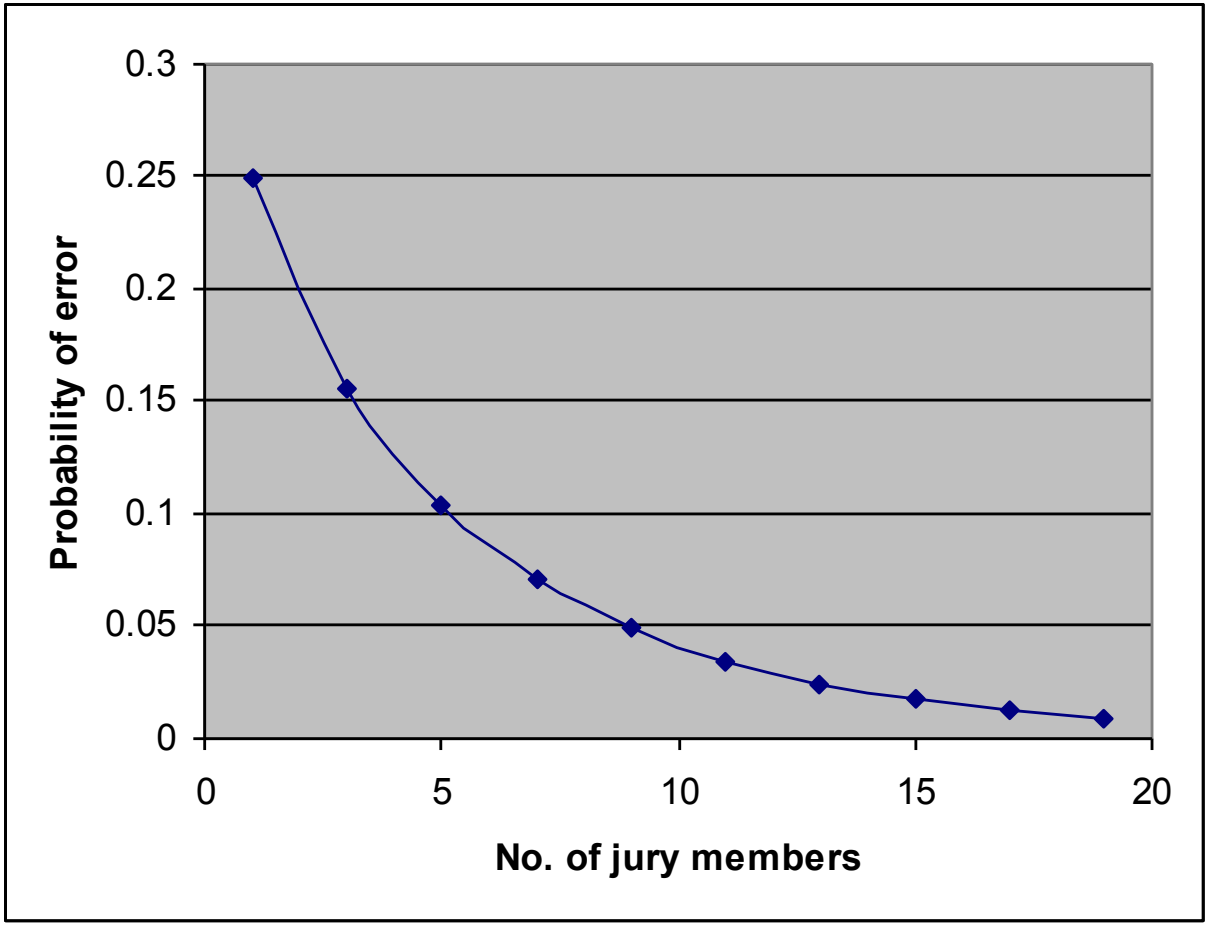
- No model building (lazy learners)
  - Lazy learners: computational time in classification
  - Eager learners: computational time in model building
- Decision trees try to find global models, k-NN take into account local information
- K-NN classifiers depend a lot on the choice of proximity measure

Condorcet's jury theorem

- *If each member of a jury is more likely to be right than wrong,*
- *then the majority of the jury, too, is more likely to be right than wrong*
- *and the probability that the right outcome is supported by a majority of the jury is a (swiftly) increasing function of the size of the jury,*
- *converging to 1 as the size of the jury tends to infinity*

Condorcet, 1785

Condorcet's jury theorem



Random forests

Random forests (Breiman 2001) are generated by combining two techniques:

- bagging (Breiman 1996)
- the random subspace method (Ho 1998)

L. Breiman. Random forests. Machine Learning, 45(1):5–32, 2001

L. Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996

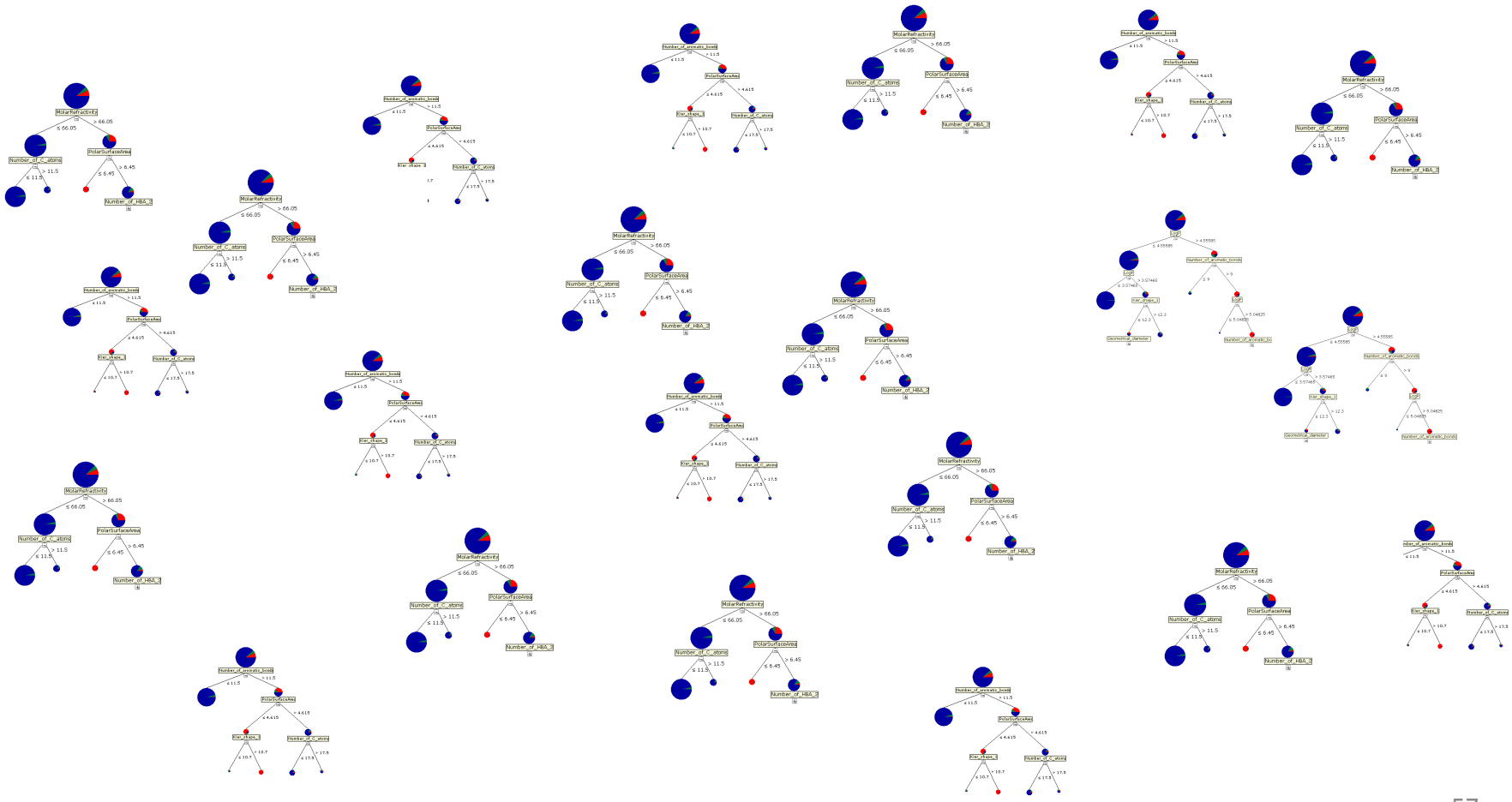
T. K. Ho. The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8): 832-844, 1998

Bagging

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	some	yes

A *bootstrap replicate* E' of a set of examples E is created by randomly selecting $n = |E|$ examples from E with replacement.

Forests



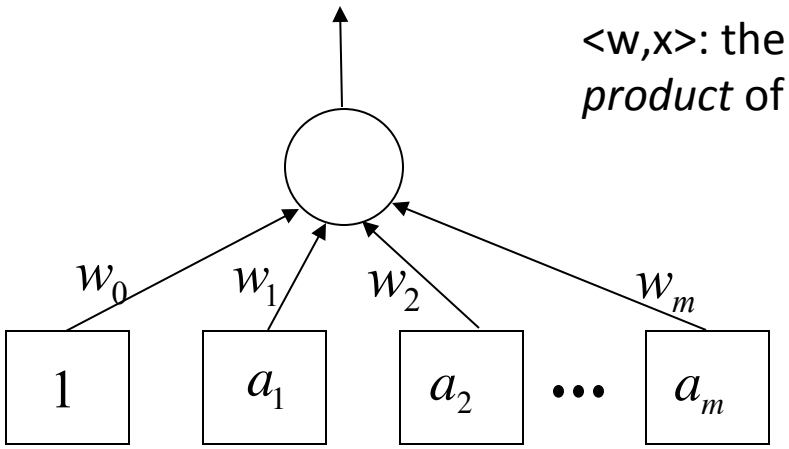
The perceptron

Input: each example $\boldsymbol{x}_i$ has a set of attributes $\boldsymbol{x}_i = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and is of class y_i

Estimated classification output: u_i

Task: express each sample $\boldsymbol{x}_i$ as a weighted combination (linear combination) of the attributes

$$f(x) = \langle w, x \rangle + w_0 = w_1 a_1 + w_2 a_2 + \ldots + w_m a_m + w_0$$



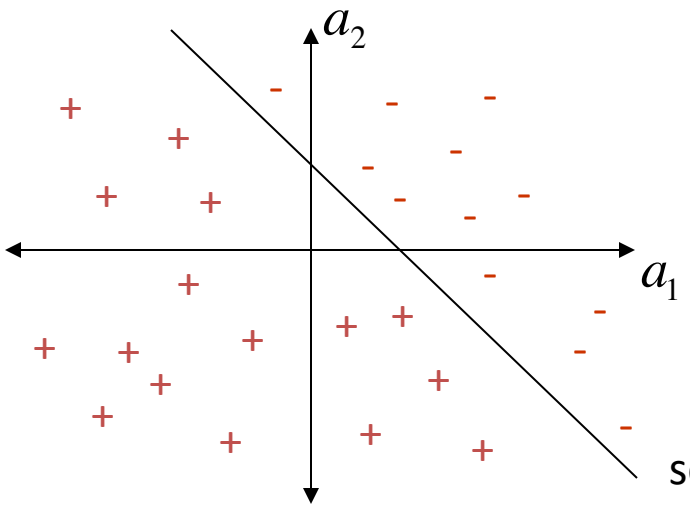
<w,x>: the *inner* or *dot product* of w and x

$$f(x) > 0 \rightarrow u_i = 1$$

$$f(x) \leq 0 \rightarrow u_i = -1$$

The perceptron

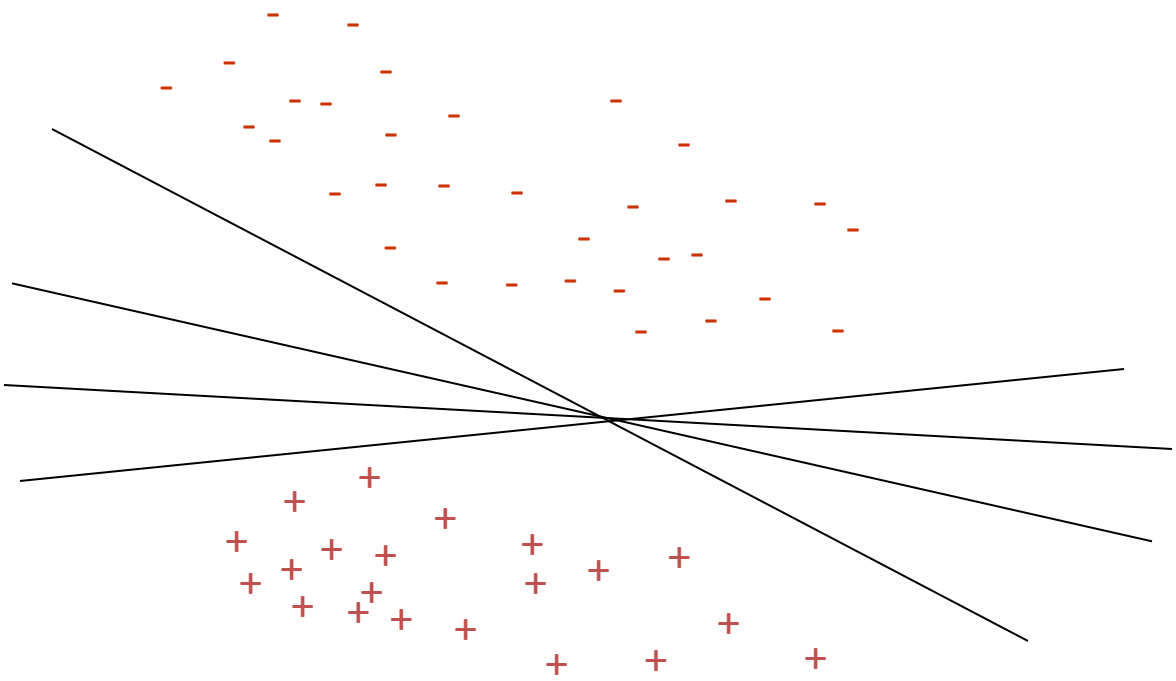
$$w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_m a_m = 0$$



The perceptron learning algorithm is guaranteed to find a separating hyperplane – if there is one.

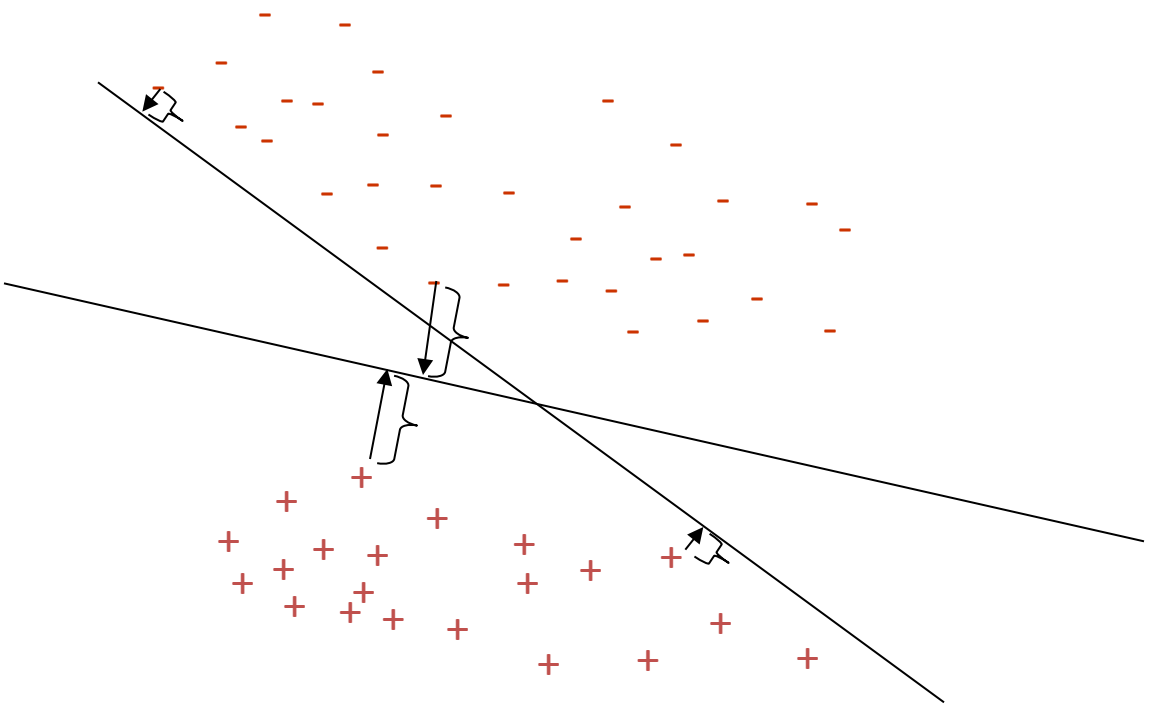
separating hyperplane

Many possible separating hyperplanes



Which hyperplane to choose?

Choosing a separating hyperplane



margin: defined by the two points (one from the '+' set and the other from the '–' set) with the minimum distance to the separating hyperplane

The maximum margin hyperplane

- There are many hyperplanes that might classify the data
- One reasonable choice:
  - The hyperplane that represents the largest separation, or ***margin***, between the two classes
  - We choose the hyperplane so that the distance from it to the nearest data point on each side is maximized
  - If such a hyperplane exists, it is known as the **maximum-margin hyperplane**

The maximum margin hyperplane

- It is desirable to design linear classifiers that maximize the margins of their decision boundaries
- This ensures that their worst-case generalization errors are minimized

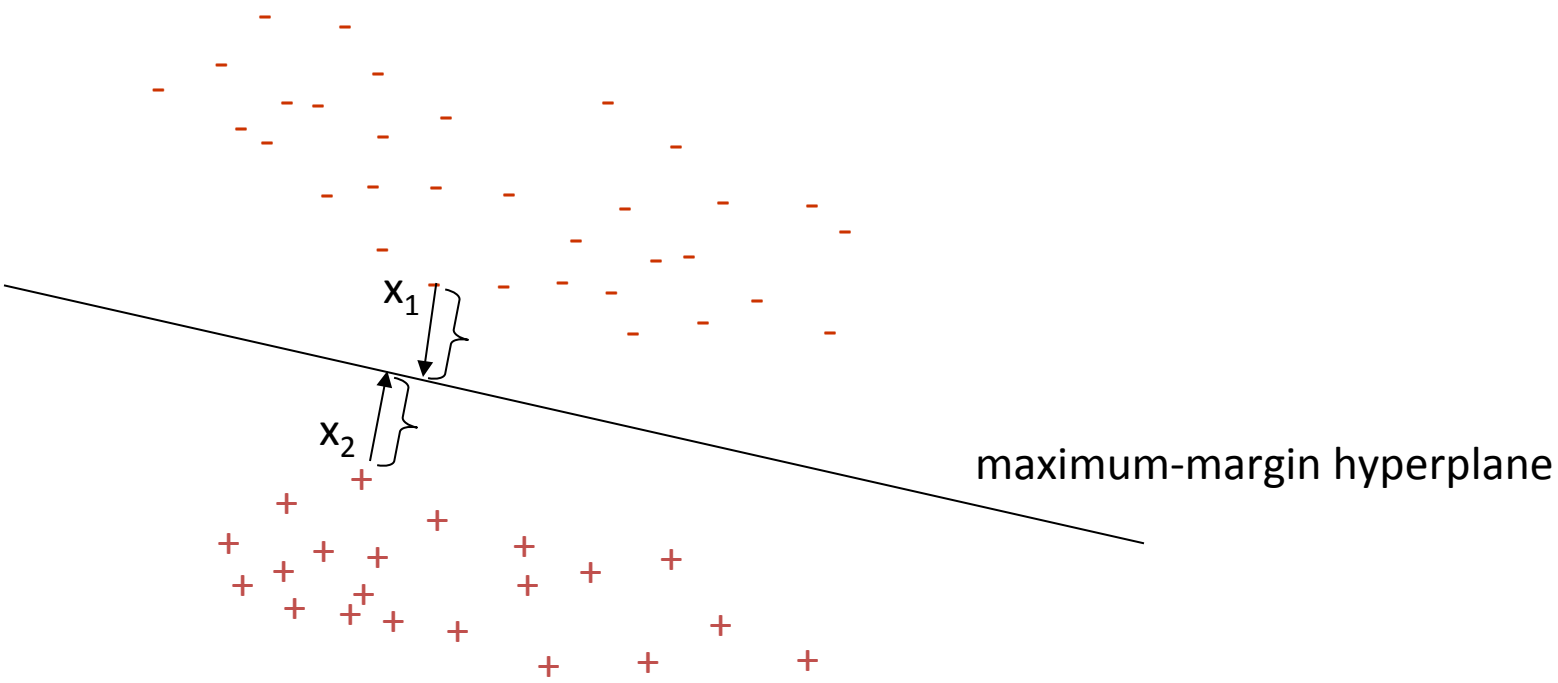
One example of such classifiers:

- **Linear support vector machines:**
  - Find the maximum-margin hyperplane
  - Express this hyperplane as a linear combination of the data points (x_i, y_i):

$$f(x) = \sum_i k_i y_i \langle x_i, x \rangle + b$$

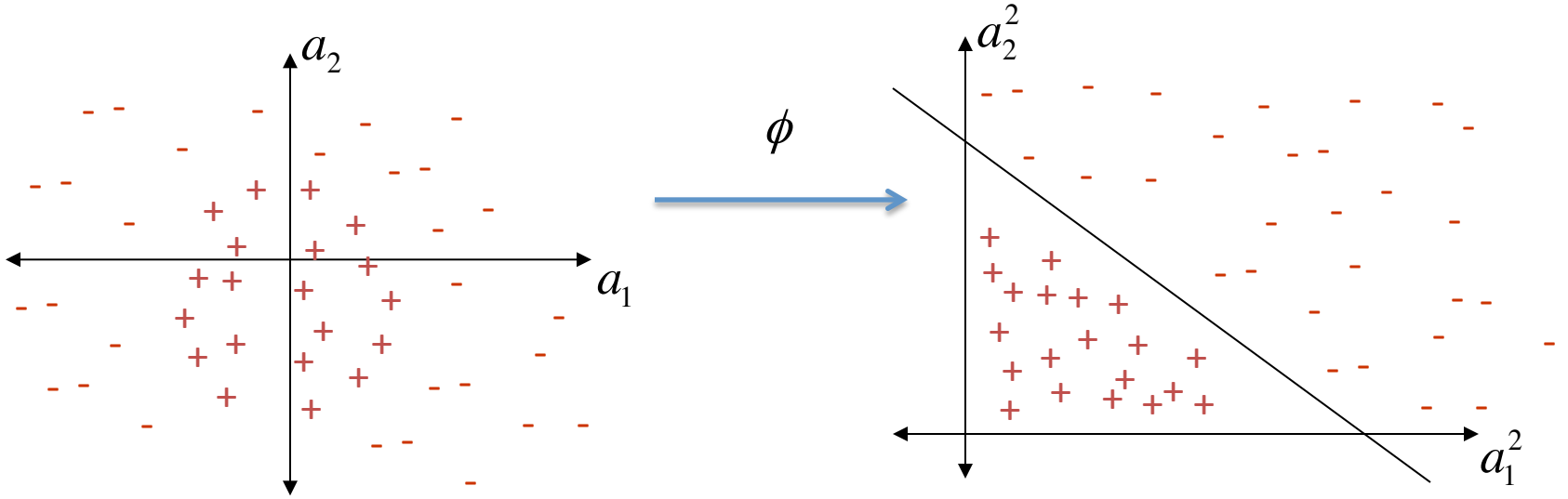
  - The two points (one from each side) with the smallest distance from the hyperplane are called ***support vectors***

Linear SVM



x_1 and x_2 are the support vectors

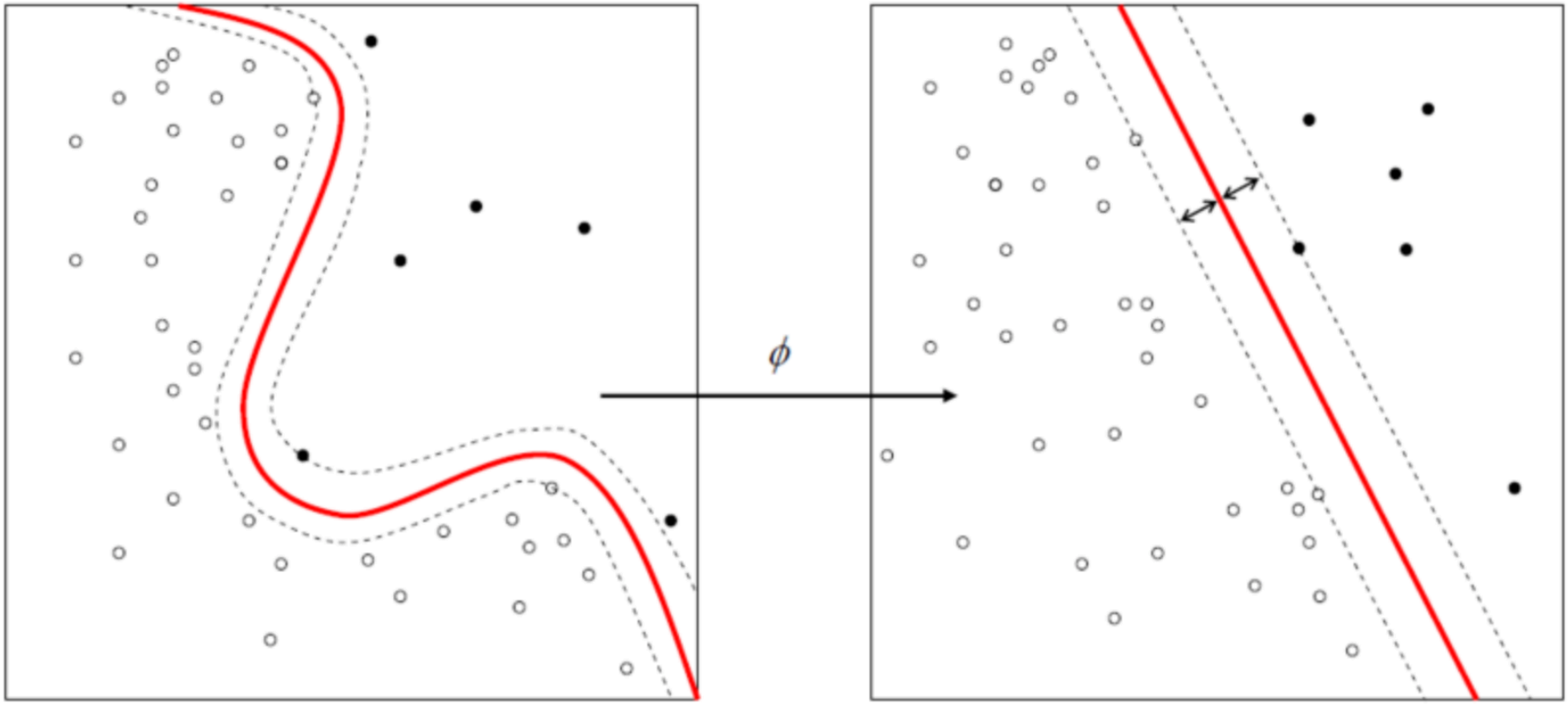
What if the classes are not linearly separable?



$$w_0 + w_1 a_1 a_2 + w_2 a_1^2 + w_3 a_2^2 = 0$$

Support Vector Machines

Support vector machines



Define a *mapping φ(x)* that maps each attribute of point x to a new space where points are *linearly separable*

Recap

- What is classification
- Overview of classification methods
- Decision trees
- Forests
- Support Vector Machines (SVMs)