

Classification & Regression

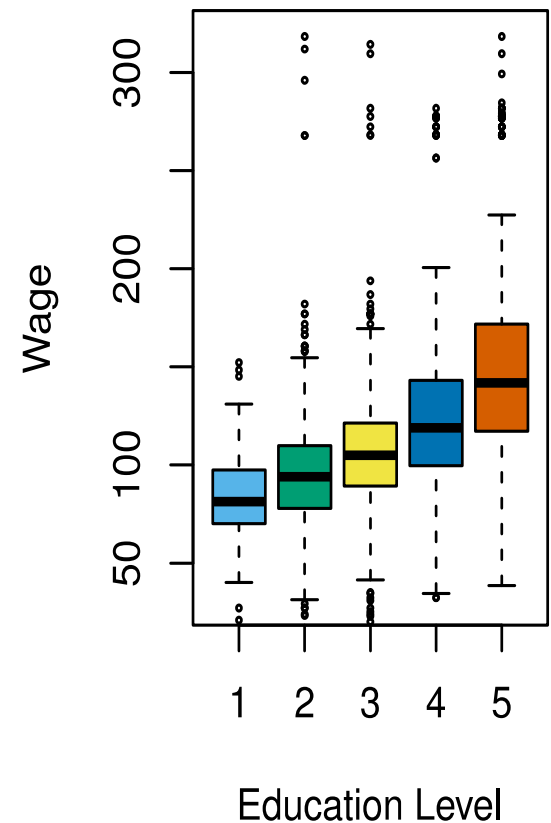
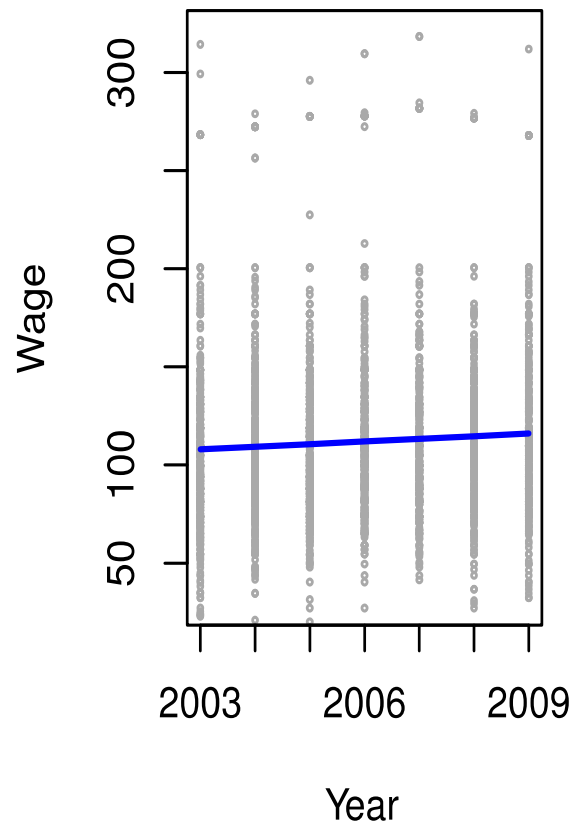
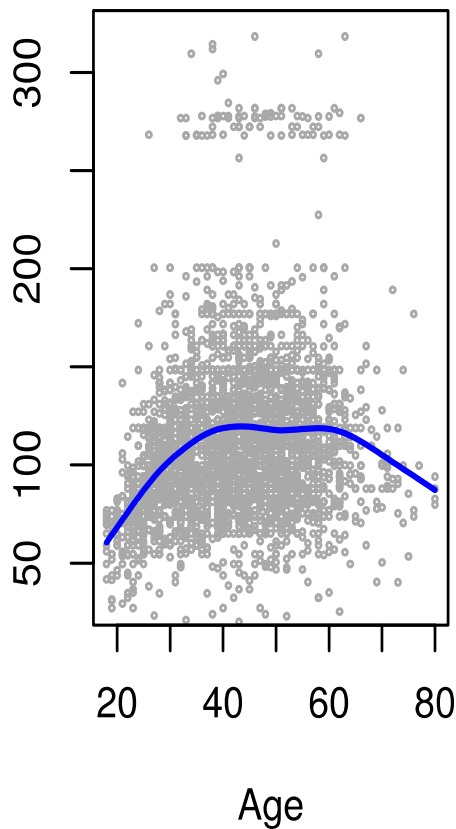
Dr. David C. Anastasiu
San José State University

Outline

- Basic concepts
- Decision trees
- Instance-based learning
- Bayesian classifiers
- Other classifiers:
 - Rule-based
 - Artificial neural networks
 - Support vector machines.
- Ensemble methods

BASIC CONCEPTS

Application 1: Wage Data



Application 1 (cont'd)

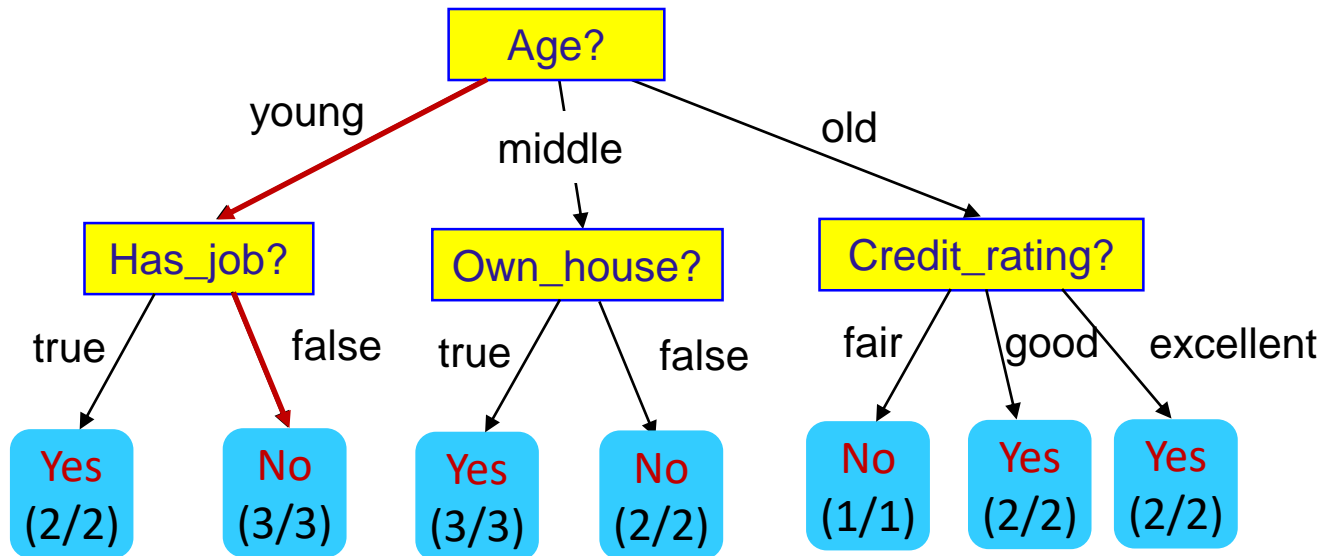
- Objective #1: Given input (age, educational level, etc.), try to predict the output (wage).
 - Regression: Prediction of a quantitative (continuous) variable
- Objective #2: Find possible dependencies between the different variables (e.g. age & wage)
 - Which particular predictors actually affect the response?
 - Is the relationship positive or negative?
 - Is the relationship a simple linear one or is it more complicated etc.?
- Inference

Application 2: Loan application data

ID	Age	Has_job	Own_house	Credit_rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Application 2

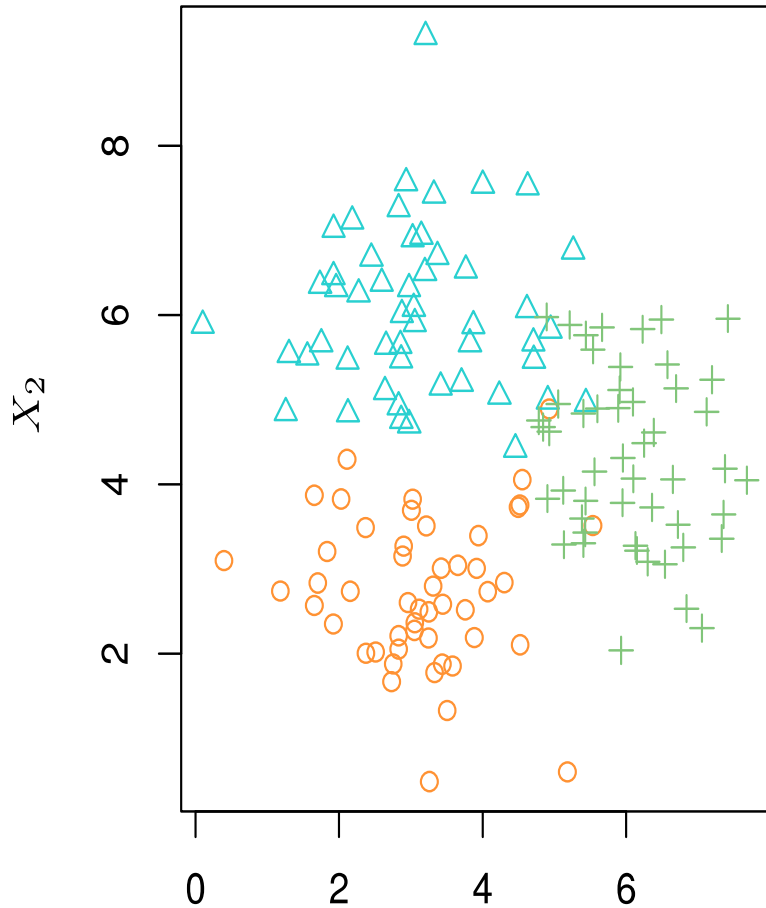
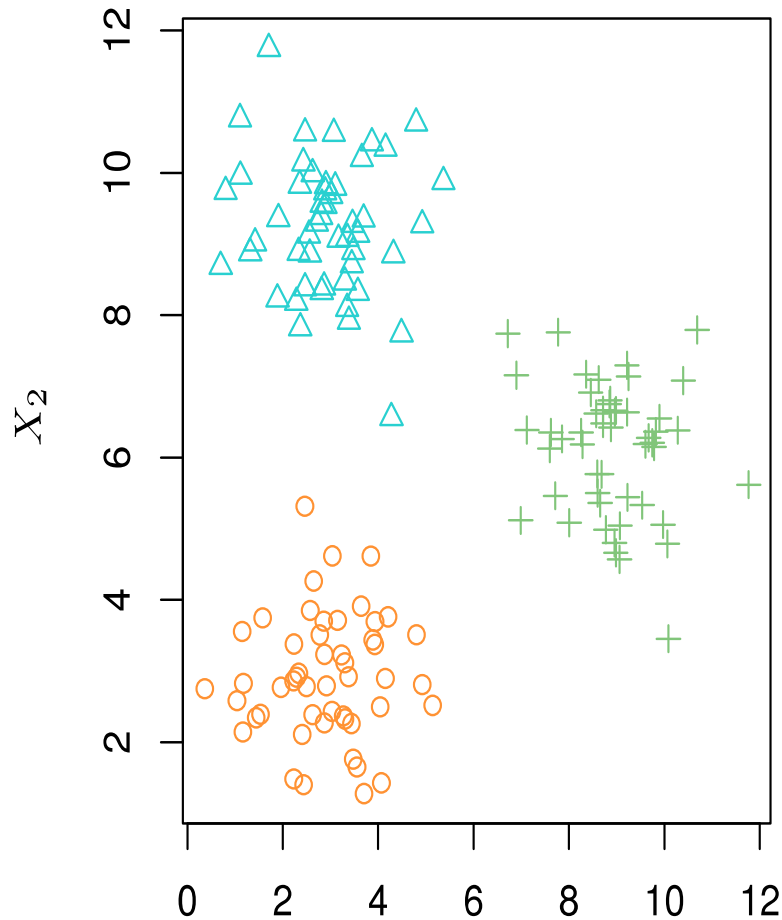
Age	Has_job	Own_house	Credit_rating	Class
young	false	false	good	?No



Application 2 (cont'd)

- Objective #1: Given input (demographics, etc.) predict the output (loan approved?)
 - **Classification: Prediction** of a **categorical (qualitative)** variable.
- Objective #2: Find dependencies between variables (e.g. age and home ownership)
 - **Inference**

Application 3: Loan applicants



Application 3 (cont'd)

- Objective: Given the information on loan applicants, put them into groups
 - No output variable.
 - **Clustering** (Descriptive method)

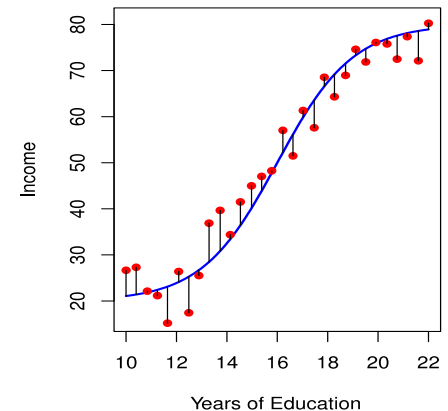
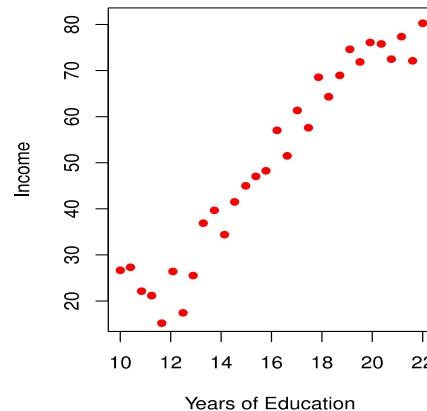
Machine learning

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- We call this the training set.
- It consists of pairs (X, Y) (training examples)

The data and the goal

- **Data:** A set of data records (also called examples, instances, or cases) described by
 - k attributes (predictors): X_1, X_2, \dots, X_k .
 - a response Y
- **Goal:** To learn the relationship between X and Y :

$$Y = f(X) + \varepsilon$$



Supervised vs. Unsupervised Learning

- **Supervised learning:** Learn model from examples (observations).
 - Both predictors (X) and response (Y) are observed.
- **Unsupervised learning**
 - There is no response variable or it has not been observed
 - Find inherent patterns in the data.
 - Later in this class....

Supervised Learning

- Prediction

- Based on a new value X predict the outcome Y (approximate by $f(X)$)
 - Function should be generic enough to cover unseen samples

- Inference

- Study the relationship(s) between Y and X s.
 - Function should approximate training data as closely as possible

Regression vs. Classification

- Supervised learning
- Depends on the type of Y
- If Y is continuous/numerical:
 - Regression problem
 - E.g. predict someone's wage based on age, education etc.
- If Y is categorical:
 - Classification problem
 - E.g. predict whether a loan application will be approved or not.
 - Predictor Y is called the **class**
 - In this class we will focus on Classification

Classification: Definition

- We are given a collection of records (training set)
 - Each record is characterized by a tuple (x, y) , where x is a set of attributes and y is the class label
 - x : set of attributes, predictors, independent variables, inputs.
 - y : class, response, dependent variable, or output.
- Task:
 - Learn a model that maps each set of attributes x into one of the predefined class labels y .

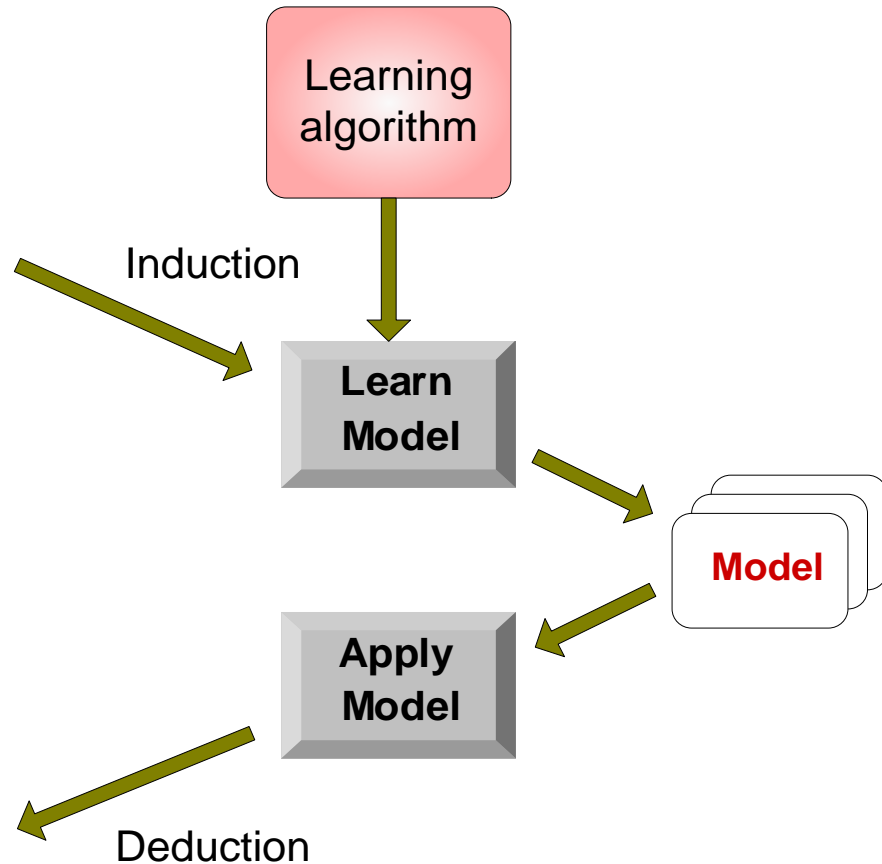
Building and using a classification model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

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

Test Set



Classification techniques

- Base classifiers
 - Decision tree-based methods.
 - Rule-based methods.
 - Nearest-neighbor.
 - Neural networks.
 - Naïve Bayes and Bayesian belief networks.
 - Support vector machines.
 - ...and others.
- Ensemble classifiers
 - Boosting, bagging, random forests, etc.

Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.



What if the accuracy is not satisfactory?

- Go back and...
 - Do some further processing on the data (pre-processing), or
 - Choose another algorithm
- Many iterations might be needed
- It is also possible that we are unable to build a satisfactory model due to:
 - Randomness in the data
 - Limitations of current algorithms

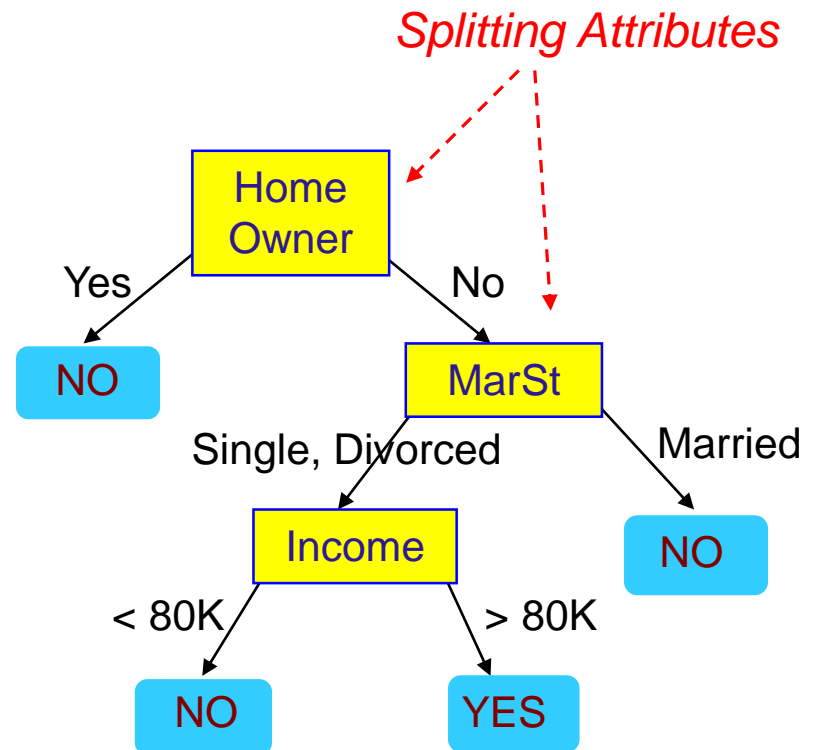
We will use this method to illustrate various concepts and issues associated with the classification task.

DECISION TREES

Example of a decision tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

categorical
categorical
continuous
class



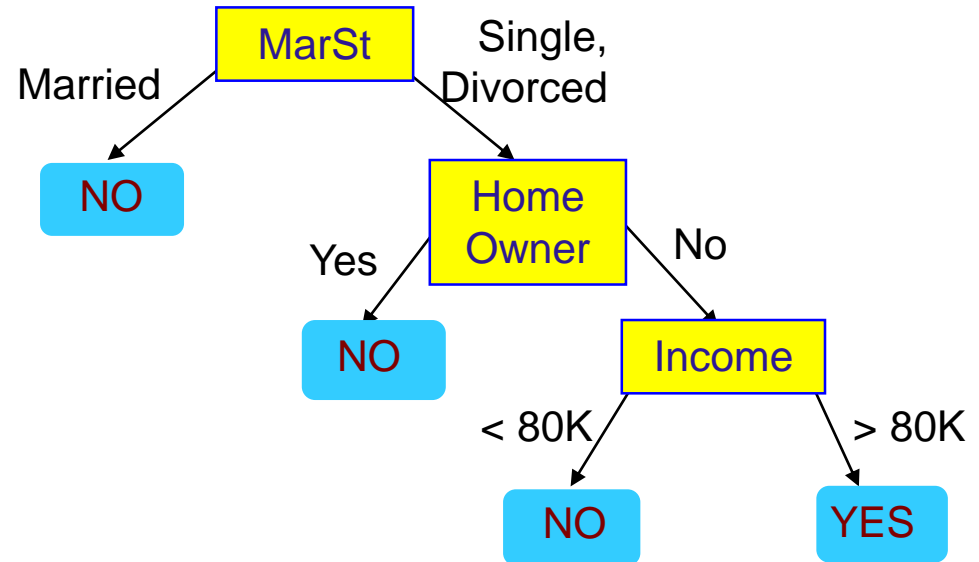
Training Data

Model: Decision tree

Example of decision tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

categorical
categorical
continuous
class



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

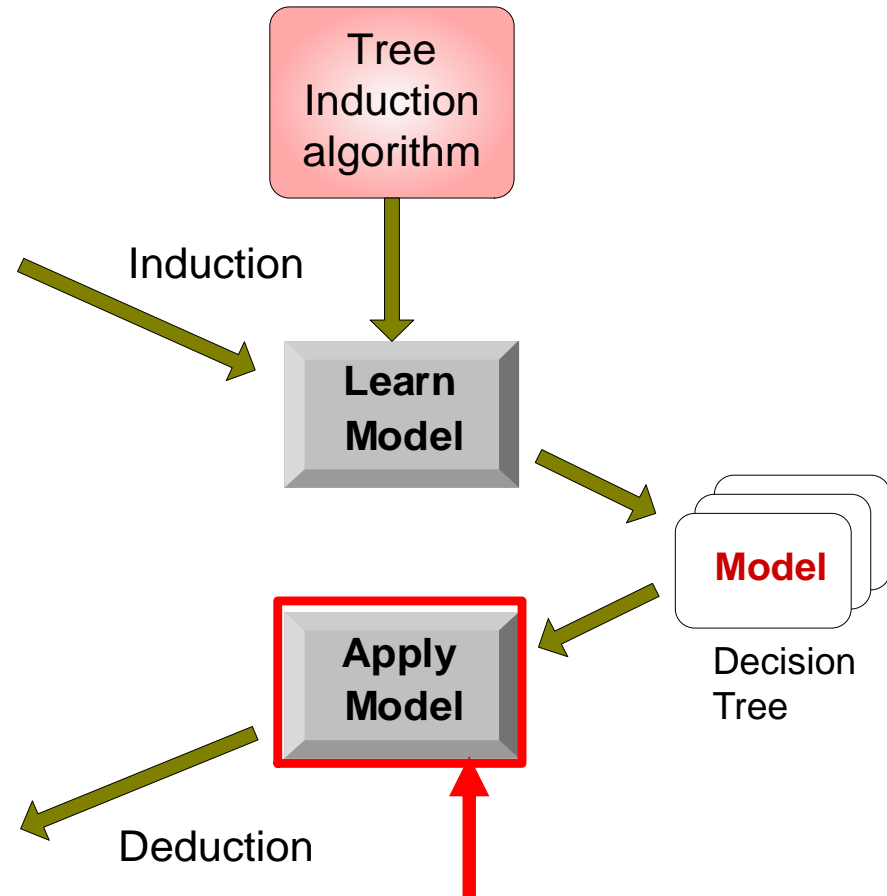
Training Data

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

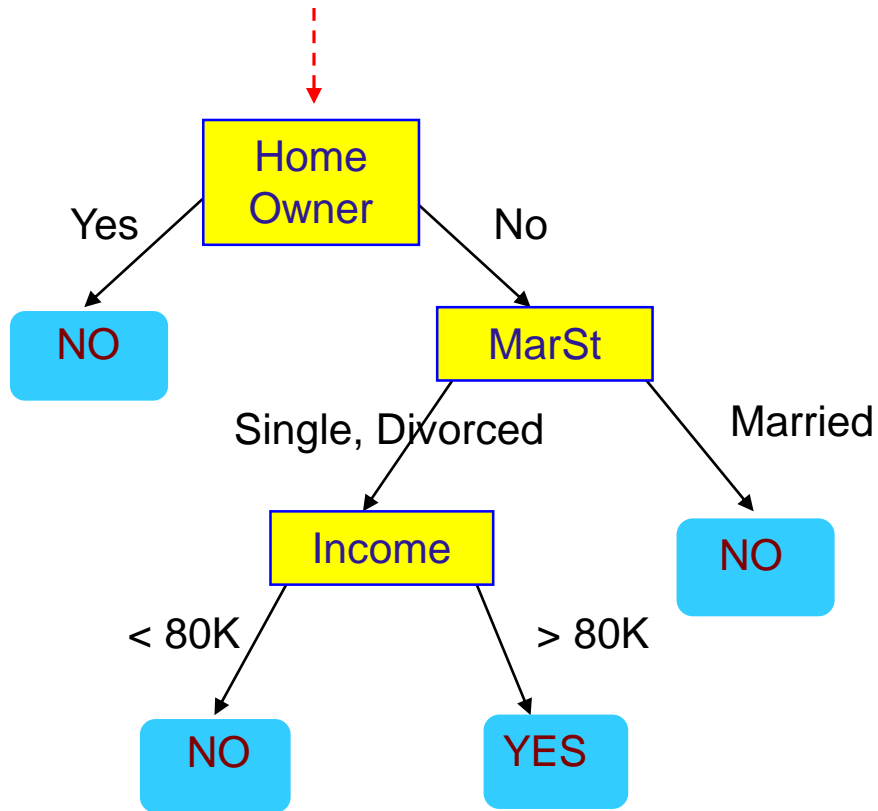
<i>Tid</i>	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply model to test data

Start from the root of tree.



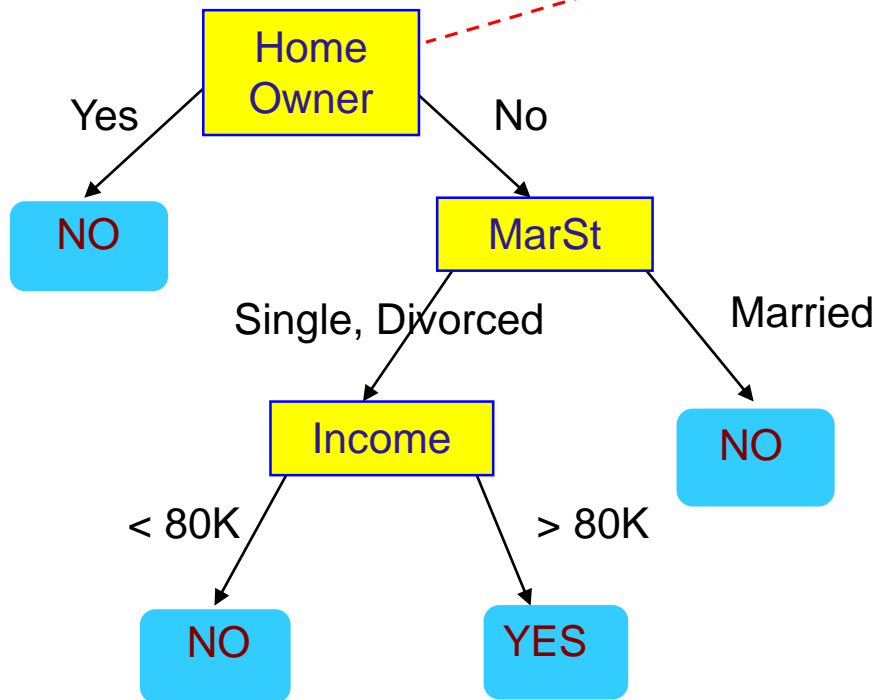
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Apply model to test data

Test Data

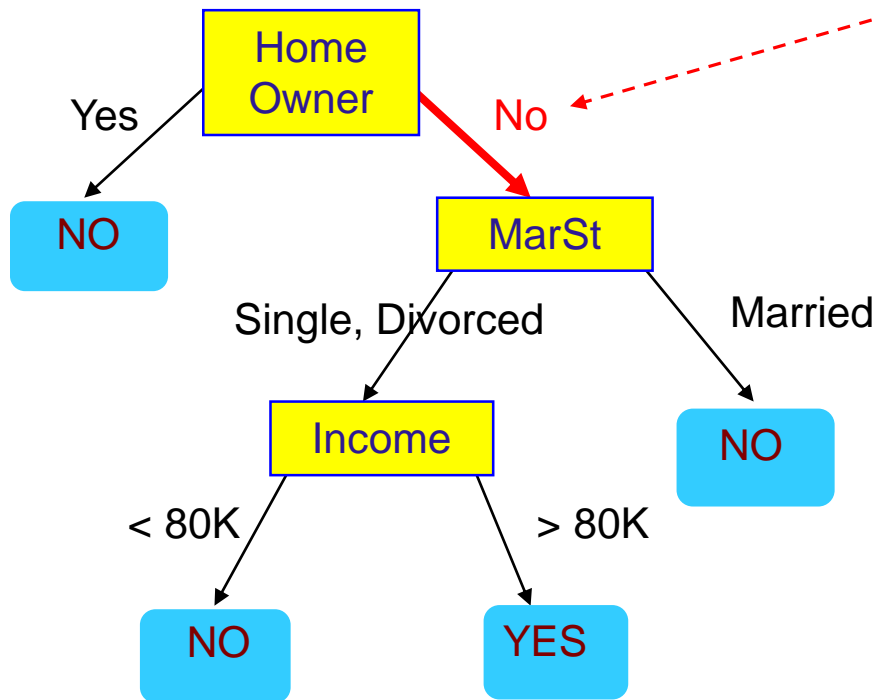
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply model to test data

Test Data

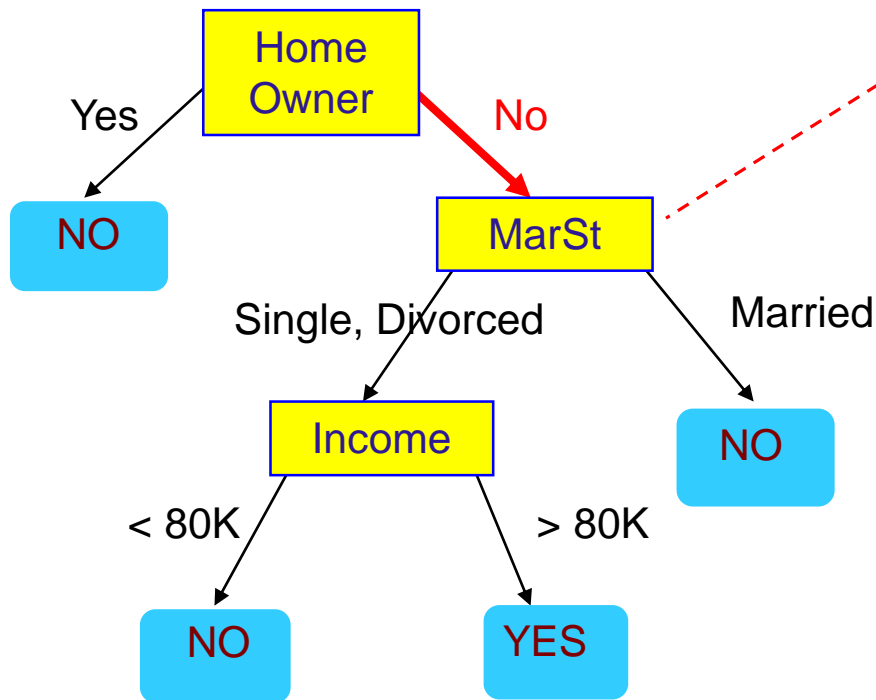
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply model to test data

Test Data

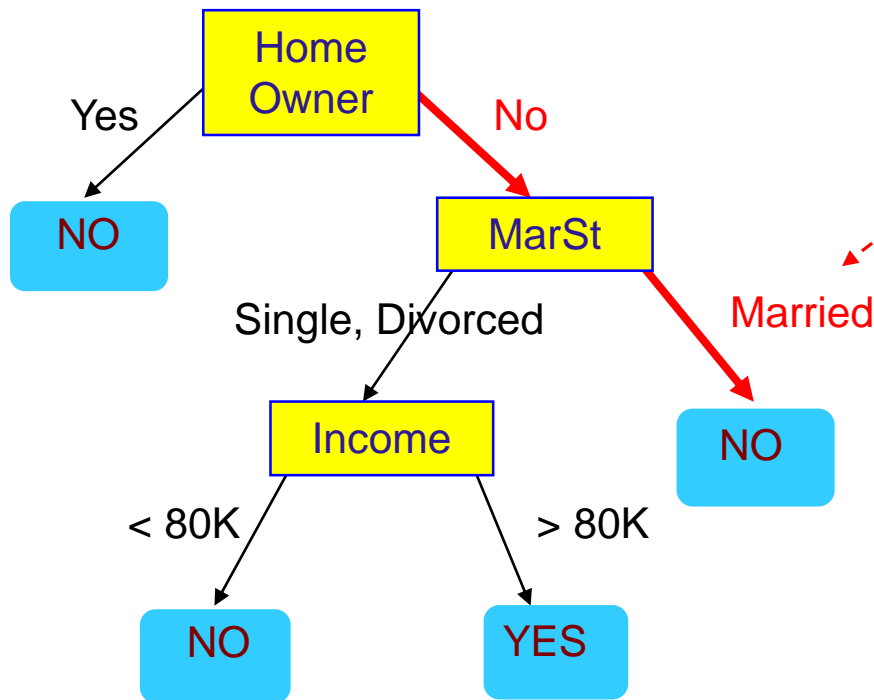
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply model to test data

Test Data

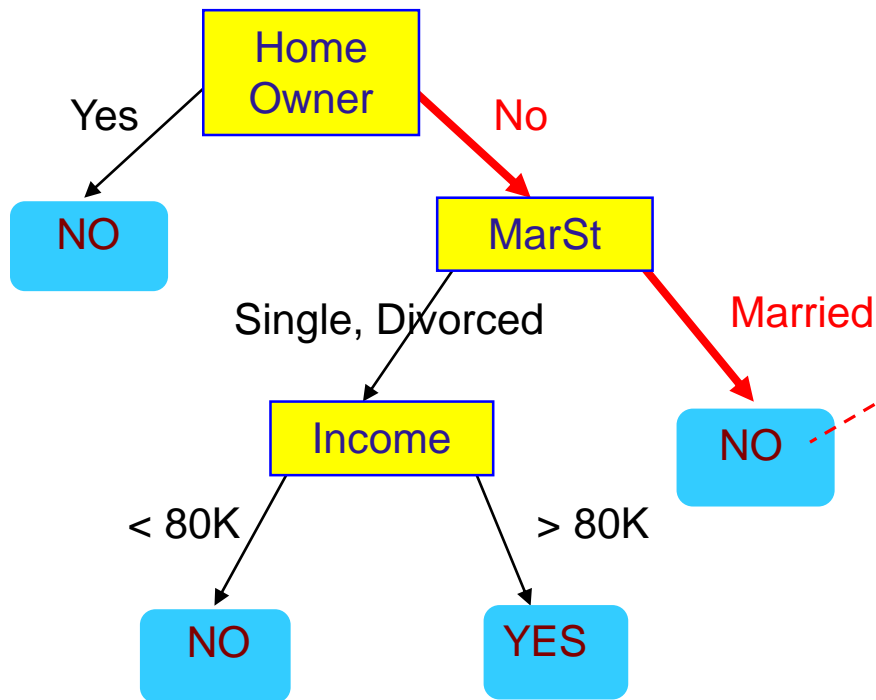
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply model to test data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Assign Defaulted to
"No"

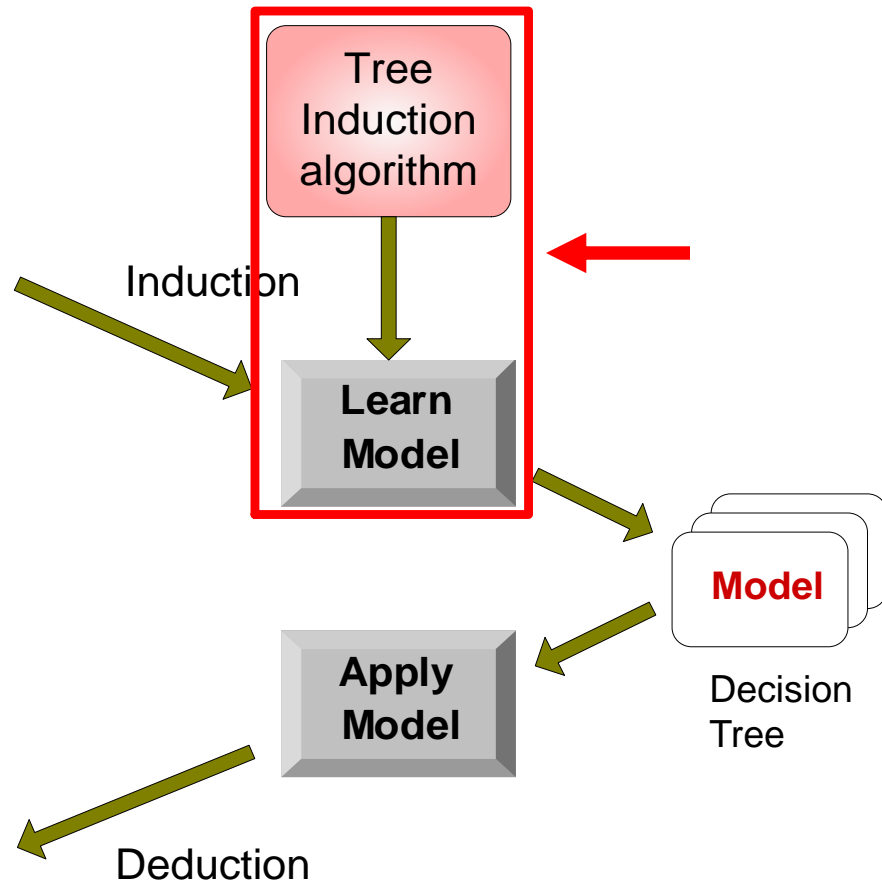
Decision tree classification task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

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

Test Set



Decision tree induction

- Many algorithms:
 - Hunt's Algorithm (one of the earliest).
 - CART.
 - ID3, C4.5, and C5.0.
 - SLIQ and SPRINT.

Design issues of decision tree induction

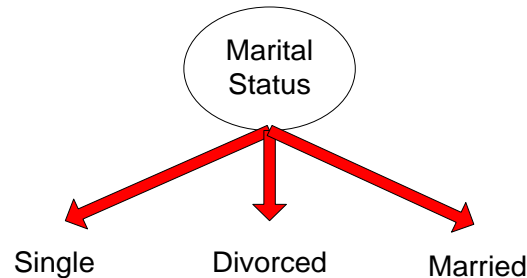
- How should training records be split?
 - Method for specifying test condition.
 - This depends on the attribute types.
 - Method for selecting which attribute and split condition to choose.
 - Need a measure for evaluating the goodness of a test condition.
- When should the splitting procedure stop?
 - Stop splitting if all the records belong to the same class or have identical attribute values.
 - Early termination.

Methods for expressing test conditions

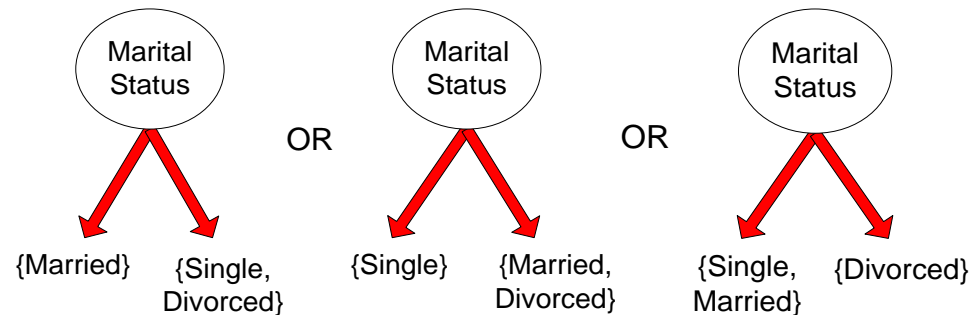
- Depends on attribute types:
 - Binary
 - Nominal
 - Ordinal
 - Continuous
- Depends on number of ways to split:
 - 2-way split
 - Multi-way split

Test condition for nominal attributes

Multi-way split: Use as many partitions as distinct values:

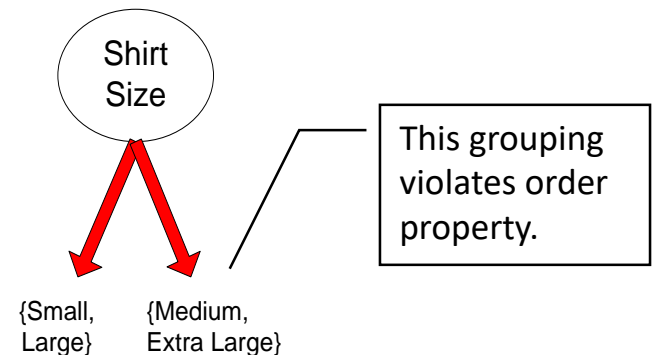
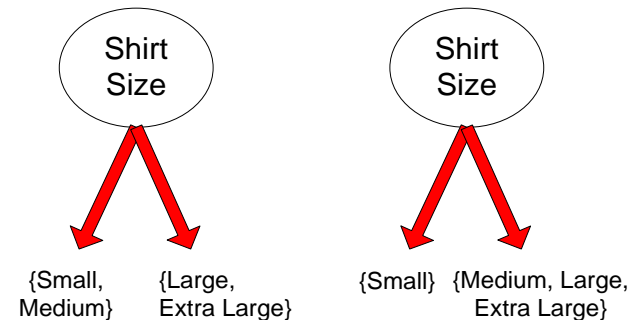
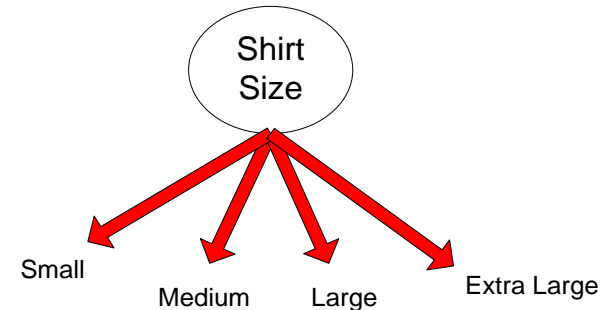


Binary split: Divide values into two subsets:

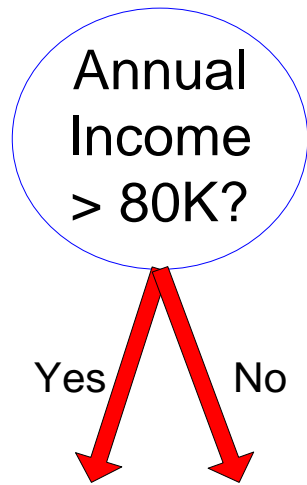


Test condition for ordinal attributes

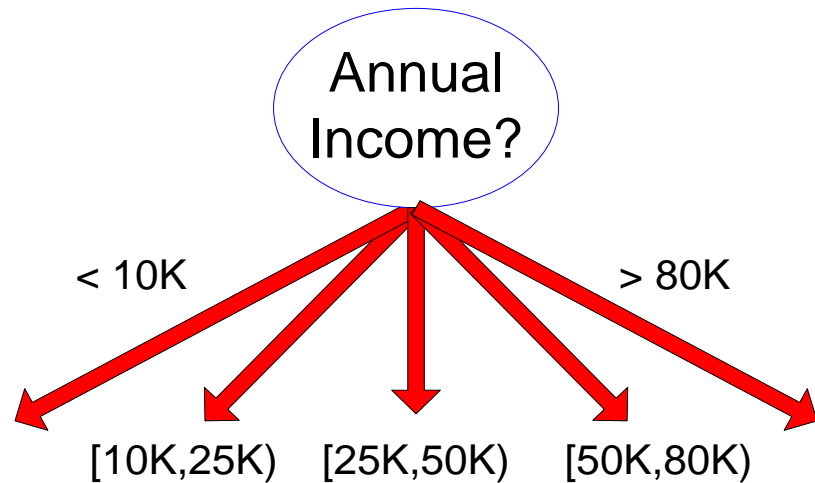
- Multi-way split:
 - Use as many partitions as distinct values.
- Binary split:
 - Divides values into two subsets.
 - Preserve order property among attribute values.



Test condition for continuous attributes



(i) Binary split



(ii) Multi-way split

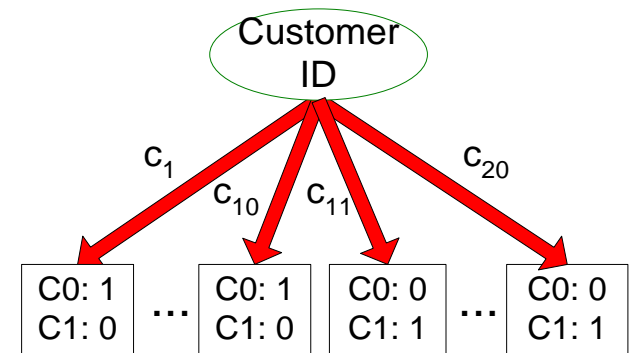
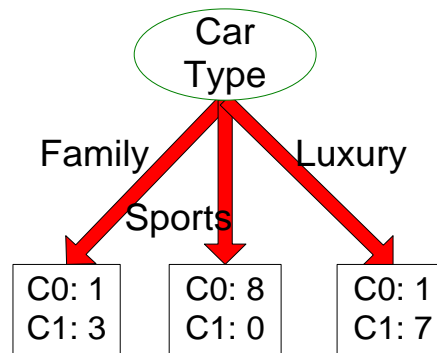
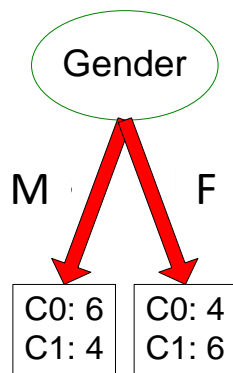
Splitting based on continuous attributes

- Different ways of handling such attributes:
 - **Discretization** to form an ordinal categorical attribute.
 - Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
 - Static – discretize once at the beginning.
 - Dynamic – repeat at each node.
 - **Binary Decision**: $(A < v)$ or $(A \geq v)$
 - Consider all possible splits and find the best splitting point.
 - Can be more compute intensive.

How to determine the best split?

Before splitting:
10 records of class 0, and
10 records of class 1.

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1



Which test condition is the best?

How to determine the best split?

- Greedy approach:
 - Nodes with **pur**er class distribution are preferred.
- Need a measure of node purity/impurity:

C0: 5
C1: 5

High degree of impurity

C0: 9
C1: 1

Low degree of impurity

Finding the best split

1. Compute impurity measure (P) before splitting.
2. Compute impurity measure (M) after splitting.
 - Compute impurity measure of each child node.
 - M is the size-weighted impurity of the children.
3. Choose the attribute test condition that produces the highest gain:

$$\text{Gain} = P - M,$$

or equivalently, lowest impurity measure after splitting (M).

Finding the best split

Before Splitting:

C0	N00
C1	N01

→ P

A?

Yes

No

Node N1

Node N2

C0	N10
C1	N11

C0	N20
C1	N21

M11

M12

M1

$$M_{split} = \sum_{i=1}^k \frac{n_i}{n} M(i)$$

Gain = P - M1 vs P - M2

B?

Yes

No

Node N3

Node N4

C0	N30
C1	N31

C0	N40
C1	N41

M21

M22

M2

Measures of node impurity

- Gini Index

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

- Entropy

$$Entropy(t) = -\sum_j p(j | t) \log p(j | t)$$

- Misclassification error

$$Error(t) = 1 - \max_i p(i | t)$$

Measure of impurity: GINI index

Gini index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

(NOTE: $p(j | t)$ is the relative frequency of class j at node t).

- Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information.
 - Minimum (0.0) when all records belong to one class, implying most interesting information.
-
- Measures population diversity

Measure of impurity: GINI index

Gini index for a given node t:

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

(NOTE: $p(j|t)$ is the relative frequency of class j at node t).

For 2-class problem ($p, 1 - p$):

- $GINI = 1 - p^2 - (1 - p)^2 = 2p(1-p)$

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Computing Gini index of a single node

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

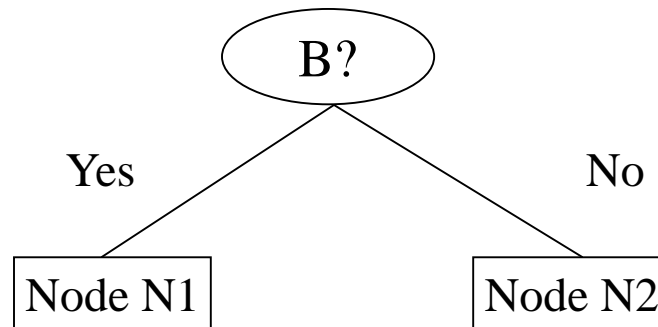
C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

Example: GINI index for a binary split

- Splits into two partitions.
- Effect of size weighing partitions:
 - Larger and Purer Partitions are sought for.



	Parent
C1	7
C2	5
Gini = 0.486	

$$\begin{aligned} \text{Gini}(N1) &= 1 - (5/6)^2 - (1/6)^2 \\ &= 0.278 \end{aligned}$$

$$\begin{aligned} \text{Gini}(N2) &= 1 - (2/6)^2 - (4/6)^2 \\ &= 0.444 \end{aligned}$$

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

$$\begin{aligned} \text{Weighted Gini of N1 N2 Split} &= 6/12 * 0.278 + \\ &\quad 6/12 * 0.444 \\ &= 0.361 \end{aligned}$$

$$\text{Gain} = 0.486 - 0.361 = 0.125$$

Categorical attributes: Computing Gini index

- For each distinct value, gather counts for each class in the dataset.
- Use the count matrix to make decisions.

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

Which of these is the best?

Measure of Impurity: Entropy

- Entropy at a given node t :

$$Entropy(t) = -\sum_j p(j | t) \log p(j | t)$$

(NOTE: $p(j | t)$ is the relative frequency of class j at node t).

- Maximum ($\log n_c$) when records are equally distributed among all classes implying least information.
 - Minimum (0.0) when all records belong to one class, implying most information.
-
- Measures the amount of disorder in a system
 - The lower the entropy, the lower the disorder
 - Entropy based computations are quite similar to the GINI index computations.

Computing entropy of a single node

$$Entropy(t) = -\sum_j p(j | t) \log_2 p(j | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Entropy = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Entropy = - (1/6) \log_2 (1/6) - (5/6) \log_2 (5/6) = 0.65$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Entropy = - (2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$

Computing information gain after splitting

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Parent node, p is split into k partitions;

n_i is number of records in partition i

- Choose the split that achieves most reduction (maximizes GAIN).
- Used in the ID3 and C4.5 decision tree algorithms.

Decision tree based classification

- Advantages:
 - Inexpensive to construct.
 - Extremely fast at classifying unknown records.
 - Easy to interpret for small-sized trees.
 - Robust to noise (especially when methods to avoid overfitting are employed).
 - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting).
- Disadvantages:
 - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
 - Does not take into account interactions between attributes.
 - Each decision boundary involves only a single attribute.

OVERFITTING

Classification errors

- Training errors (apparent errors):
 - Errors committed on the training set.
- Test errors:
 - Errors committed on the test set.
- Generalization errors:
 - Expected error of a model in a randomly selected subset of records from the same distribution.

Example data set

Two class problem:

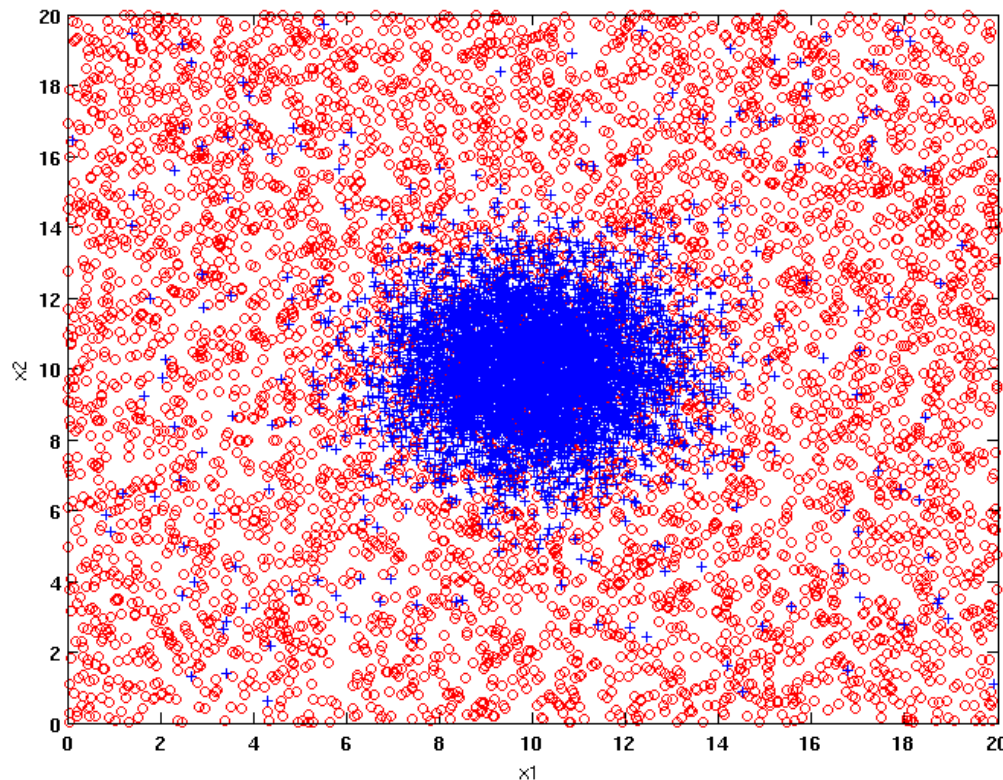
+ : 5400 instances

- 5000 instances generated from a Gaussian centered at (10,10)
- 400 noisy instances added

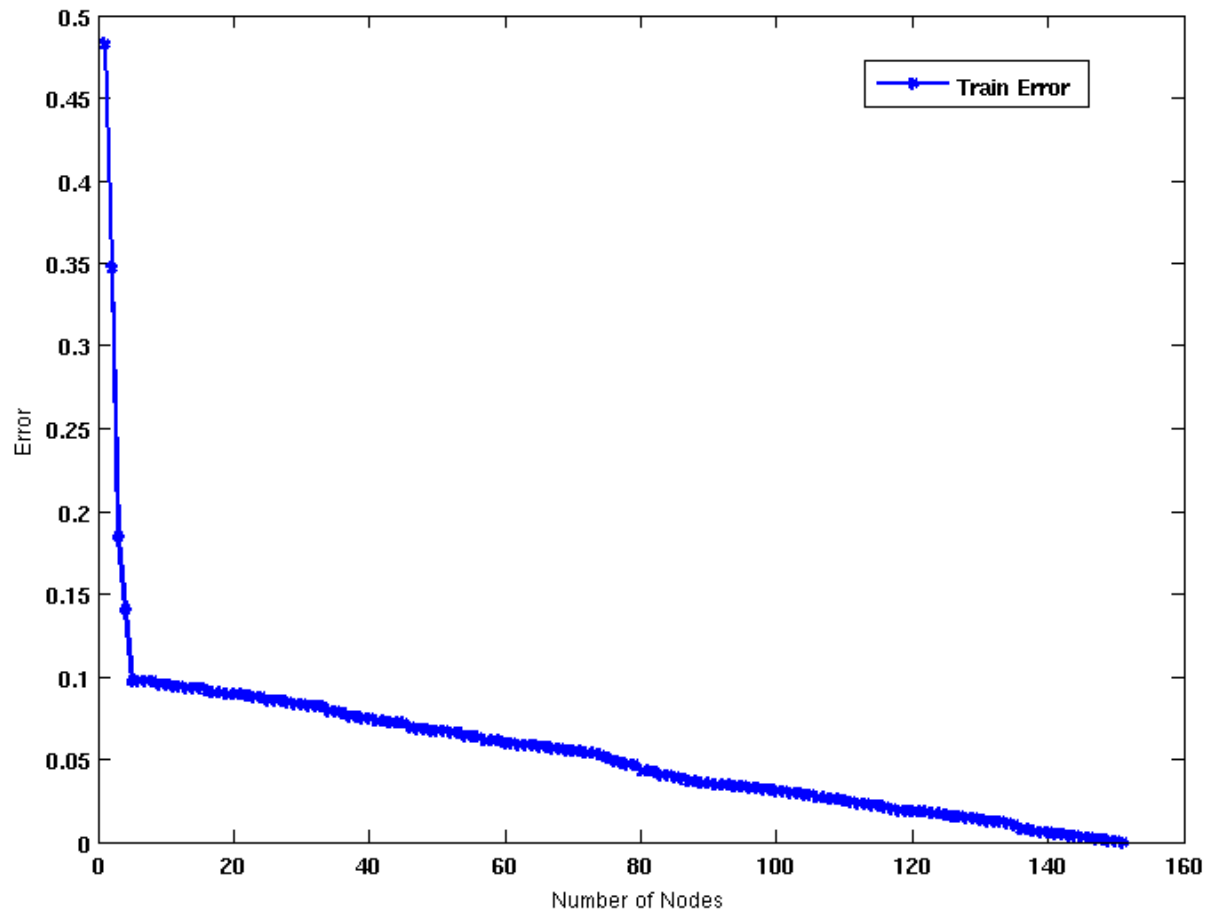
o : 5400 instances

- Generated from a uniform distribution

10 % of the data used for training and 90% of the data used for testing

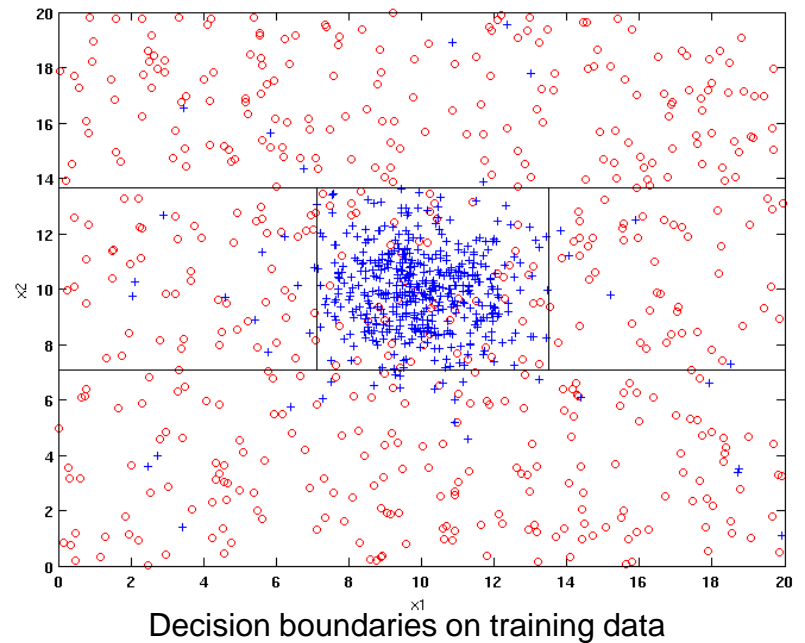
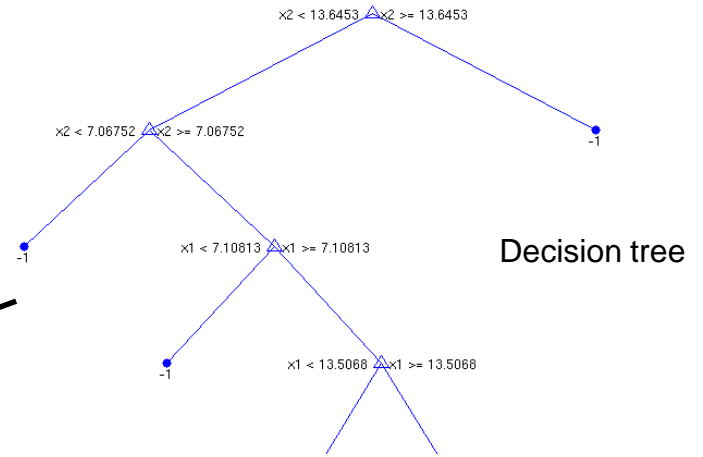
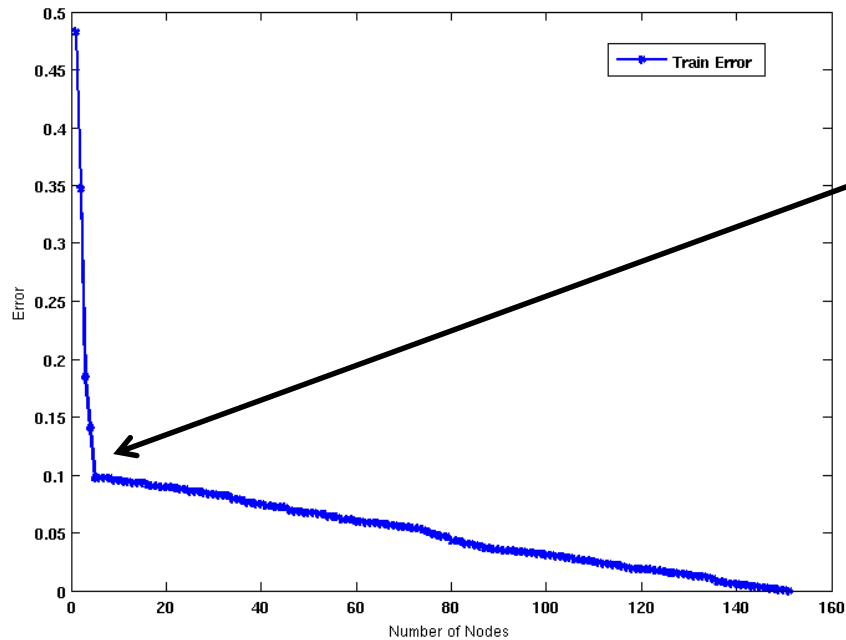


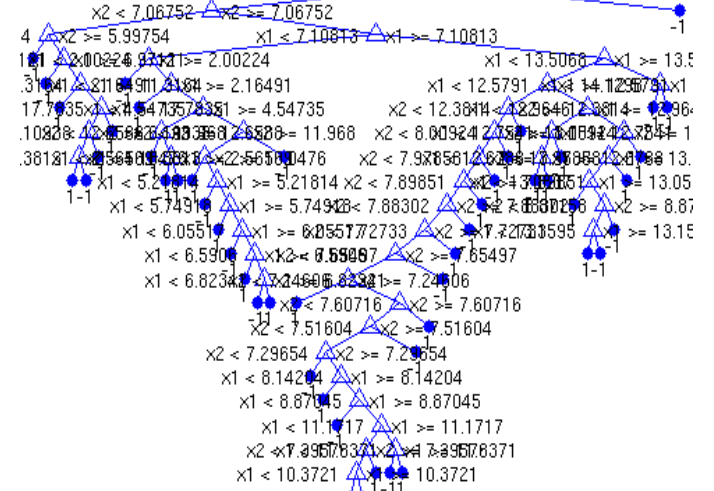
Increasing number of nodes in the decision tree



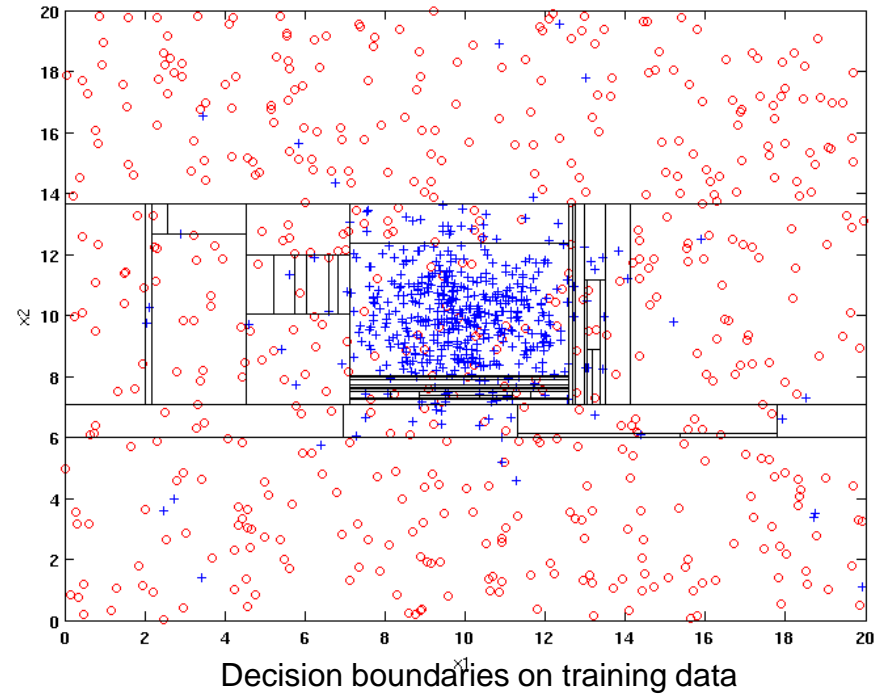
The rate at which a model is able to approximate training data well generally increases with the complexity of the model.

Decision tree with 4 nodes

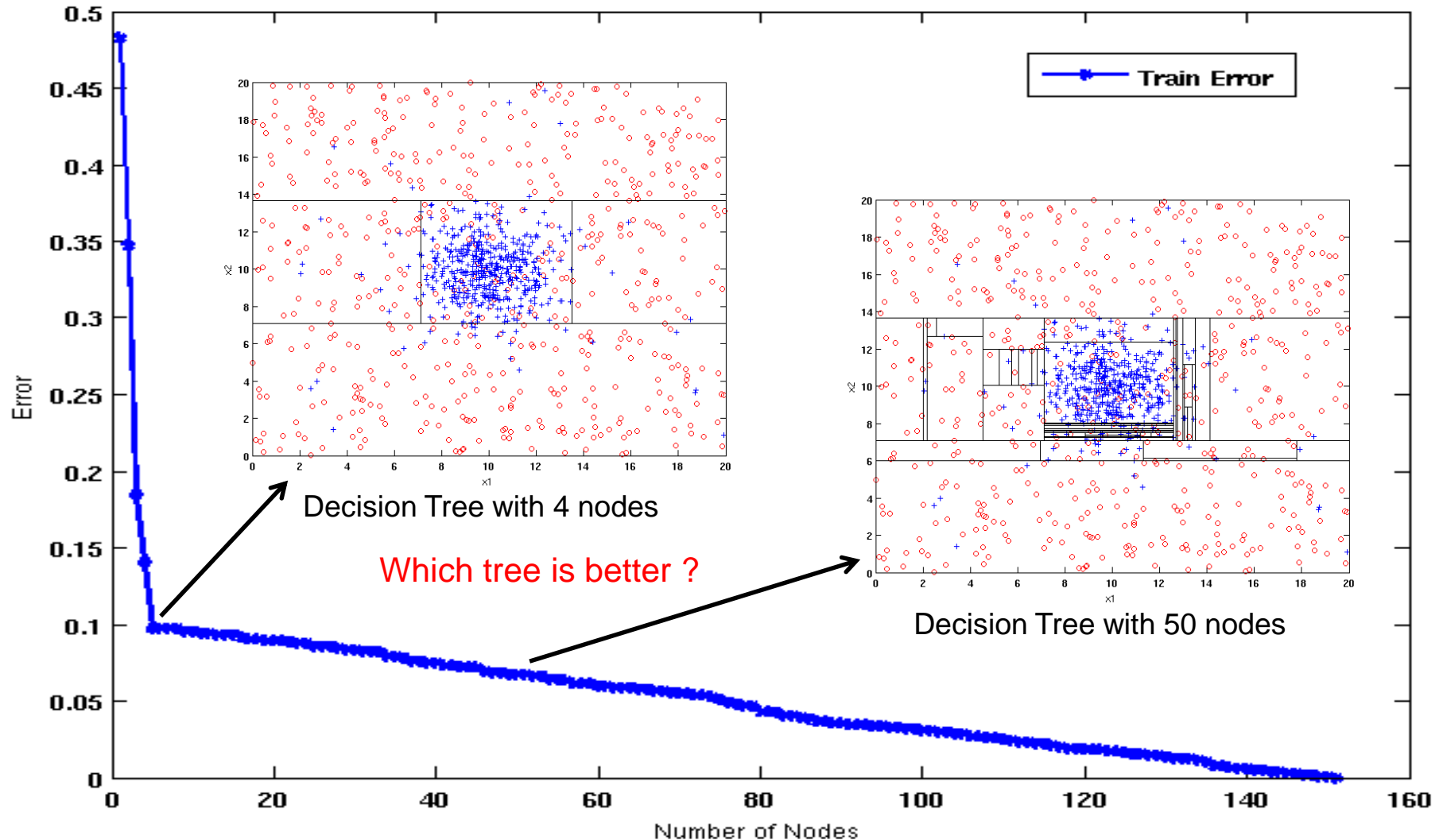




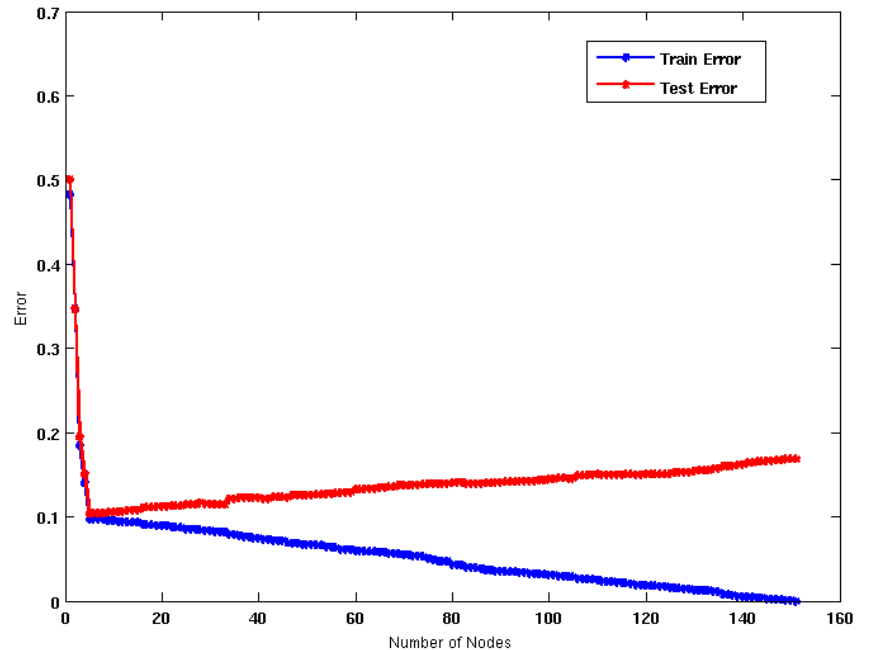
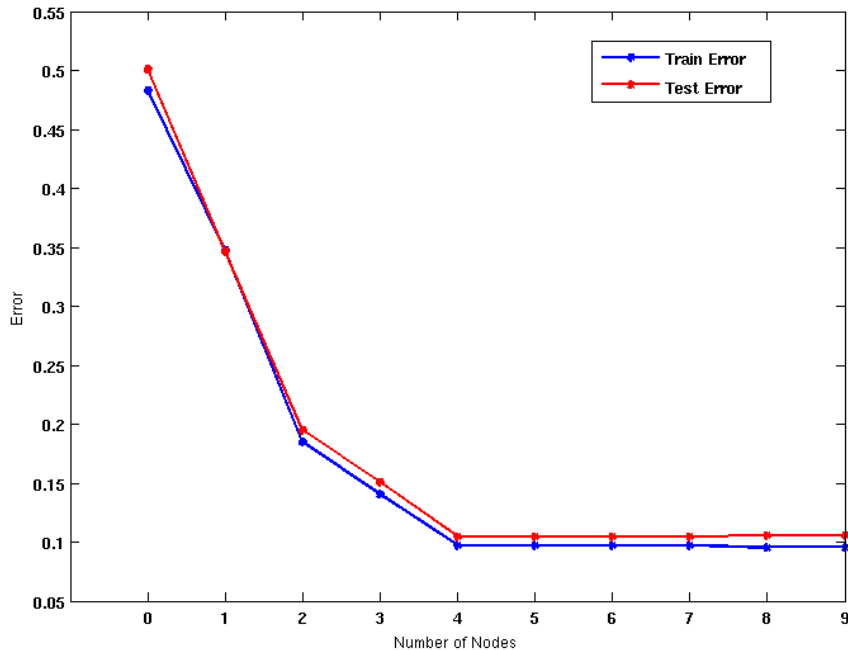
Decision Tree



Increasing number of nodes in decision trees



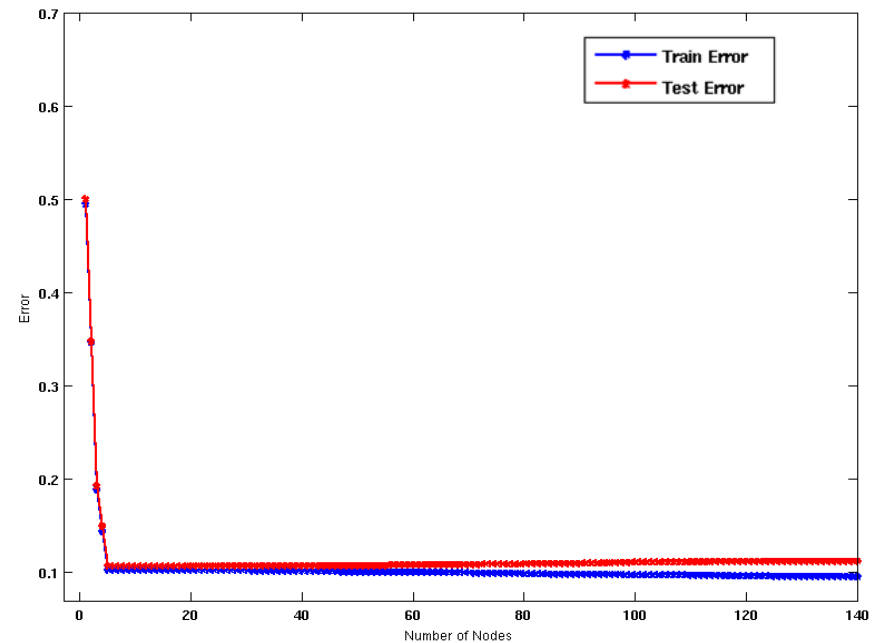
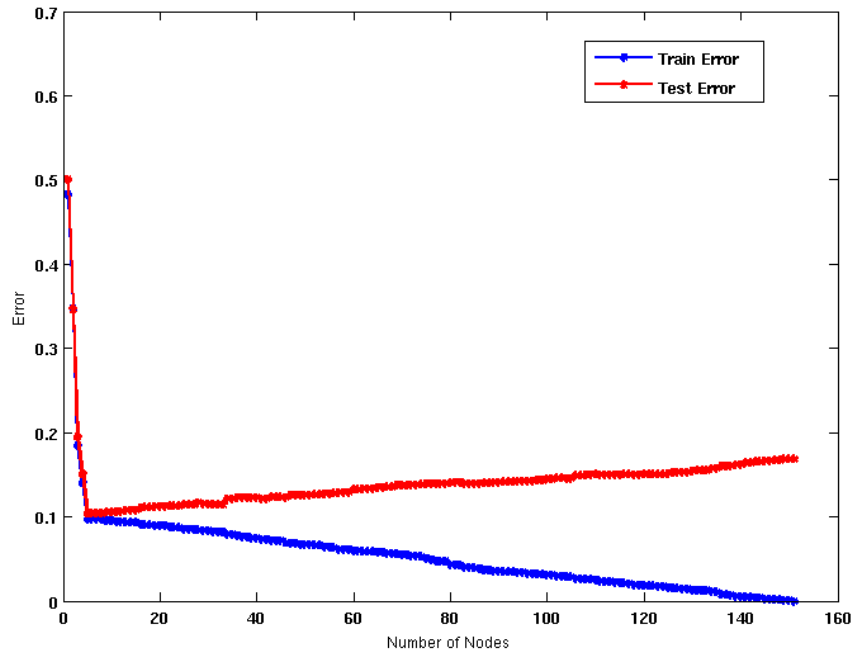
Model overfitting



Underfitting: when model is too simple, both training and test errors are large.

Overfitting: when model is too complex, training error is small but test error is large.

Model overfitting



Using twice the number of data instances

- If training data is under-representative, testing errors increase and training errors decrease on increasing number of nodes.
- Increasing the size of training data reduces the difference between training and testing errors at a given number of nodes.

Handling overfitting in decision trees

Pre-Pruning (early stopping rule):

- Stop the algorithm before it becomes a fully-grown tree.
- Typical stopping conditions for a node:
 - Stop if all instances belong to the same class.
 - Stop if all the attribute values are the same.
- More restrictive conditions:
 - Stop if number of instances is less than some user-specified threshold.
 - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test).
 - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
 - Stop if estimated generalization error falls below certain threshold.

Handling overfitting in decision trees

Post-pruning:

- Grow decision tree to its entirety.
- Subtree replacement:
 - Trim the nodes of the decision tree in a bottom-up fashion.
 - If generalization error improves after trimming, replace sub-tree by a leaf node.
 - Class label of leaf node is determined from majority class of instances in the sub-tree.
- Subtree raising:
 - Replace subtree with most frequently used branch.

PERFORMANCE ASSESSMENT

Evaluating performance of classifier

- Model evaluation
 - Performed after model has been constructed.
 - Purpose is to estimate **performance of classifier on previously unseen data** (e.g., test set).
- Model selection
 - Performed during model building.
 - Purpose is to ensure that model is not overly complex (to avoid overfitting).
 - Need to estimate **generalization error**.

Accuracy

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	a (TP)	b (FN)
	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

Problem with Accuracy

- Consider a 2-class problem
 - Number of Class NO samples = 990
 - Number of Class YES samples = 10
- Evaluation measures such as accuracy is not well-suited for imbalanced class
- Detecting the rare class is like finding needle in a haystack

Problem with Accuracy

- Consider a 2-class problem
 - Number of Class NO samples = 990
 - Number of Class YES samples = 10
- If a model predicts everything to be class NO, accuracy is $990/1000 = 99\%$
 - This is misleading because the model does not detect any class YES example
 - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

Alternative Measures

- **Precision** p is the number of **correctly classified positive examples** divided by the total number of examples that are classified as positive.
- **Recall** r is the number of **correctly classified positive examples** divided by the total number of actual positive examples in the test set.

$$p = \frac{TP}{TP + FP}$$

$$r = \frac{TP}{TP + FN}$$

Alternative Measures

	PREDICTED CLASS		
		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

$$\text{Precision (p)} = \frac{a}{a + c}$$

$$\text{Recall (r)} = \frac{a}{a + b}$$

$$\text{F - measure (F)} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

Alternative Measures

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	40	10
	10	40

Precision (p) = 0.8

Recall (r) = 0.8

F - measure (F) = 0.8

Accuracy = 0.8

ACTUAL CLASS	PREDICTED CLASS	
	Class=Yes	Class=No
Class=Yes	40	10
	1000	4000

Precision (p) = ~ 0.04

Recall (r) = 0.8

F - measure (F) = ~ 0.08

Accuracy = 0.8

Alternative Measures

Precision, recall, and F-measure are class-specific!

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

	PREDICTED CLASS		
		Class=No	Class=Yes
ACTUAL CLASS	Class=No	d	c
	Class=Yes	b	a

$$\text{Precision}_{\text{YES}} (p) = \frac{a}{a + c}$$

$$\text{Recall}_{\text{YES}} (r) = \frac{a}{a + b}$$

$$\text{F-measure}_{\text{YES}} (F) = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

$$\text{Precision}_{\text{NO}} (p) = \frac{d}{d + b}$$

$$\text{Recall}_{\text{NO}} (r) = \frac{d}{d + c}$$

$$\text{F-measure}_{\text{NO}} (F) = \frac{2rp}{r + p} = \frac{2d}{2d + b + c}$$

Evaluation metrics

ACTUAL CLASS	PREDICTED CLASS		
		Yes	No
	Yes	TP	FN
	No	FP	TN

α is the probability that we reject the null hypothesis when it is true. This is a Type I error or a false positive (FP).

β is the probability that we accept the null hypothesis when it is false. This is a Type II error or a false negative (FN).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive Predictive Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN Rate = \frac{TN}{TN + FP}$$

$$FP Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

Model selection

- Classification methods often have input parameters
 - k in kNN
 - tree height or number of nodes in decision tree
- How to choose the right model?
 - Split the **training set** into **training** + **validation**
 - Same split ratio as for the initial **training/test** split (80/20 or 90/10)
 - Train multiple models using different input parameter values on the new **training** set
 - Evaluate each one using the **validation** set as unknown samples
 - Choose the model that provides the best tradeoff between sensitivity and fall-out (1-specificity)
- Report results only on chosen “best” model
 - Evaluating the model using the **test** set as unknown samples

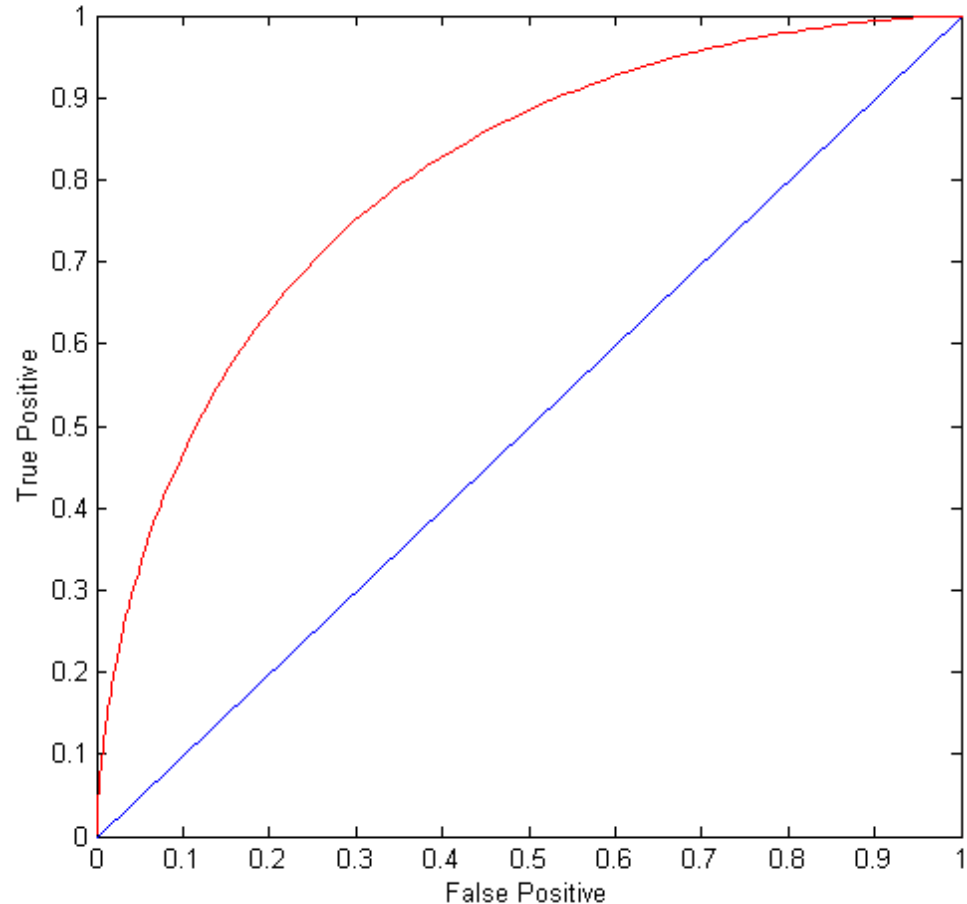
Receiver operating characteristic (ROC Curve)

- A graphical approach for displaying trade-off between detection rate and false alarm rate
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TP Rate (TPR) against FP Rate (FPR)
 - Performance of a model represented as a point in an ROC curve
 - Changing the threshold parameter of classifier changes the location of the point

ROC Curve

(TPR,FPR):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal
- Diagonal line:
 - Random guessing
 - Below diagonal line:
 - prediction is opposite of the true class



Model Selection Issues

- **Accuracy**
 - Classifier accuracy: predicting class label
- **Speed**
 - Time to construct the model (training time)
 - Time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
 - Understanding and insight provided by the model
 - Model (e.g., decision tree) size or compactness

Methods for classifier evaluation

- Holdout
 - Reserve $k\%$ for training and $(100-k)\%$ for testing.
- Random subsampling
 - Repeated holdout.
- Cross validation
 - Partition data into k disjoint subsets.
 - k -fold: train on $k-1$ partitions, test on the remaining one.
 - Leave-one-out: $k=n$.

Handling multiple classes

- There are two issues of importance:
 - The dataset has multiple classes:
 - E.g., classify documents into different topics.
 - The objects can belong to multiple classes:
 - E.g., each document can belong to multiple topics.
- Handling multiple classes:
 - Approach #1: Build a classifier that can directly classify in one of the different classes;
 - E.g., decision trees, k-NN, etc.
 - Approach #2: Build a set of binary classifiers.

Handling multiple classes via a set of binary classifiers

- Consider the case in which you have m classes.
- Build m classifiers, each of them designed to classify the objects of one class versus the rest:
 - *One-vs-rest classifiers*
 - The “once class” is often called the +ve class and the objects of the other classes are treated as the –ve class.
- During prediction, apply each one of the m binary models, and assign the object to the model that generates the *strongest* prediction.
- How do we measure prediction strength?

INSTANCE-BASED LEARNING

Instance based classifiers

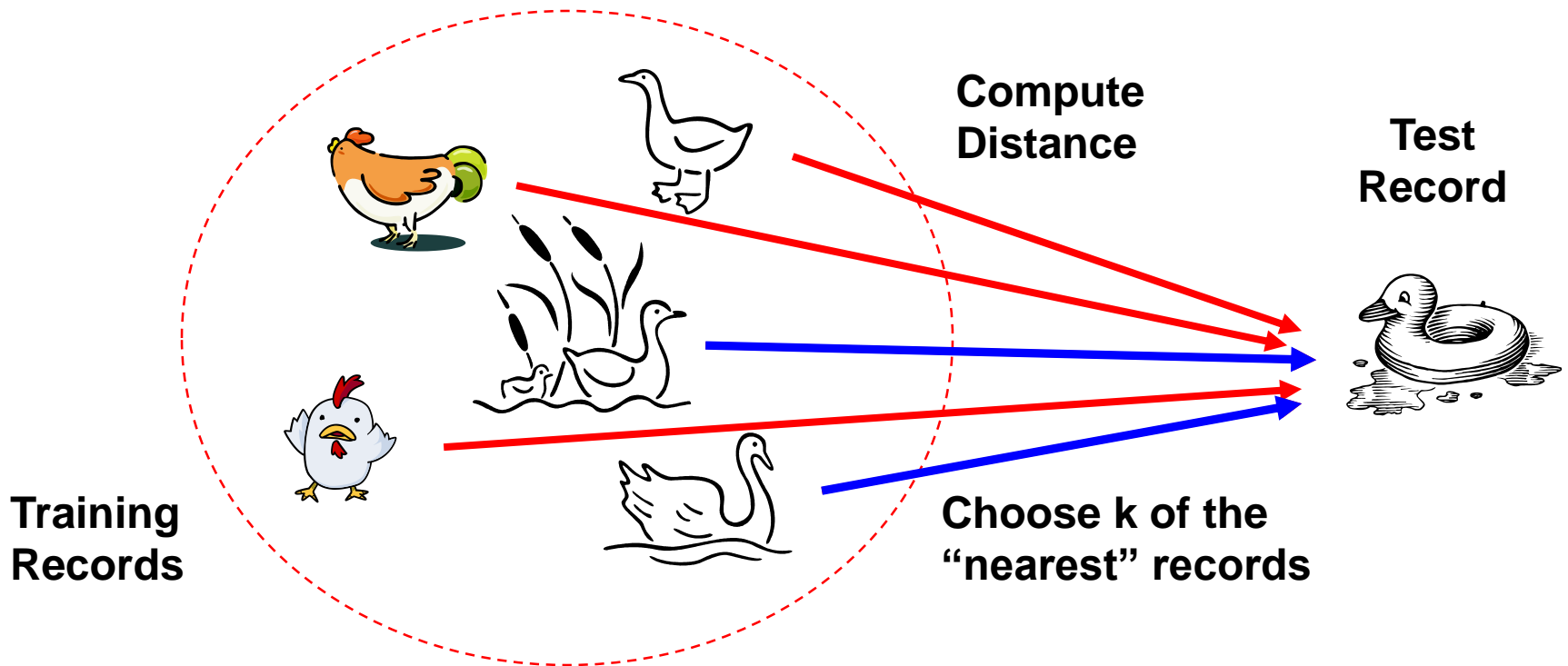
Examples:

- Rote-learner
 - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly.
- Nearest neighbor
 - Uses k “closest” points (nearest neighbors) for performing classification.

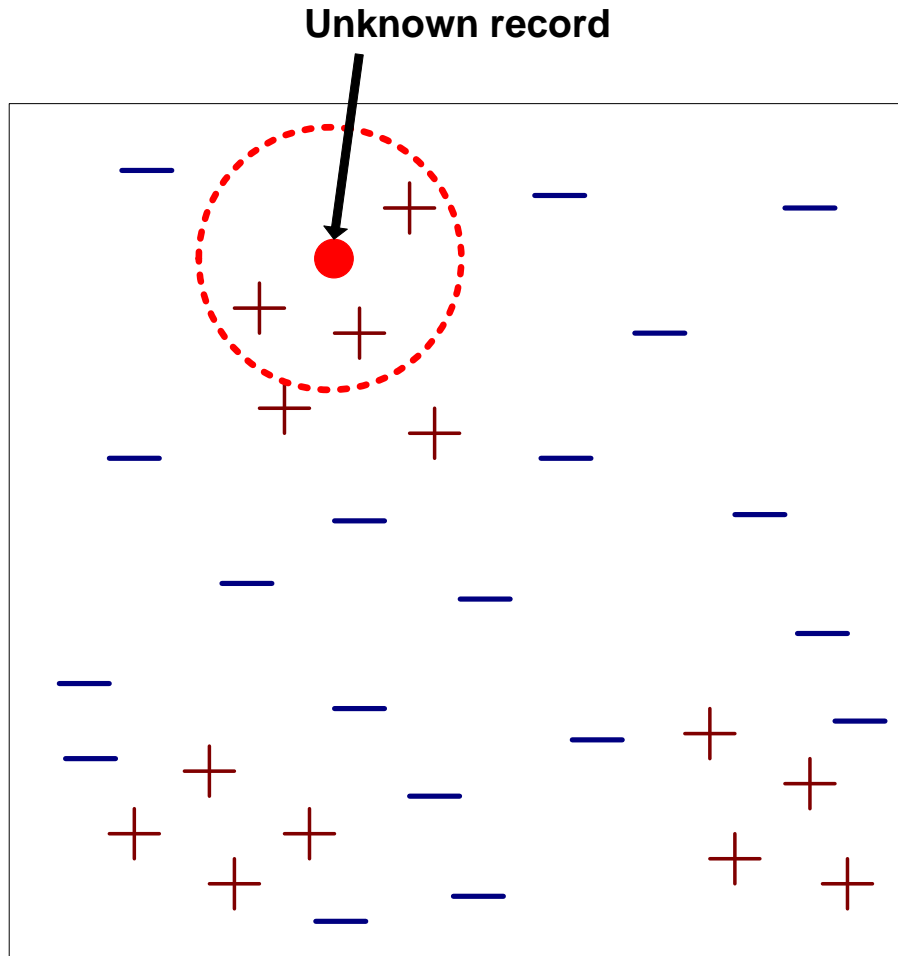
Nearest neighbor classifiers

Basic idea:

- If it walks like a duck, quacks like a duck, then it's probably a duck.



Nearest neighbor classifiers



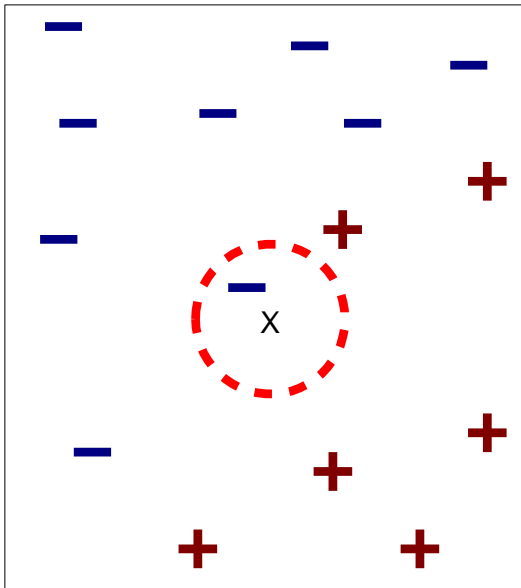
Requires three things:

1. The set of labeled records.
2. Distance metric to compute distance between records/
3. The value of k , the number of nearest neighbors to retrieve.

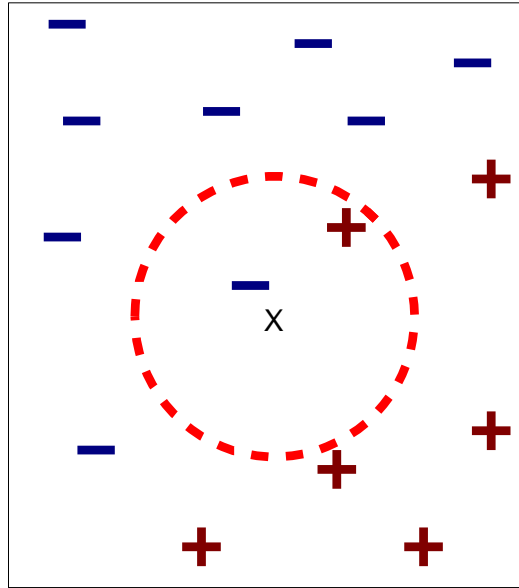
To classify an unknown record:

- Compute distance to other training records.
- Identify k nearest neighbors .
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote).

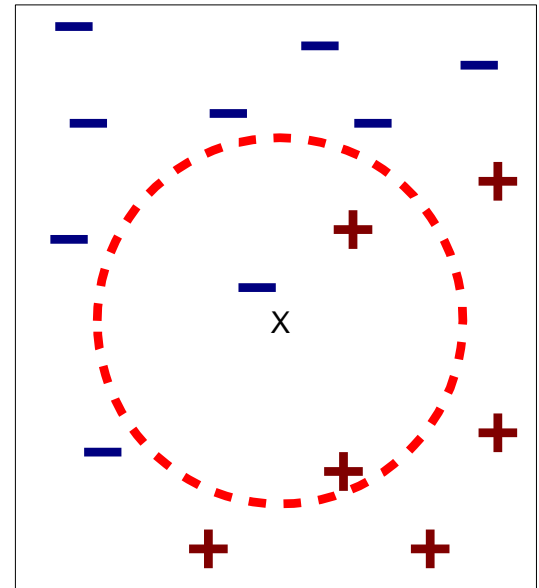
Definition of nearest neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor

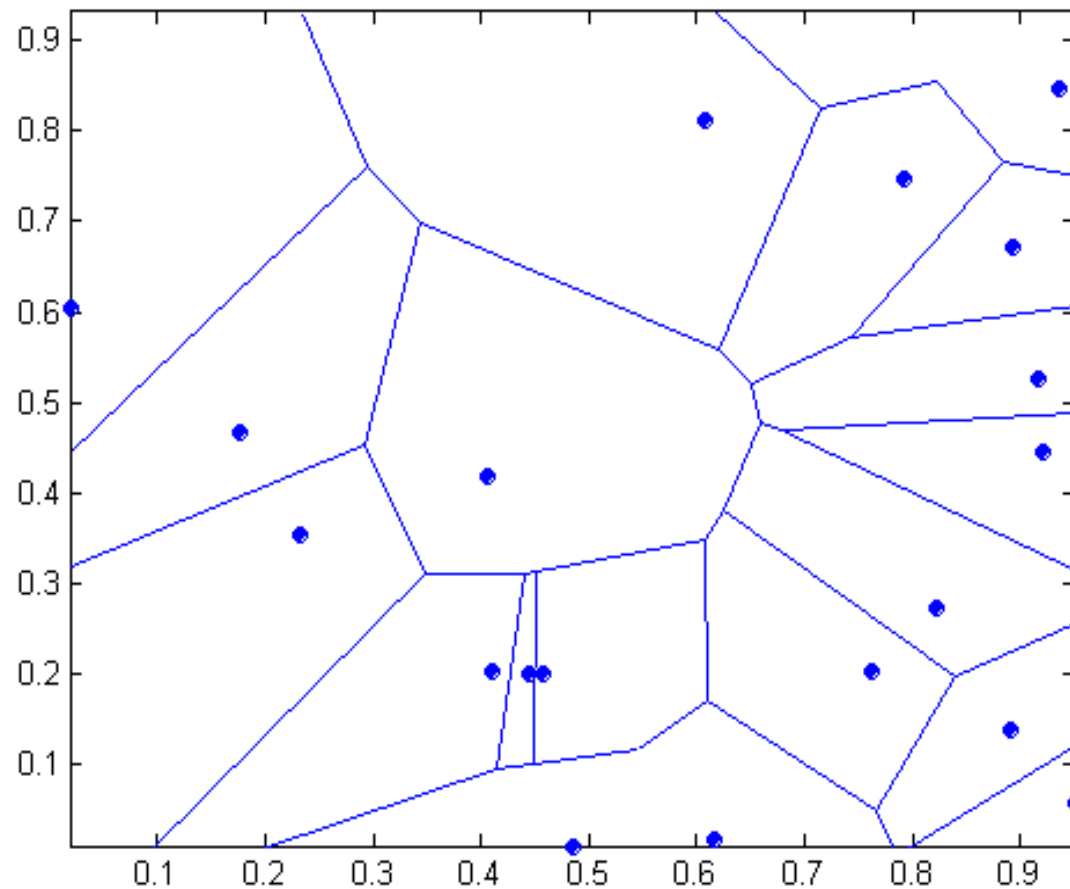


(c) 3-nearest neighbor

The k -nearest neighbors of a record x are data points that have the k smallest distances to x .

1 nearest-neighbor

Voronoi diagram



Nearest neighbor classification

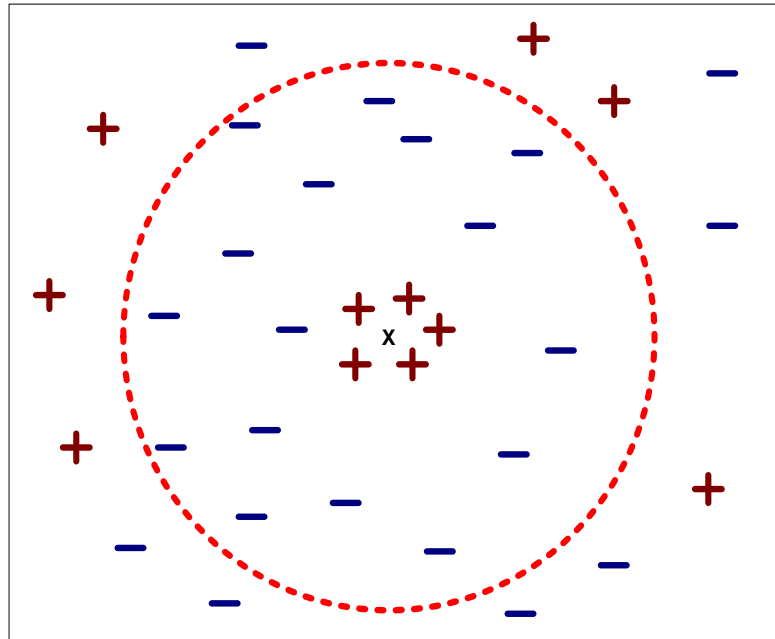
- Compute distance between two points:
 - Euclidean distance

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Other proximity measures can also be used
- Determine the class from nearest neighbor list
 - Take the majority vote of class labels among the k-nearest neighbors.
 - Weigh the vote according to distance
 - e.g., weight factor, $w = 1/d^2$.

Nearest neighbor classification...

- Choosing the value of k :
 - If k is too small, sensitive to noise points.
 - If k is too large, neighborhood may include points from other classes.



Nearest neighbor classification...

- Scaling issues
 - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes.
 - Example:
 - height of a person may vary from 1.5m to 1.8m.
 - weight of a person may vary from 90lb to 300lb.
 - income of a person may vary from \$10K to \$1M.

Nearest neighbor classification...

Selection of the right similarity measure is critical:

1 1 1 1 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 1 1 1 1

vs

0 0 0 0 0 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 0 0 0 0

Euclidean distance = 1.4142 for both pairs

Cosine distance = 0.0909

Cosine distance = 1.0

Nearest neighbor classification...

- k-NN classifiers are lazy learners since they do not build models explicitly.
- Classifying unknown records are relatively expensive.
- Can produce arbitrarily shaped decision boundaries.
- Easy to handle variable interactions since the decisions are based on local information.
- Selection of right proximity measure is essential.
- Superfluous or redundant attributes can create problems.
- Missing attributes are hard to handle.

Improving k-NN efficiency

- Avoid having to compute distance to all objects in the training set:
 - Multi-dimensional access methods (k-d trees).
 - Intelligently prune the search space.
 - Fast approximate similarity search.
 - Locality Sensitive Hashing (LSH).
 - Filtering-based exact similarity search.
- Condensing
 - Determine a smaller set of objects that give the same performance.
- Editing
 - Remove objects to improve efficiency.
- Summarizing
 - Represent each class in a form of one-or-more summaries and compute distances to these summaries.

1. Naïve Bayes
2. Bayesian Belief Networks

BAYESIAN CLASSIFIERS

Bayes classifier

A probabilistic framework for solving classification problems.

- Conditional Probability:

$$P(H|E) = \frac{P(H, E)}{P(E)}$$

$$P(E|H) = \frac{P(H, E)}{P(H)}$$

- Bayes theorem:

$$P(\textcolor{red}{H}|\textcolor{blue}{E}) = \frac{P(\textcolor{blue}{E}|\textcolor{red}{H})P(\textcolor{red}{H})}{P(\textcolor{blue}{E})}$$

Example of Bayes theorem

- Given:
 - A doctor knows that meningitis causes stiff neck 50% of the time.
 - Prior probability of any patient having meningitis is 1/50,000.
 - Prior probability of any patient having stiff neck is 1/20.
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Using Bayes theorem for classification

- Consider each attribute and class label as random variables.
- Given a record with attributes (X_1, X_2, \dots, X_d)
 - Goal is to predict class Y .
 - Specifically, we want to find the value of Y that maximizes $P(Y | X_1, X_2, \dots, X_d)$.
- Can we estimate $P(Y | X_1, X_2, \dots, X_d)$ directly from data?

Example data

Given a test record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Can we estimate

$P(\text{Evade} = \text{Yes} \mid \text{Refund}=\text{No}, \text{Divorced}, \text{Income}=120\text{k})$

and

$P(\text{Evade} = \text{No} \mid \text{Refund}=\text{No}, \text{Divorced}, \text{Income}=120\text{k})?$

In the following we will replace

Evade = Yes by Yes, and

Evade = No by No

Using Bayes theorem for classification

- Approach:
 - compute posterior probability $P(Y | X_1, X_2, \dots, X_d)$ using the Bayes theorem

$$P(Y | X_1 X_2 \dots X_n) = \frac{P(X_1 X_2 \dots X_d | Y) P(Y)}{P(X_1 X_2 \dots X_d)}$$

- *Maximum a-posteriori*: Choose Y that maximizes $P(Y | X_1, X_2, \dots, X_d)$
 - Equivalent to choosing value of Y that maximizes $P(X_1, X_2, \dots, X_d | Y) P(Y)$
- How to estimate $P(X_1, X_2, \dots, X_d | Y)$?

Example Data

Given a Test Record:

$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Using Bayes Theorem:

$$\square P(\text{Yes} | X) = \frac{P(X | \text{Yes})P(\text{Yes})}{P(X)}$$

$$\square P(\text{No} | X) = \frac{P(X | \text{No})P(\text{No})}{P(X)}$$

\square How to estimate $P(X | \text{Yes})$ and $P(X | \text{No})$?

Naïve Bayes classifier

- Assume independence among attributes X_i when class is given:
 - $P(X_1, X_2, \dots, X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) \dots P(X_d | Y_j)$
 - Now we can estimate $P(X_i | Y_j)$ for all X_i and Y_j combinations from the training data
 - New point is classified to Y_j if

$$P(Y_j) \times \prod_{i=1}^m P(X_i | Y_j)$$

is maximal.

Conditional independence

- **X** and **Y** are conditionally independent given **Z** if $P(\mathbf{X} | \mathbf{YZ}) = P(\mathbf{X} | \mathbf{Z})$.
- Example: Arm length and reading skills
 - Young child has shorter arm length and limited reading skills, compared to adults.
 - If age is fixed, no apparent relationship between arm length and reading skills.
 - Arm length and reading skills are conditionally independent given age.

Naïve Bayes on example data

Given a Test Record:

$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

□ $P(X \mid \text{Yes}) =$

$P(\text{Refund} = \text{No} \mid \text{Yes}) \times$

$P(\text{Divorced} \mid \text{Yes}) \times$

$P(\text{Income} = 120\text{K} \mid \text{Yes})$

□ $P(X \mid \text{No}) =$

$P(\text{Refund} = \text{No} \mid \text{No}) \times$

$P(\text{Divorced} \mid \text{No}) \times$

$P(\text{Income} = 120\text{K} \mid \text{No})$

Estimate Probabilities from Data

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class prior: $P(Y) = N_c / N$
 - e.g., $P(\text{No}) = 7/10$,
 $P(\text{Yes}) = 3/10$
- For categorical attributes:
$$P(X_i \mid Y_k) = |X_{ik}| / N_{c_k}$$
 - where $|X_{ik}|$ is number of instances having attribute value X_i and belonging to class Y_k
 - Examples:
 $P(\text{Status}=\text{Married} \mid \text{No}) = 4/7$
 $P(\text{Refund}=\text{Yes} \mid \text{Yes})=0$

Estimate Probabilities from Data

- For continuous attributes:
 - **Discretization:** Partition the range into bins:
 - Replace continuous value with bin value
 - Attribute changed from continuous to ordinal
 - **Probability density estimation:**
 - Assume attribute follows a normal distribution
 - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
 - Once probability distribution is known, use it to estimate the conditional probability $P(X_i | Y)$

Estimate Probabilities from Data

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(X_i | Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

– One for each (X_i, Y_i) pair

- For (Income, Class=No):

– If Class=No

- sample mean = 110
- sample variance = 2550

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi(50.50)}} e^{-\frac{(120-110)^2}{2(2550)}} = 0.0077$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} \mid \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} \mid \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} \mid \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} \mid \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} \mid \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} \mid \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} \mid \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} \mid \text{Yes}) = 0$$

For Taxable Income:

If class = No: sample mean = 110

sample variance = 2550

If class = Yes: sample mean = 90

sample variance = 16.6

$$\begin{aligned} \square \quad P(X \mid \text{No}) &= P(\text{Refund}=\text{No} \mid \text{No}) \\ &\quad \times P(\text{Divorced} \mid \text{No}) \\ &\quad \times P(\text{Income}=120\text{K} \mid \text{No}) \\ &= 4/7 \times 1/7 \times 0.0077 = 0.0006 \end{aligned}$$

$$\begin{aligned} \square \quad P(X \mid \text{Yes}) &= P(\text{Refund}=\text{No} \mid \text{Yes}) \\ &\quad \times P(\text{Divorced} \mid \text{Yes}) \\ &\quad \times P(\text{Income}=120\text{K} \mid \text{Yes}) \\ &= 1 \times 1/3 \times 1.8 \times 10^{-13} = 6.12 \times 10^{-14} \end{aligned}$$

Since $P(X \mid \text{No})P(\text{No}) > P(X \mid \text{Yes})P(\text{Yes})$

Therefore $P(\text{No} \mid X) > P(\text{Yes} \mid X)$

\Rightarrow Class = No

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Divorced}, \text{Income} = 120\text{K})$$

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} \mid \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} \mid \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} \mid \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} \mid \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} \mid \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} \mid \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} \mid \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} \mid \text{Yes}) = 0$$

For Taxable Income:

If class = No: sample mean = 110

sample variance = 2550

If class = Yes: sample mean = 90

sample variance = 16.6

$$\square \quad P(\text{Yes}) = 3/10$$

$$P(\text{No}) = 7/10$$

$$\square \quad P(\text{Yes} \mid \text{Divorced}) = 1/3 \times 3/10 / P(\text{Divorced})$$

$$P(\text{No} \mid \text{Divorced}) = 1/7 \times 7/10 / P(\text{Divorced})$$

$$\square \quad P(\text{Yes} \mid \text{Refund} = \text{No}, \text{Divorced}) = 1 \times 1/3 \times 3/10 / P(\text{Divorced}, \text{Refund} = \text{No})$$

$$P(\text{No} \mid \text{Refund} = \text{No}, \text{Divorced}) = 4/7 \times 1/7 \times 7/10 / P(\text{Divorced}, \text{Refund} = \text{No})$$

Issues with Naïve Bayes Classifier

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} \mid \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} \mid \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} \mid \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} \mid \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} \mid \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} \mid \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} \mid \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} \mid \text{Yes}) = 0$$

- $P(\text{Yes}) = 3/10$

$$P(\text{No}) = 7/10$$

- $P(\text{Yes} \mid \text{Married}) = 0 \times 3/10 / P(\text{Married})$

$$P(\text{No} \mid \text{Married}) = 4/7 \times 7/10 / P(\text{Married})$$

For Taxable Income:

If class = No: sample mean = 110

sample variance = 2550

If class = Yes: sample mean = 90

sample variance = 16.6

Issues with Naïve Bayes Classifier

Consider the table with Tid = 7 deleted

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

$$P(\text{Refund} = \text{Yes} \mid \text{No}) = 2/6$$

$$P(\text{Refund} = \text{No} \mid \text{No}) = 4/6$$

$$\rightarrow P(\text{Refund} = \text{Yes} \mid \text{Yes}) = 0$$

$$P(\text{Refund} = \text{No} \mid \text{Yes}) = 1$$

$$P(\text{Marital Status} = \text{Single} \mid \text{No}) = 2/6$$

$$\rightarrow P(\text{Marital Status} = \text{Divorced} \mid \text{No}) = 0$$

$$P(\text{Marital Status} = \text{Married} \mid \text{No}) = 4/6$$

$$P(\text{Marital Status} = \text{Single} \mid \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} \mid \text{Yes}) = 0/3$$

For Taxable Income:

If class = No: sample mean = 91.6

sample variance = 622.2

If class = Yes: sample mean = 90

sample variance = 16.6

Given $X = (\text{Refund} = \text{Yes}, \text{Divorced}, 120K)$

$$P(X \mid \text{No}) = 2/6 \times 0 \times 0.0083 = 0$$

$$P(X \mid \text{Yes}) = 0 \times 1/3 \times 1.8 \times 10^{-13} = 0$$

Naïve Bayes will not be able to classify

X as Yes or No!

Issues with Naïve Bayes Classifier

- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:

$$\text{Original : } P(A_i | C) = \frac{N_{ic}}{N_c}$$

c : number of classes

p : prior probability of the class

m : parameter

$$\text{Laplace : } P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$$

N_c : number of instances in the class

$$\text{m - estimate : } P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

N_{ic} : number of instances having attribute value A_i in class c

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A | M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A | N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A | M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A | N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

$$P(A|M)P(M) > P(A|N)P(N)$$

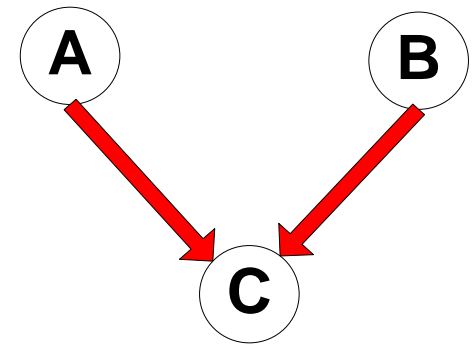
=> Mammals

Naïve Bayes (Summary)

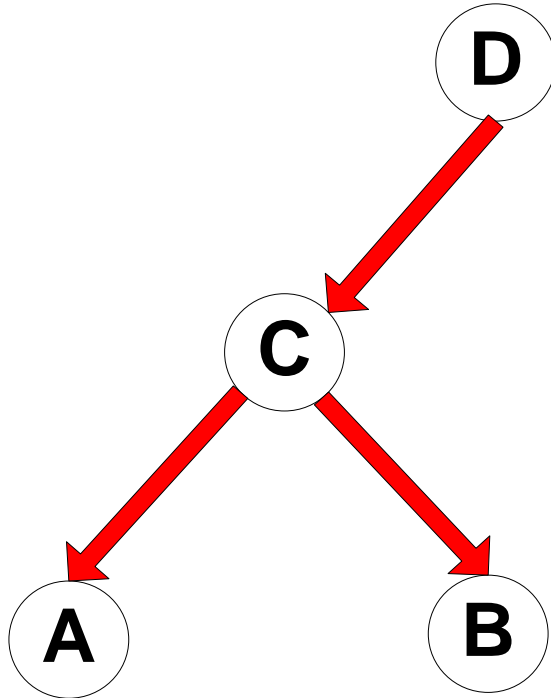
- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
 - Use other techniques such as Bayesian Belief Networks (BBN)

Bayesian Belief Networks

- Provides graphical representation of probabilistic relationships among a set of random variables
- Consists of:
 - A directed acyclic graph (dag)
 - Node corresponds to a variable
 - Arc corresponds to dependence relationship between a pair of variables
 - A probability table associating each node to its immediate parent



Conditional Independence



D is parent of C

A is child of C

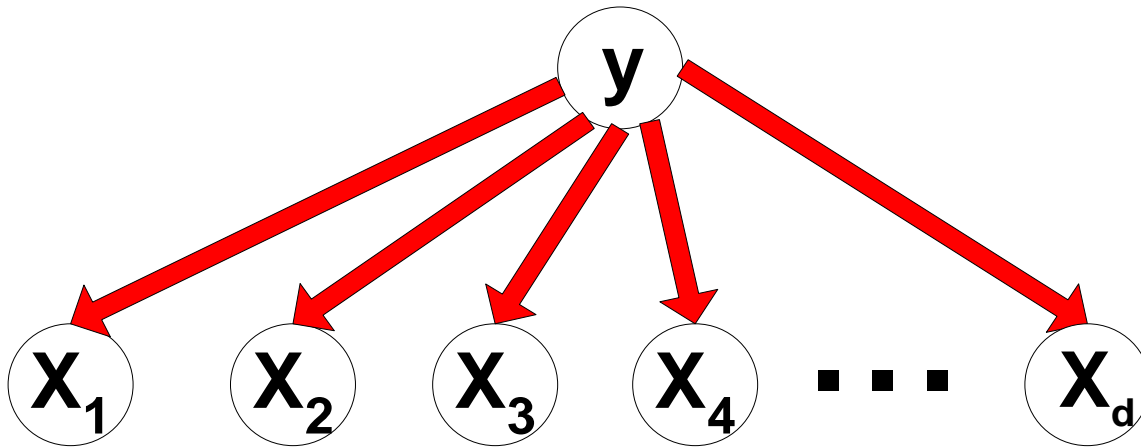
B is descendant of D

D is ancestor of A

- A node in a Bayesian network is conditionally independent of all of its nondescendants, if its parents are known

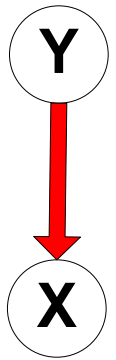
Conditional Independence

- Naïve Bayes assumption:



Probability Tables

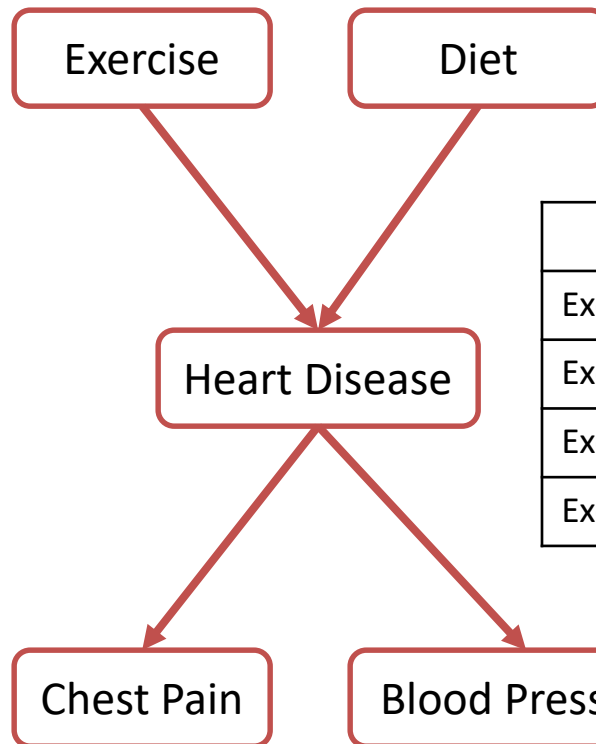
- If X does not have any parents, table contains prior probability $P(X)$
- If X has only one parent (Y), table contains conditional probability $P(X|Y)$
- If X has multiple parents (Y_1, Y_2, \dots, Y_k), table contains conditional probability $P(X|Y_1, Y_2, \dots, Y_k)$



Example of Bayesian Belief Network

Exercise=Yes	0.70
--------------	------

Diet=Healthy	0.25
--------------	------



	HD=Yes
Exercise=Yes, Diet=Healthy	0.25
Exercise=No, Diet=Healthy	0.55
Exercise=Yes, Diet=Unhealthy	0.45
Exercise=No, Diet=Unhealthy	0.75

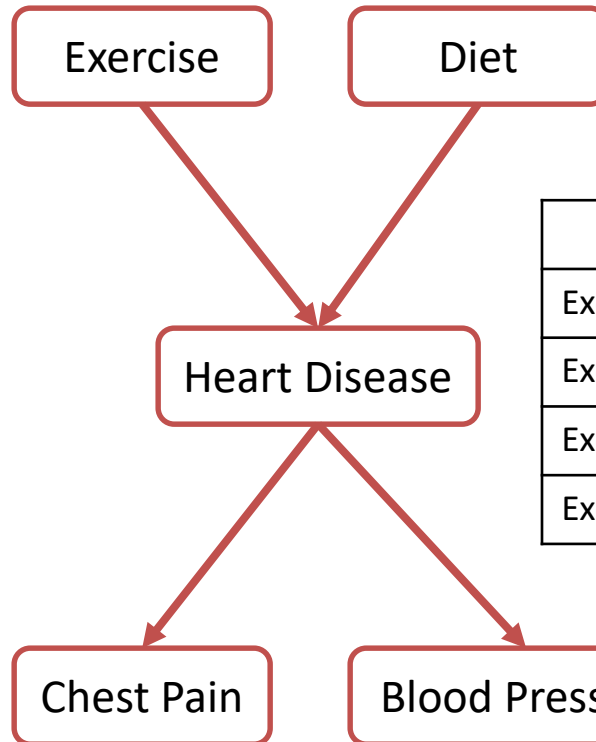
	CP=Yes
HD=Yes	0.80
HD=No	0.01

	BP=High
HD=Yes	0.85
HD=No	0.20

Example of Bayesian Belief Network

Exercise=Yes	0.70
Exercise=No	0.30

Diet=Healthy	0.25
Diet=Unhealthy	0.75



	HD=Yes	HD=No
Exercise=Yes, Diet=Healthy	0.25	0.75
Exercise=No, Diet=Healthy	0.55	0.45
Exercise=Yes, Diet=Unhealthy	0.45	0.55
Exercise=No, Diet=Unhealthy	0.75	0.25

	CP=Yes	CP=No
HD=Yes	0.80	0.20
HD=No	0.01	0.99

	BP=High	BP=Low
HD=Yes	0.85	0.15
HD=No	0.20	0.80

Example of Inferencing using BBN

● Given: $X = (E=\text{No}, D=\text{Healthy}, CP=\text{Yes}, BP=\text{High})$

– Compute $P(HD | E, D, CP, BP)$?

• $P(HD=\text{Yes} | E=\text{No}, D=\text{Healthy}) = 0.55$

$P(CP=\text{Yes} | HD=\text{Yes}) = 0.8$

$P(BP=\text{High} | HD=\text{Yes}) = 0.85$

– $P(\text{HD}=\text{Yes} | E=\text{No}, D=\text{Healthy}, CP=\text{Yes}, BP=\text{High})$

$$\propto 0.55 \times 0.8 \times 0.85 = 0.374$$

• $P(HD=\text{No} | E=\text{No}, D=\text{Healthy}) = 0.45$

$P(CP=\text{Yes} | HD=\text{No}) = 0.01$

$P(BP=\text{High} | HD=\text{No}) = 0.2$

– $P(\text{HD}=\text{No} | E=\text{No}, D=\text{Healthy}, CP=\text{Yes}, BP=\text{High})$

$$\propto 0.45 \times 0.01 \times 0.2 = 0.0009$$

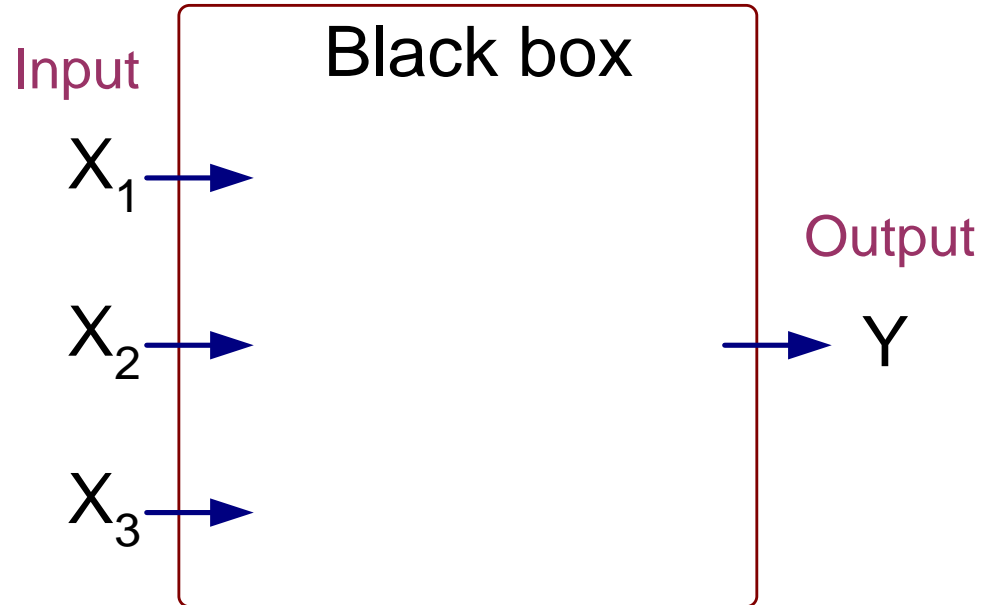


**Classify X
as Yes**

ARTIFICIAL NEURAL NETWORKS

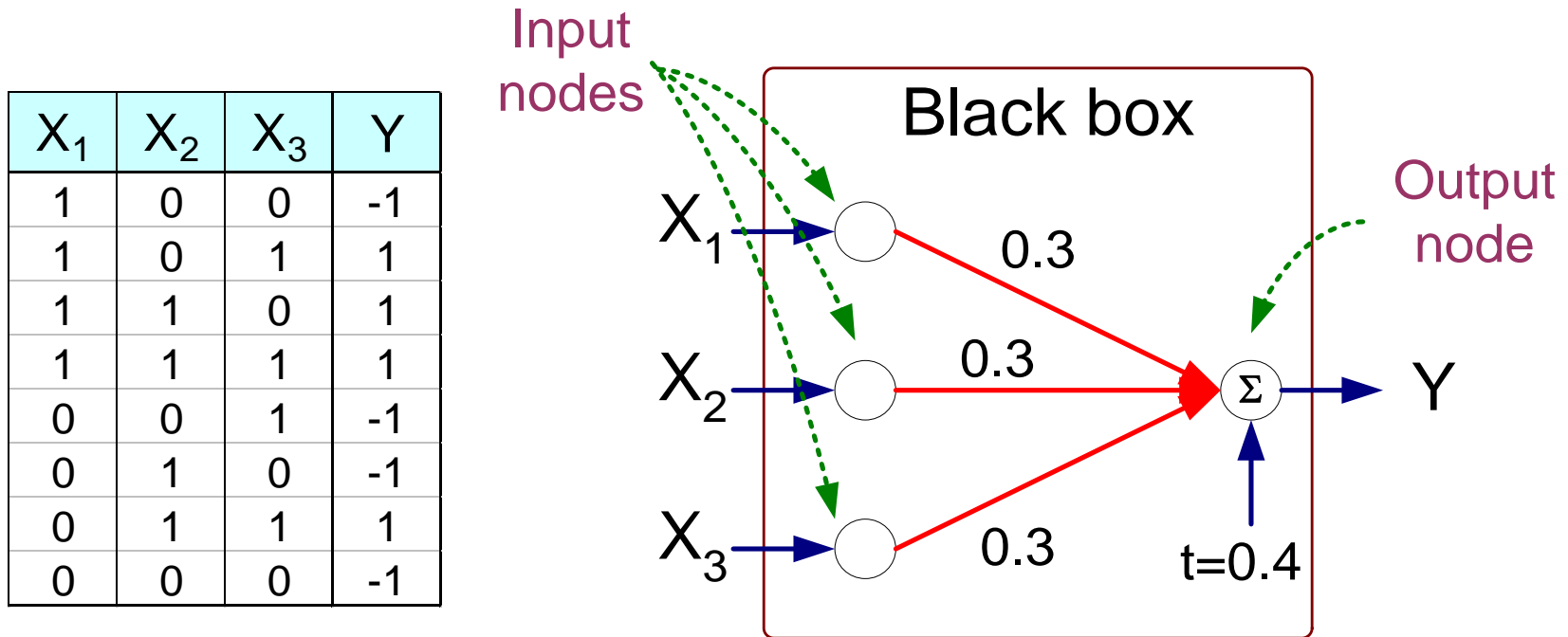
Consider the following

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

Consider the following

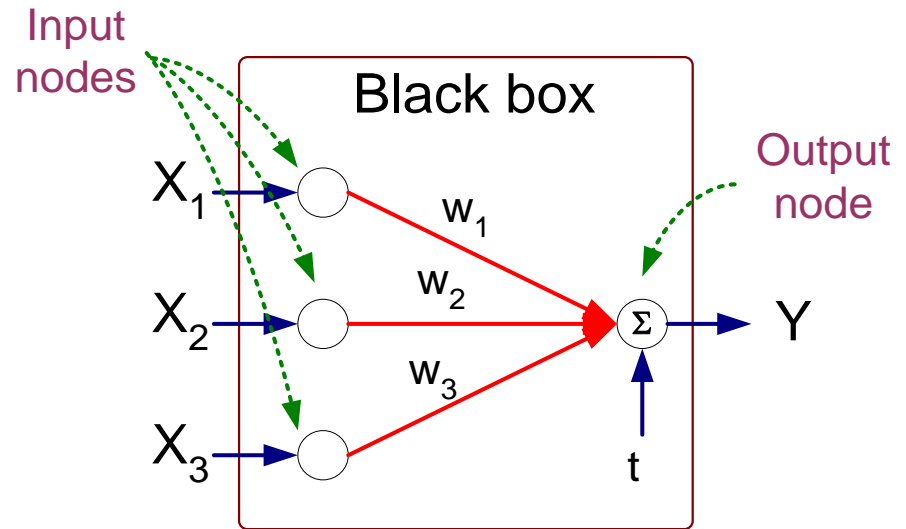


$$Y = \text{sign}(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } \text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Perceptron

- Model is an assembly of inter-connected nodes and weighted links.
- Output node sums up each of its input value according to the weights of its links.
- Compare output node against some threshold t .



Perceptron Model

$$Y = \text{sign}\left(\sum_{i=1}^d w_i X_i - t\right)$$
$$= \text{sign}\left(\sum_{i=0}^d w_i X_i\right)$$

Perceptron

- Single layer network
 - Contains only input and output nodes.
- Activation function: $f(w, x) = \text{sign}(\langle x, w \rangle)$
- Applying model is straightforward:

$$Y = \text{sign}(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } \text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

- E.g., $X_1 = 1, X_2 = 0, X_3 = 1 \Rightarrow y = \text{sign}(0.2) = 1$

Perceptron learning rule

- Initialize the weights (w_0, w_1, \dots, w_d)
- Repeat
 - For each training example (x_i, y_i)

- Compute $f(w, x_i)$
- Update the weights:

$$w^{(k+1)} = w^{(k)} + \eta [y_i - f(w^{(k)}, x_i)] x_i$$

- Until stopping condition is met.
- The above is an example of a **stochastic gradient descent** optimization method.

Perceptron learning rule

- Weight update formula:

$$w^{(k+1)} = w^{(k)} + \eta [y_i - f(w^{(k)}, x_i)] x_i \quad ; \quad \eta : \text{learning rate}$$

- Intuition:

- Update weight based on error: $e = [y_i - f(w^{(k)}, x_i)]$
- If $y=f(x,w)$, $e=0$: no update needed.
- If $y>f(x,w)$, $e=2$: weight must be increased so that $f(x,w)$ will increase.
- If $y<f(x,w)$, $e=-2$: weight must be decreased so that $f(x,w)$ will decrease.

Example of perceptron learning

$$w^{(k+1)} = w^{(k)} + \lambda[y_i - f(w^{(k)}, x_i)]x_i$$

$$Y = \text{sign}\left(\sum_{i=0}^d w_i x_i\right)$$

Epoch 1, $\lambda = 0.1$

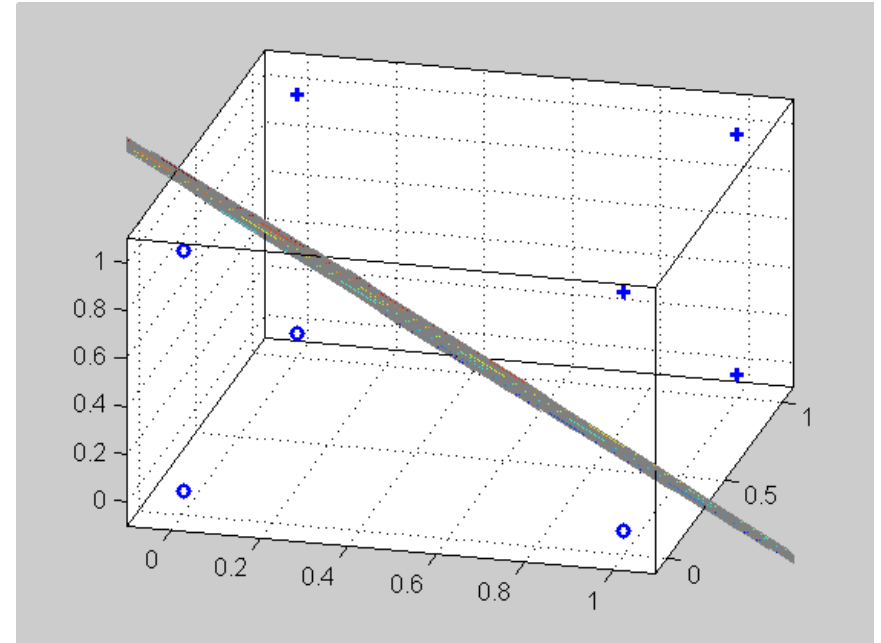
x_0	x_1	x_2	x_3	y
1	1	0	0	-1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1
1	0	0	1	-1
1	0	1	0	-1
1	0	1	1	1
1	0	0	0	-1

	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

Epoch	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	0	0.2	0.2
2	-0.2	0	0.4	0.2
3	-0.4	0	0.4	0.2
4	-0.4	0.2	0.4	0.4
5	-0.6	0.2	0.4	0.2
6	-0.6	0.4	0.4	0.2

Perceptron learning rule

- Since $f(w, x)$ is a linear combination of input variables, decision boundary is linear.



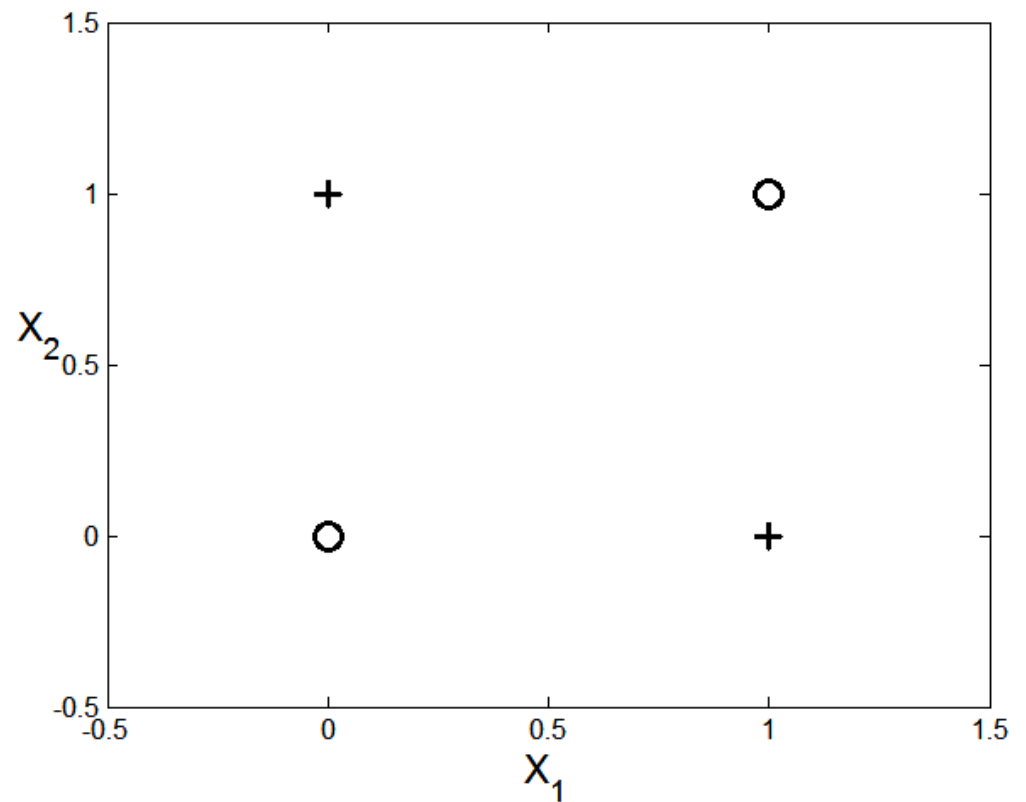
- For nonlinearly separable problems, perceptron learning algorithm will fail because no linear hyperplane can separate the data perfectly.

Nonlinearly separable data

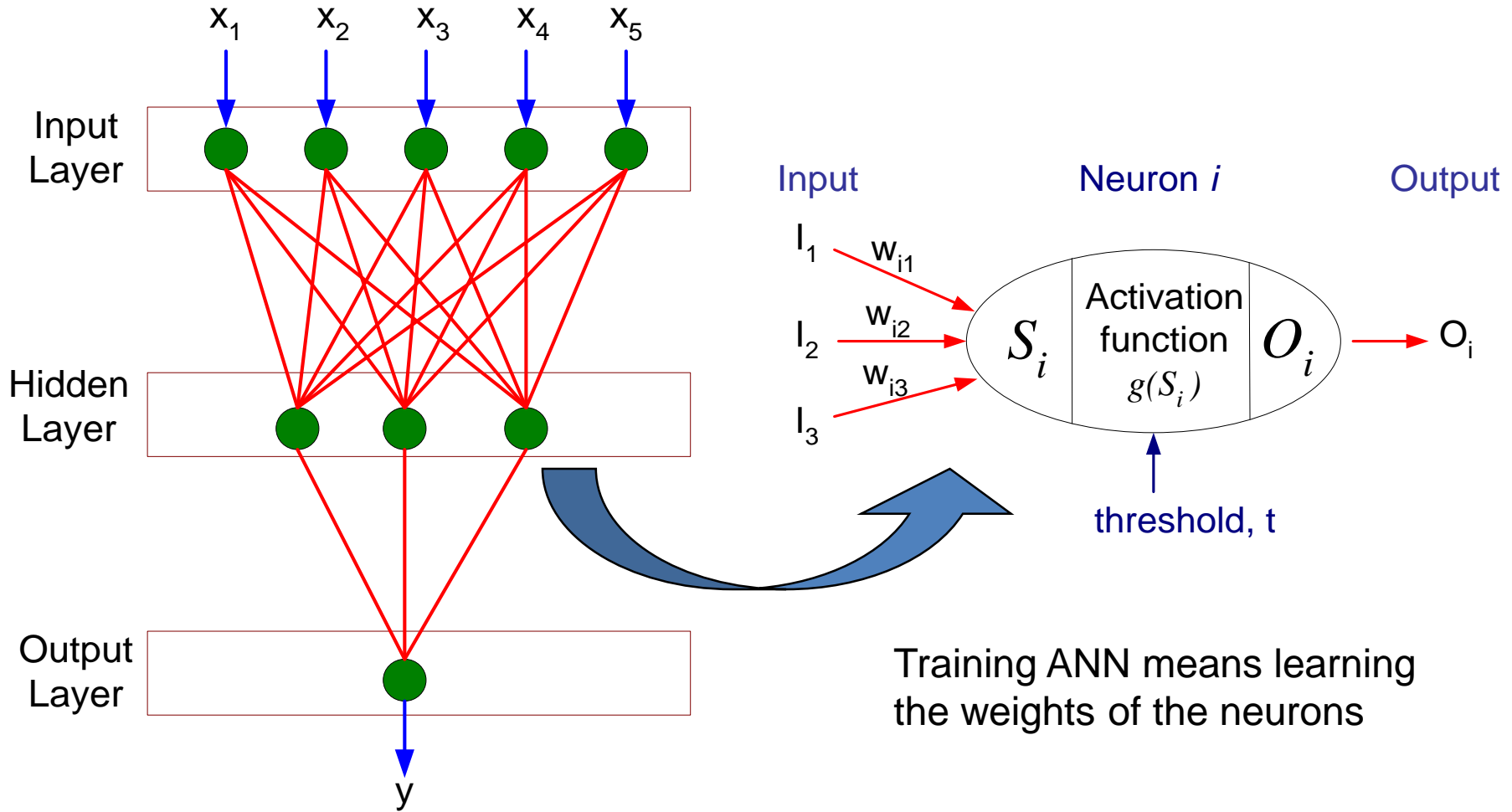
$$y = x_1 \oplus x_2$$

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1

XOR Data



Multilayer artificial neural networks (ANN)

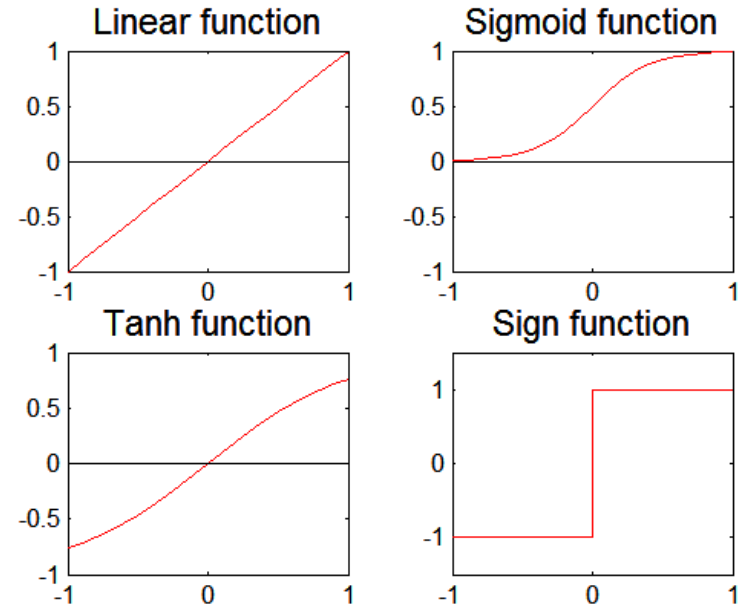


Artificial neural networks

- Various types of neural network topologies:
 - Single-layered network (perceptron) versus multi-layered network.
 - Feed-forward versus recurrent network.

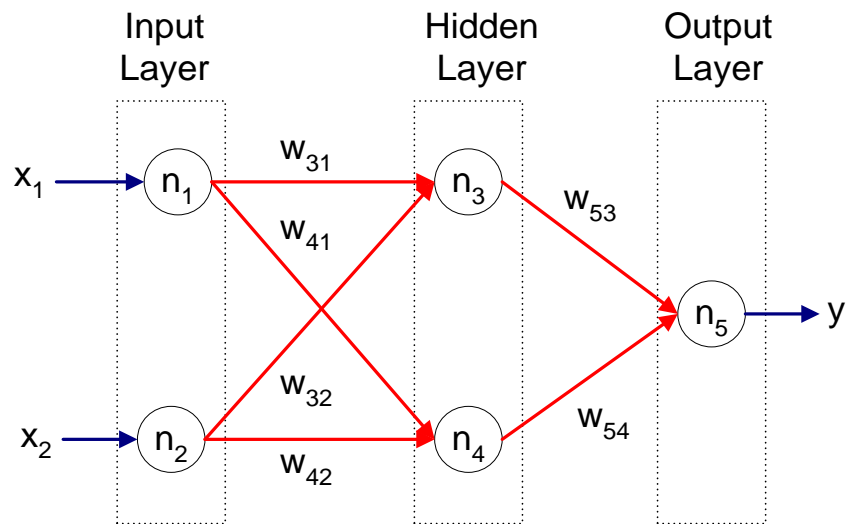
- Various types of activation functions (f):

$$Y = f\left(\sum_i w_i X_i\right)$$

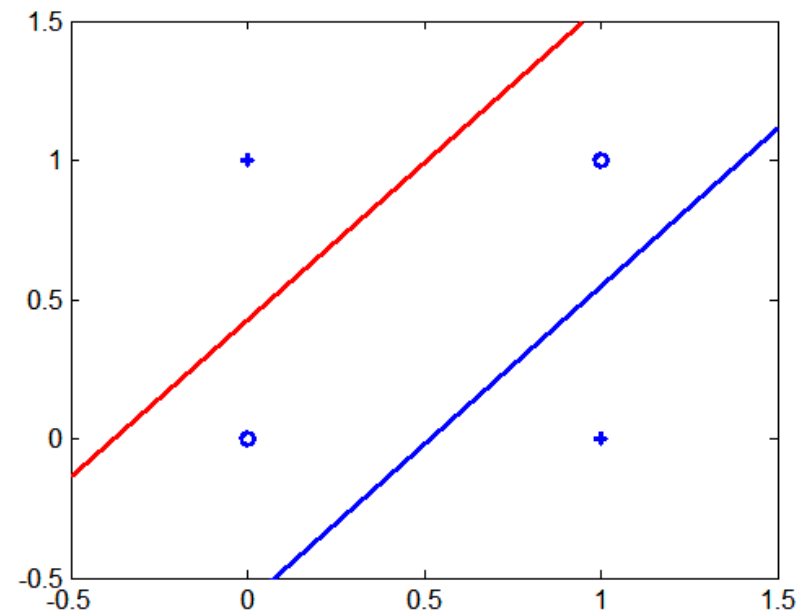


Artificial neural networks

Multi-layer neural network can solve any type of classification task involving nonlinear decision surfaces.



XOR Data



Characteristics of ANN

- Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large.
- Gradient descent may converge to local minimum.
- Model building can be very time consuming, but testing can be very fast.
- Can handle redundant attributes because weights are automatically learnt.
- Sensitive to noise in training data.
- Difficult to handle missing attributes.

Recent noteworthy developments in ANN

- Use in deep learning and unsupervised feature learning.
 - Seek to automatically learn a good representation of the input from unlabeled data.
- Google Brain project:
 - Learned the concept of a 'cat' by looking at unlabeled pictures from YouTube.
 - One billion connection network.
- Baidu deep learning projects:
 - Speak first keyboard
 - AI health chatbot

RULE-BASED CLASSIFIERS

Rule-based classifier

- Classify records by using a collection of “if...then...” rules.
- Rule: $(Condition) \rightarrow y$
 - where
 - *Condition* is a conjunction of attributes
 - y is the class label
 - *LHS*: rule antecedent or condition
 - *RHS*: rule consequent
 - Examples of classification rules:
 - $(\text{Blood Type}=\text{Warm}) \wedge (\text{Lay Eggs}=\text{Yes}) \rightarrow \text{Birds}$
 - $(\text{Taxable Income} < 50\text{K}) \wedge (\text{Refund}=\text{Yes}) \rightarrow \text{Evade}=\text{No}$

Rule-based classifier

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians

Rule coverage

- A rule r **covers** an instance x if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk \Rightarrow Bird

The rule R3 covers the grizzly bear \Rightarrow Mammal

Rule coverage and accuracy

- Coverage of a rule:
 - Fraction of records that satisfy the antecedent of a rule.
- Accuracy of a rule:
 - Fraction of records that satisfy the antecedent that also satisfy the consequent of a rule.

<i>Tid</i>	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

What if multiple rules cover an instance?

R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal.

A turtle triggers both R4 and R5.

A dogfish shark triggers none of the rules.

Ordered rule set

- Rules are rank ordered according to their priority.
 - An ordered rule set is known as a decision list.
- When a test record is presented to the classifier:
 - It is assigned to the class label of the highest ranked rule it has triggered.
 - If none of the rules fired, it is assigned to the default class.

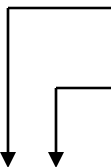
R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians



Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

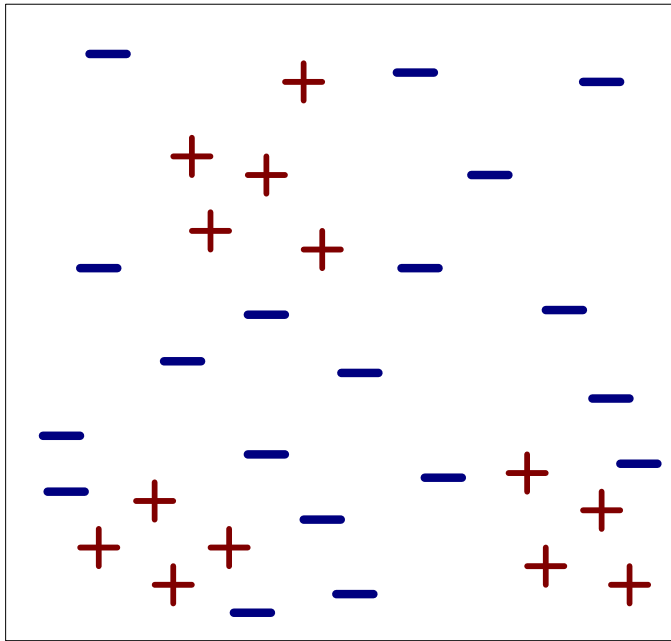
Building classification rules

- Direct method:
 - Extract rules directly from the data.
 - E.g., RIPPER, CN2, Holte's 1R.
- Indirect method:
 - Extract rules from other classification models (e.g., decision trees, neural networks, etc).
 - e.g., C4.5rules.

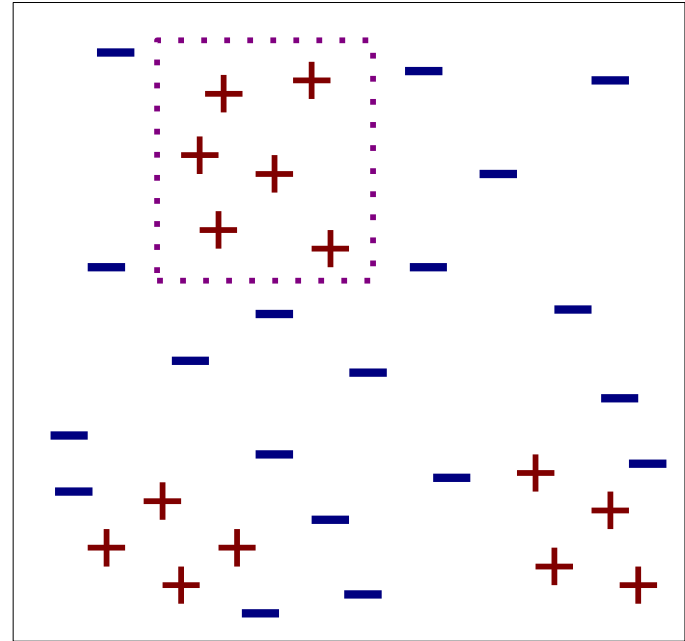
Direct method: Sequential covering

1. Start from an empty rule.
2. Grow a rule using a “Learn-One-Rule” function.
3. Remove training records covered by the rule.
4. Repeat step (2) and (3) until stopping criterion is met.

Example of sequential covering

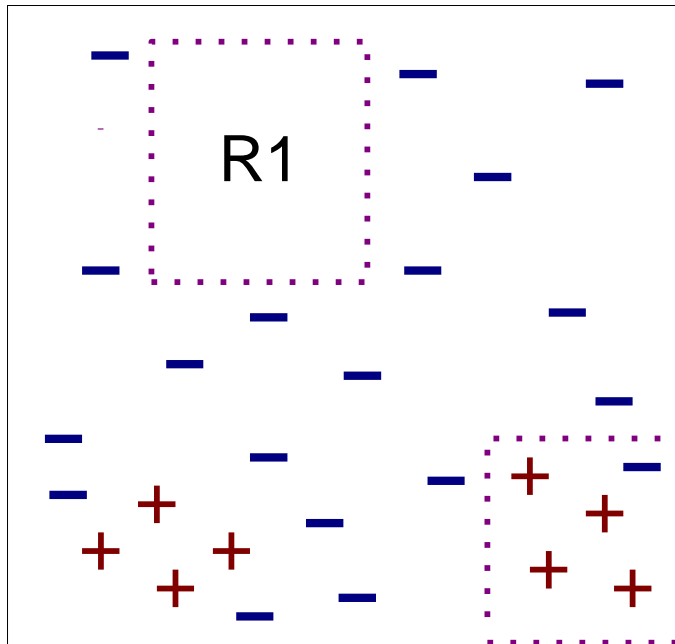


(i) Original Data

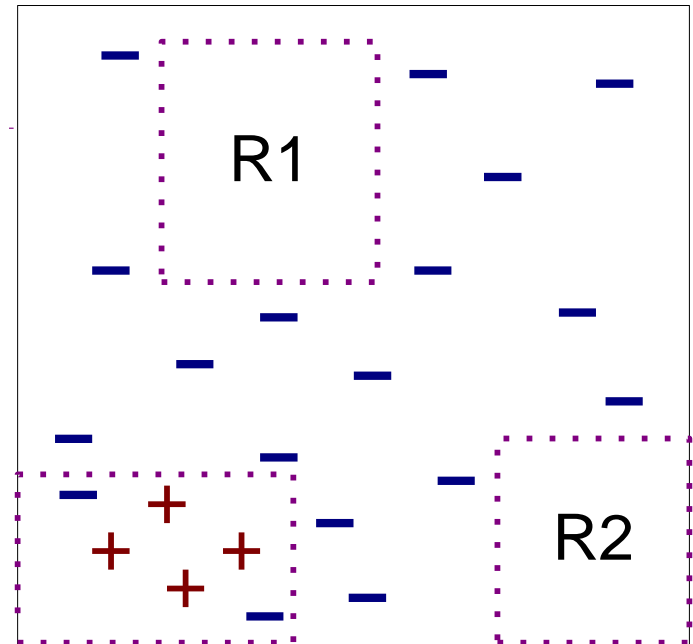


(ii) Step 1

Example of sequential covering...



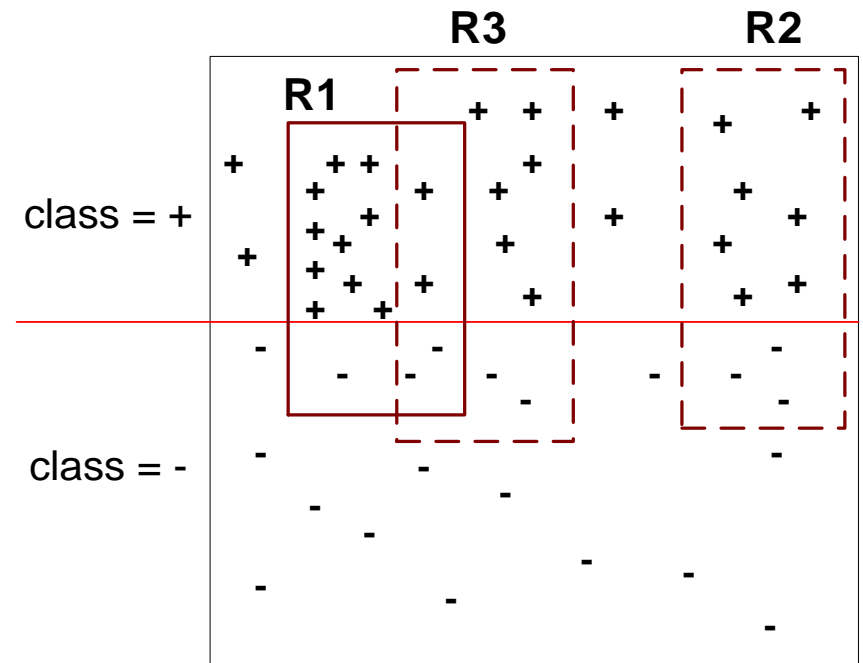
(iii) Step 2



(iv) Step 3

Instance elimination

- Why do we need to eliminate instances?
 - Ensure that the next rule is different.
- Why do we remove positive instances?
 - Ensure that the next rule is different.
- Why do we remove negative instances?
 - Prevent underestimating accuracy of rule.

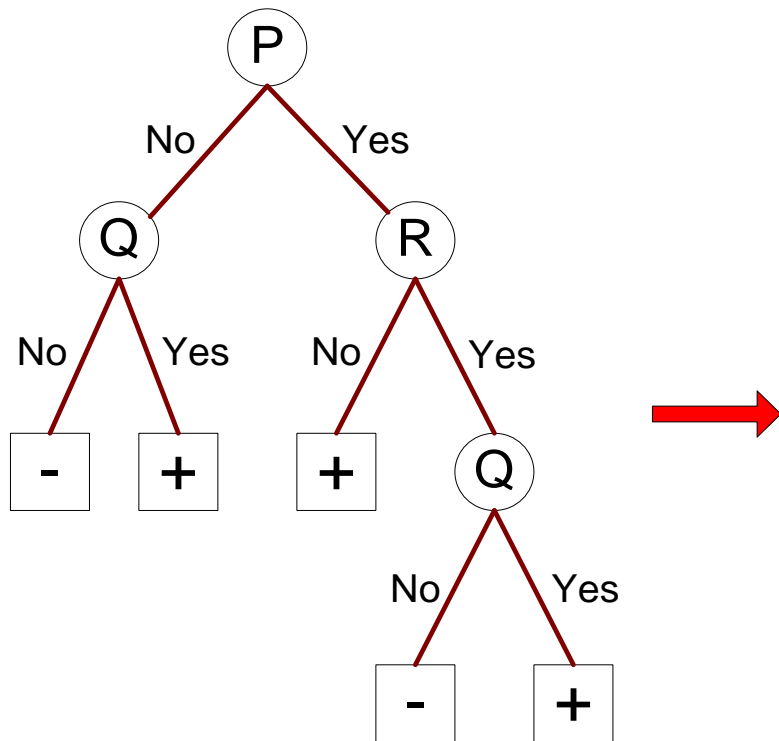


Direct method: RIPPER

- For 2-class problem, choose one of the classes as positive class, and the other as negative class.
 - Learn rules for the positive class.
 - Negative class will be default class.
- For multi-class problem:
 - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class).
 - Learn the rule set for smallest class first, treat the rest as negative class.
 - Repeat with next smallest class as positive class.

Indirect method: C4.5 rules

- Extract rules from an unpruned decision tree.



Rule Set

r1: (P=No,Q=No) ==> -

r2: (P=No,Q=Yes) ==> +

r3: (P=Yes,R=No) ==> +

r4: (P=Yes,R=Yes,Q=No) ==> -

r5: (P=Yes,R=Yes,Q=Yes) ==> +

Advantages of rule-based classifiers

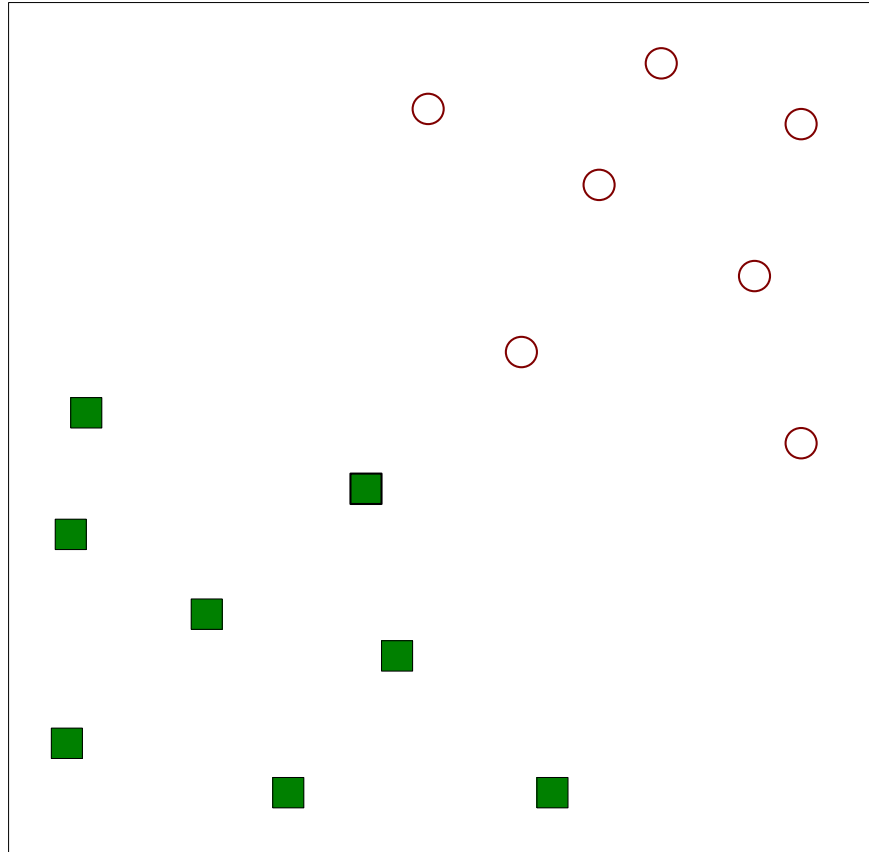
- Has characteristics quite similar to decision trees:
 - As highly expressive as decision trees.
 - Easy to interpret.
 - Performance comparable to decision trees.
 - Can handle redundant attributes.
- Better suited for handling imbalanced classes.
- Harder to handle missing values in the test set.

Drawbacks of rule-based classifiers

- Classical algorithms provide no guarantees of globally optimal solution:
 - They are not guaranteed to find the best rule set and its associated ordering.
- Rule induction is done using a greedy approach.
 - Pruning methods try to address some of the greedy nature, but not completely.
- Rule ordering needs to consider all possible permutations in order to find an optimal solution.

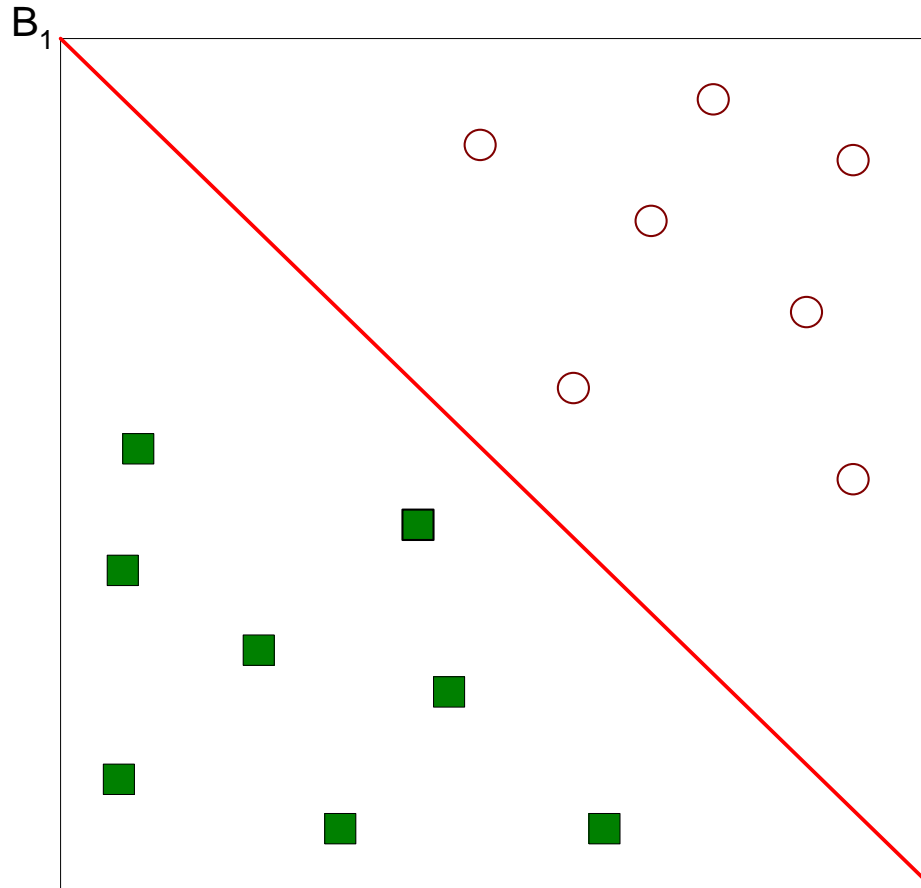
SUPPORT VECTOR MACHINES

Support Vector Machines (SVM)



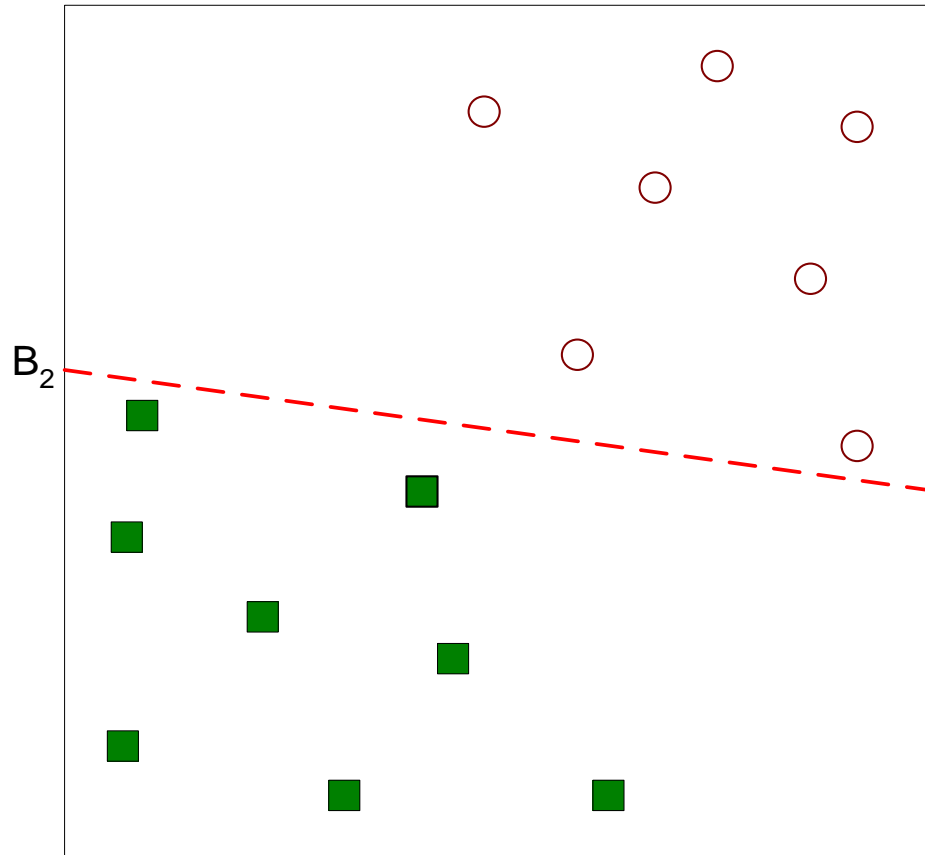
Find a linear hyperplane (decision boundary) that separates the data.

Support Vector Machines



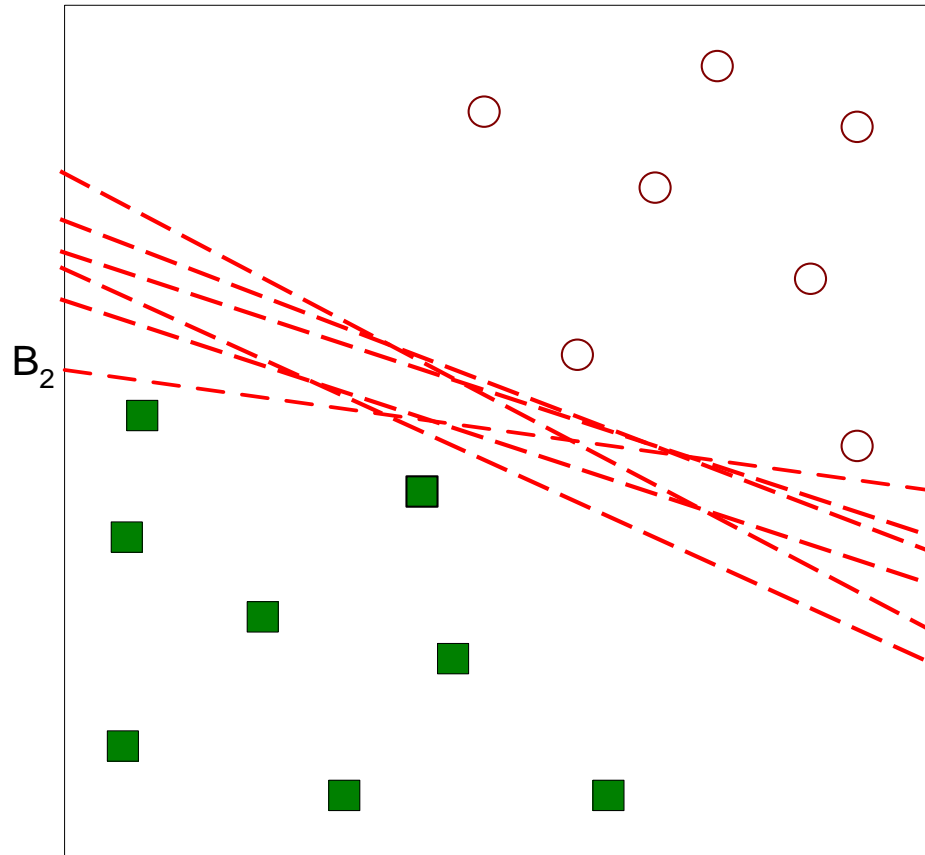
One possible solution.

Support Vector Machines



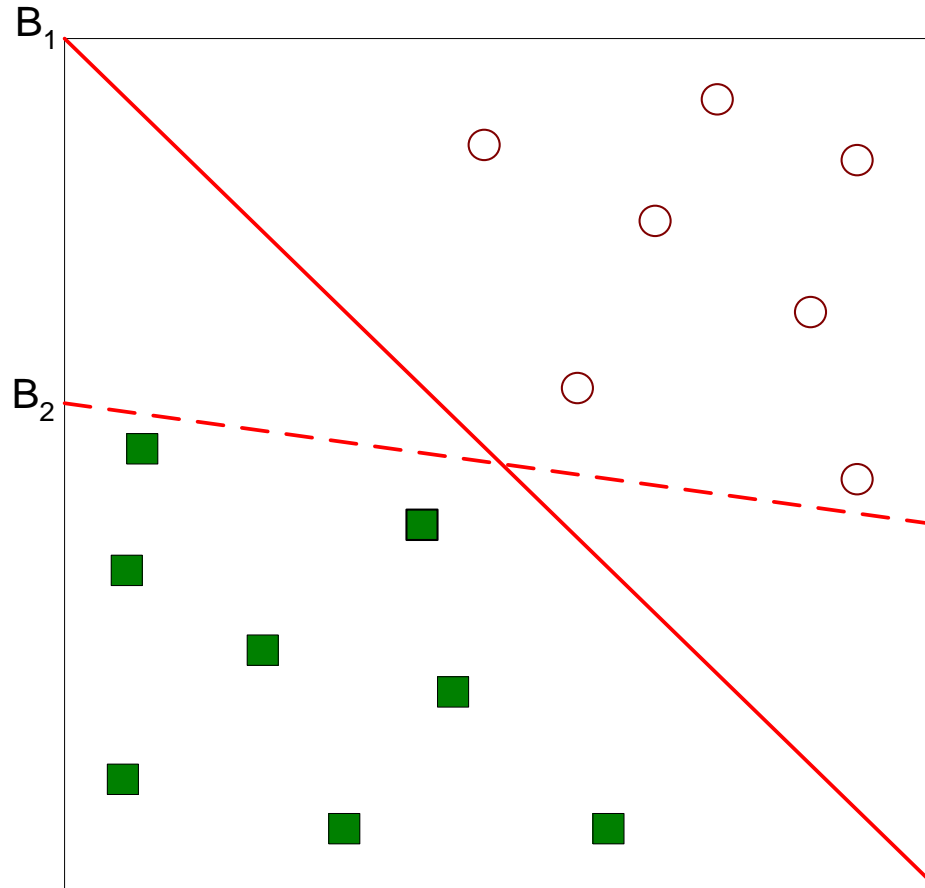
Another possible solution.

Support Vector Machines



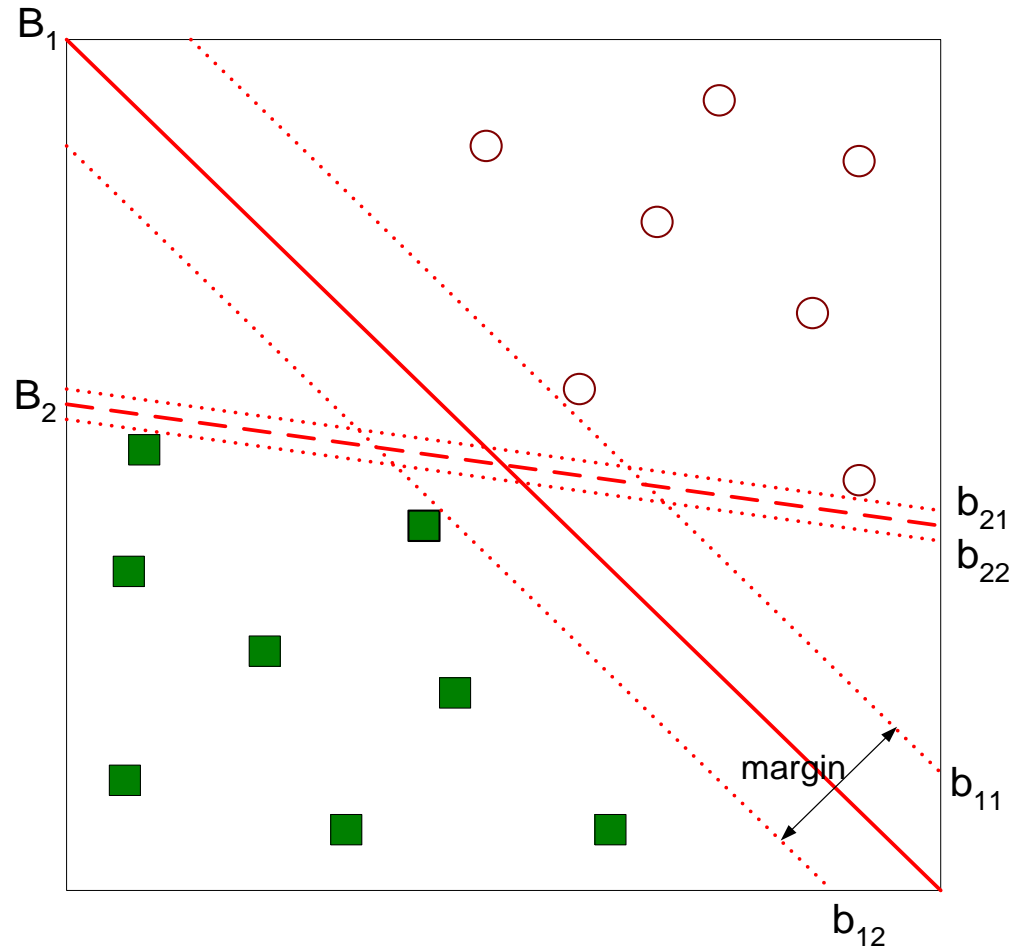
Other possible solutions.

Support Vector Machines



- Which one is better? B_1 or B_2 ?
- How do we define better?

Support Vector Machines



Find the hyperplane that **maximizes** the margin \Rightarrow B1 is better than B2.

Support Vector Machines

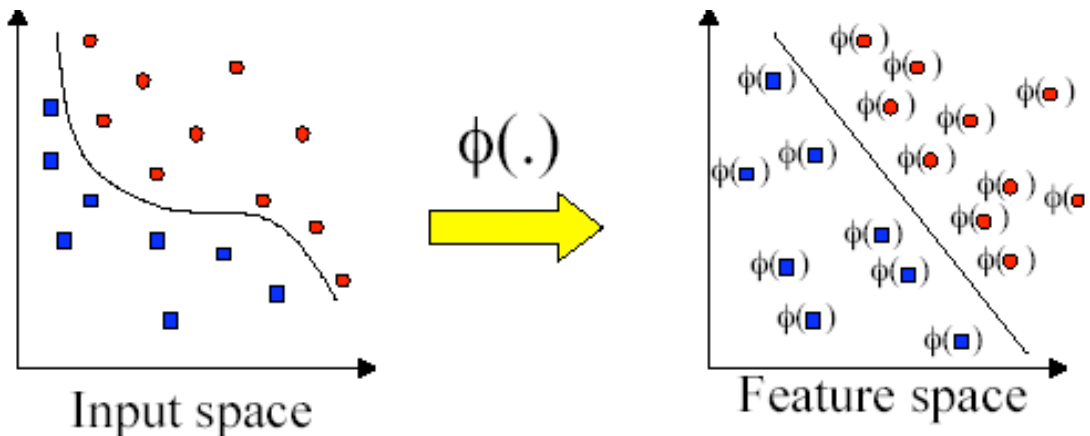
- A widely used machine learning tool that learns a maximum margin linear binary classifier.
- The classification hyperplane is defined in terms of the support vectors.
- It can deal with non-separable classes via slack variables.
- It can find non-linear hyperplanes via the use of the kernel-trick.

$$f(x) = \sum_{i \in SV} y_i \alpha_i \langle x_i \cdot x \rangle + b$$



Kernel-based
formulation

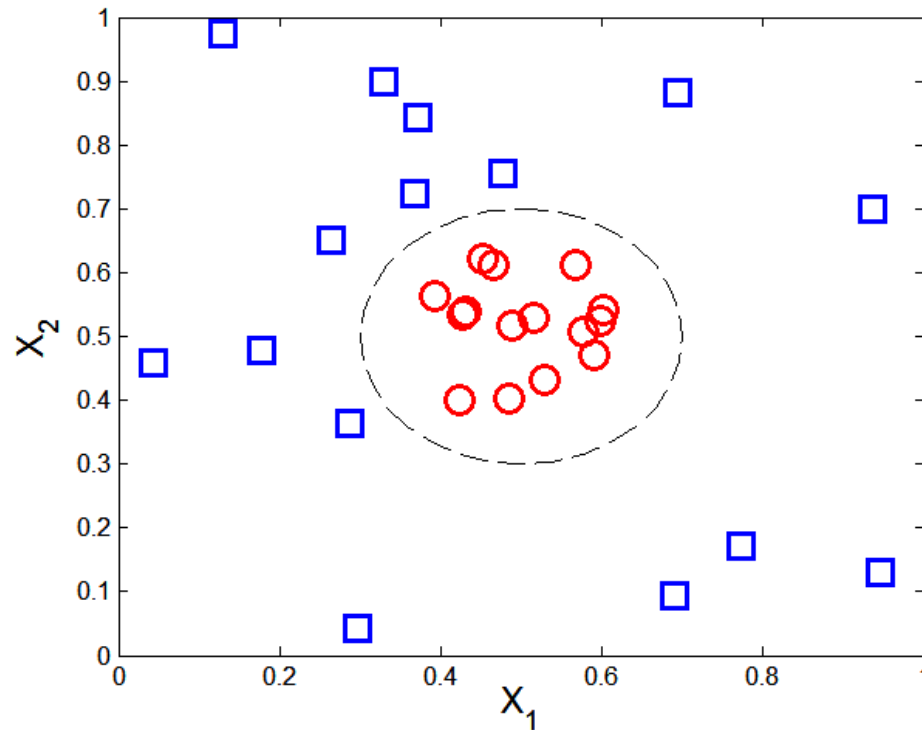
$$f(x) = \sum_{i \in SV} y_i \alpha_i K(x_i, x) + b$$



The kernel function can be considered as a measure of similarity between objects and used to encode key information about the classification problem.

Nonlinear support vector machines

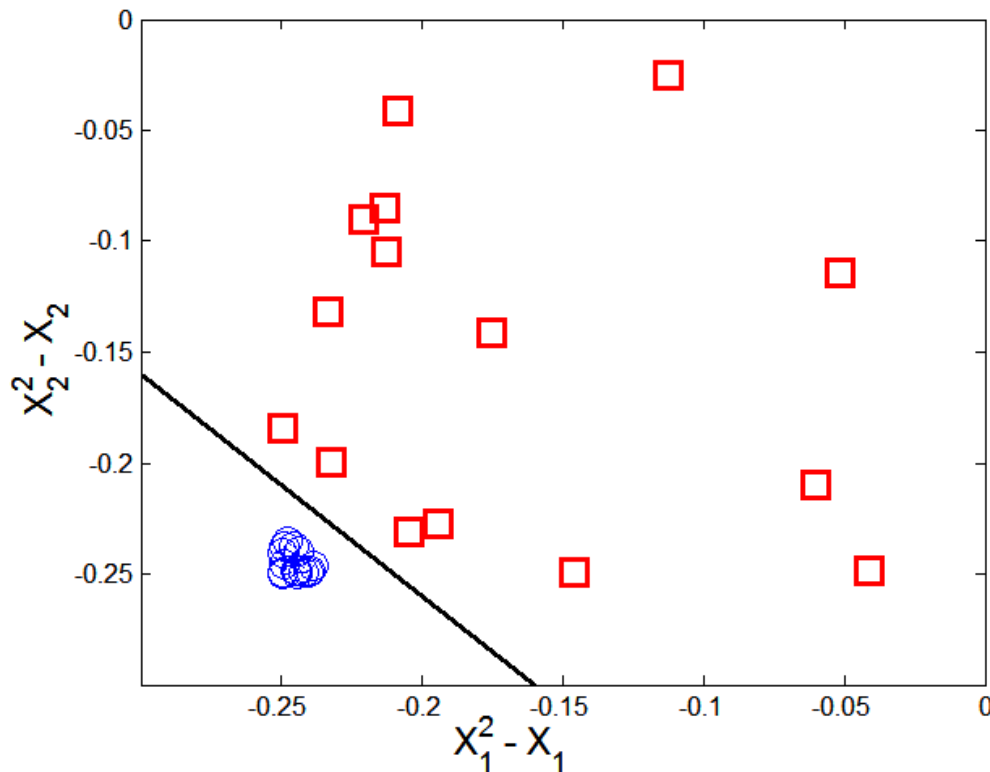
What if decision boundary is not linear?



$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

Nonlinear support vector machines

Transform data into higher dimensional space.



$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Decision boundary:

$$\vec{w} \bullet \Phi(\vec{x}) + b = 0$$

Learning nonlinear SVM

- Advantages of using kernels:
 - Don't have to know the mapping function Φ .
 - Computing dot product $\Phi(x_i) \bullet \Phi(x_j)$ in the original space avoids curse of dimensionality.
- Not all functions can be kernels:
 - Must make sure there is a corresponding Φ in some high-dimensional space.
 - Mercer's theorem (kernel function must be positive-definite). E.g.,
 - $K(x, y) = (\langle x, y \rangle + 1)^p$
 - $K(x, y) = e^{-\|x-y\|^2/(2\sigma^2)}$

Characteristics of SVM

- The learning problem is formulated as a convex optimization problem, efficient algorithms are available to find the global minima of the objective function.
- Over-fitting is addressed by maximizing the margin of the decision boundary, but the user still needs to provide the type of kernel function and cost function.
- Difficult to handle missing values.
- Robust to noise.
- High computational complexity for building the model.

SVM Summary

- Strength

- Very accurate
 - Probably the most accurate algorithm for text classification

- Weakness

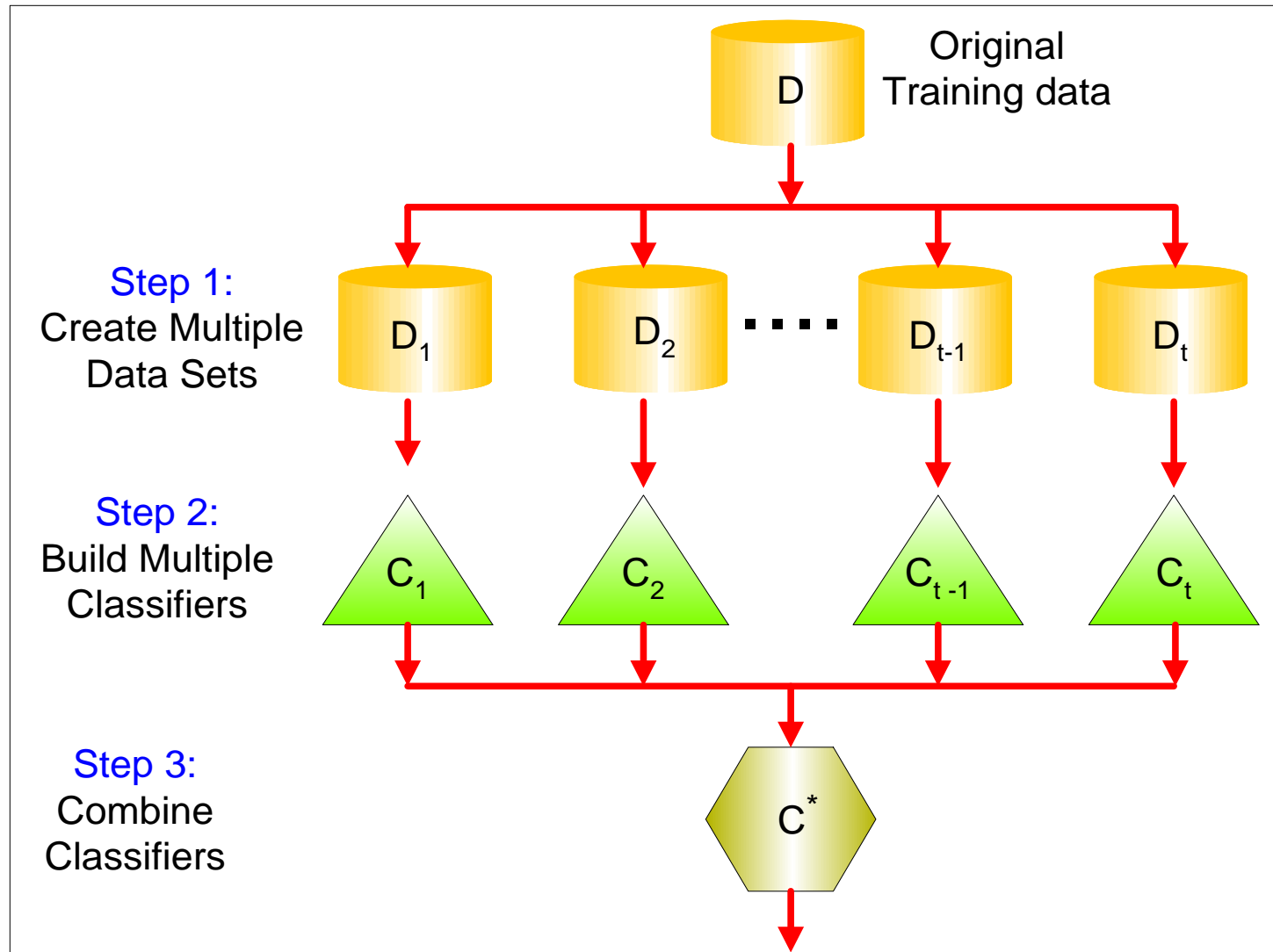
- Only works with numeric attributes
 - Use binary values for categorical attributes (appears/not)
- Slow training
- The hyperplane is difficult to understand by users

ENSEMBLE METHODS

Ensemble methods

- So far, we have only discussed individual classifiers, i.e., how to build them (one model) and use them.
- However, some algorithms suffer from *high variance*
 - Different training data might result in very different model
 - Examples?
- Can we combine multiple classifiers (i.e. models) to produce a better classifier?
 - We discuss 3 main procedures:
 - Bagging
 - Boosting
 - Random Forests

General approach



Types of ensemble methods

- Manipulate data distribution.
 - Resampling method.
 - Bagging and boosting.
- Manipulate input features.
 - Feature subset selection.
 - Random forest: Randomly select feature subsets and built decision trees.
- Manipulate class labels.
 - Randomly partition the classes into two subsets, treat them as +ve and –ve, and learn a binary classifier. Do that many times. At classification, use all binary classifiers and give credits to the constituent classes.
- By using different models.
 - E.g., Different ANN topologies.

Bagging

Bootstrap Aggregating = Bagging [Breiman, 1996]

□ Application of bootstrap sampling

- **Given:** set D containing n training examples
- Create k samples $S[i]$ of D by drawing n examples at random *with replacement* from D
 - This is called *bootstrapping*
- Build a classifier on each bootstrap sample.
- Use a majority voting prediction approach:
 - Predict an unlabeled instance using all classifiers and return the most frequently predicted class as the prediction.

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

Bagging

- Bagging provides a natural way to reduce the variance and hence increase the prediction accuracy.
- When does it help?
 - When learner is unstable
 - Small change to training set causes large change in the output classifier
 - True for decision trees, neural networks; not true for k -nearest neighbor, naïve Bayesian, class association rules
 - Experimentally, bagging can help substantially for unstable learners, may somewhat degrade results for stable learners

Bagging & decision trees

- Create 100s-1000s deep trees (no pruning)
- Final class based on majority vote
- Large number of sample k does not lead to overfitting!

Random Forests

- A tweak on bagging for decision trees that *decorrelates* the trees.
- Instead of considering all m attributes at each split, only k attributes are considered
 - Typically $k \approx \sqrt{m}$
 - How does this help?

Random Forests

- If data has a very strong predictor (attribute), this will always be the one used in the top split
- Consequently, all bagged trees will look quite similar to each other and predictions will be highly correlated.
 - Not as high reduction in variance of model
- With random forests, on average $(m - k)/m$ of the splits will not even consider the strong predictor!
 - Using small value of k is helpful when we have a large number of correlated predictors.

Bagging (for DT) vs. Random Forests

- What happens when $m=k$ in a random forest?

Boosting

- A family of methods:
 - We only study **AdaBoost** (Freund & Schapire, 1996)
- **Training**
 - Produce a sequence of classifiers (the same base learner)
 - No bootstrapping!
 - Each classifier is *dependent* on the previous one, and focuses on the previous one's errors
 - Examples that are incorrectly predicted in previous classifiers are given higher weights
- **Testing**
 - For a test case, the results of the series of classifiers are combined to determine the final class of the test case.

Boosting

- Records that are wrongly classified will have their weights increased.
- Records that are classified correctly will have their weights decreased.

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

Example 4 is hard to classify.

Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds.

AdaBoost

Weighted training set

(x_1, y_1, w_1)

(x_2, y_2, w_2)

...

(x_n, y_n, w_n)

Non-negative weights
sum to 1



called a weaker classifier



Build a classifier h_t whose accuracy on training set $> \frac{1}{2}$ (better than random)



Change weights

Does AdaBoost always work?

- The actual performance of boosting depends on the data and the base learner.
 - It requires the base learner to be unstable as in bagging.
- Boosting seems to be susceptible to noise.
 - When the number of outliers is very large, the emphasis placed on the hard examples can hurt the performance.
- Large number of models (k) might lead to overfitting
 - Should evaluate via cross-validation

REGRESSION

Regression vs. Classification

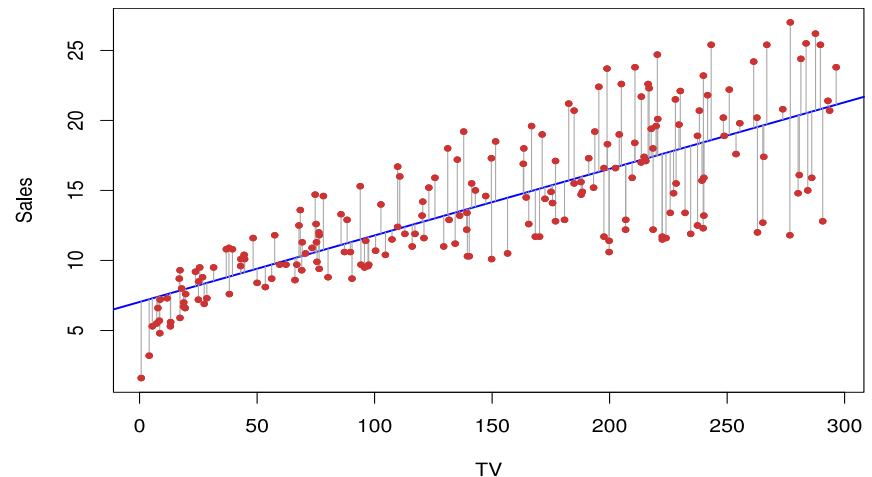
- Supervised learning
- Depends on the type of Y
- If Y is continuous/numerical:
 - Regression problem
 - E.g. predict someone's wage based on age, education etc.
 - Overview of Linear and kNN-based Regression
- If Y is categorical:
 - Classification problem
 - E.g. predict whether a loan application will be approved or not.
 - Predictor Y is called the **class**
 - We focused most of our time on Classification

Linear Regression

- Very simple supervised learning method
- Assumes there's a **linear relationship between X and Y**
- Old, but still widely used
 - (sometimes simpler is better)
- Simple Linear Regression
 - One predictor, one response
- Multiple Linear Regression
 - Many predictors, one response

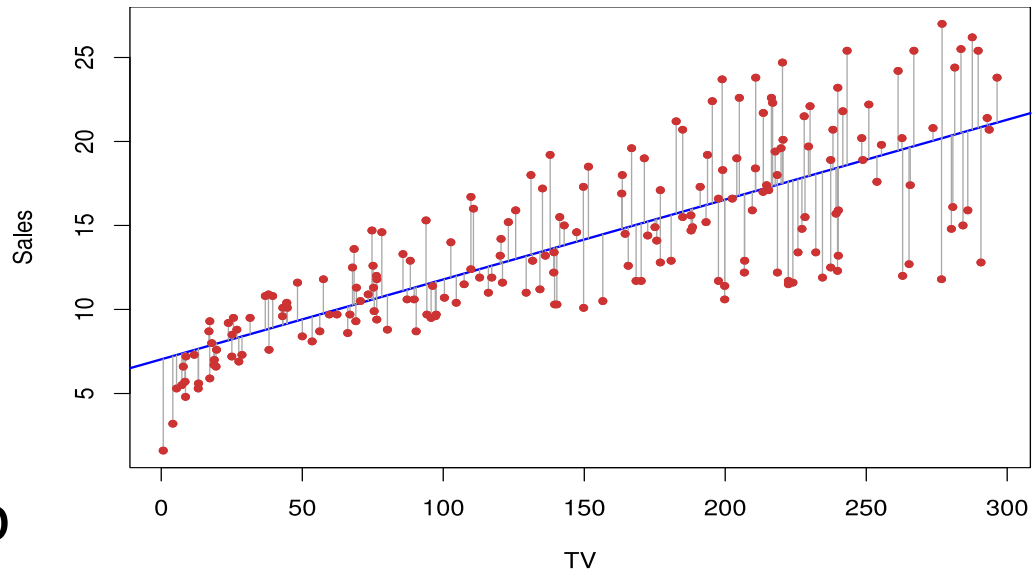
Linear Regression Task

- We are given a collection of records (training set)
 - Each record is characterized by a tuple (x, y) , where x is a set of numerical attributes and y is a value.
- Goal:
 - We want to learn a vector β such that $\langle x, \beta \rangle$ approximates y in a least squares sense.



Simple Linear Regression

- $y \approx \beta_0 + \beta_1 x_1$
- β_0 : intercept
- β_1 : slope
- Use **training data** to estimate intercept and slope
- Apply to new data (x) to make predictions (y)



e.g. What will be the sales if we spend \$150K on TV advertising?

$$\text{sales} \approx \beta_0 + \beta_1 \text{TV}$$

Simple Linear Regression

Given $\hat{y}_i = \beta_0 + \beta_1 x_i$,

$$\min_{\beta_0, \beta_1} \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- Exact solution for SLR

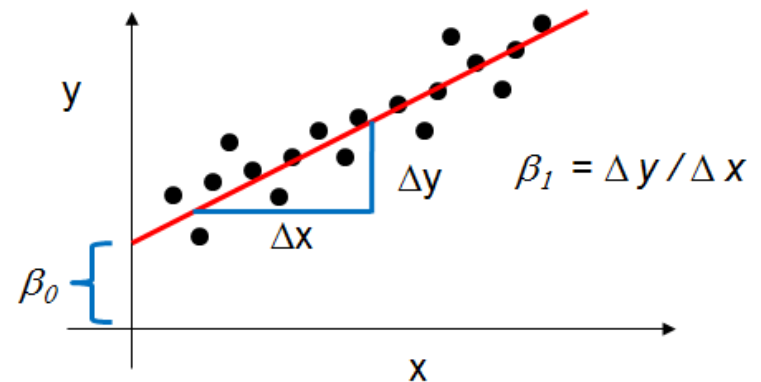
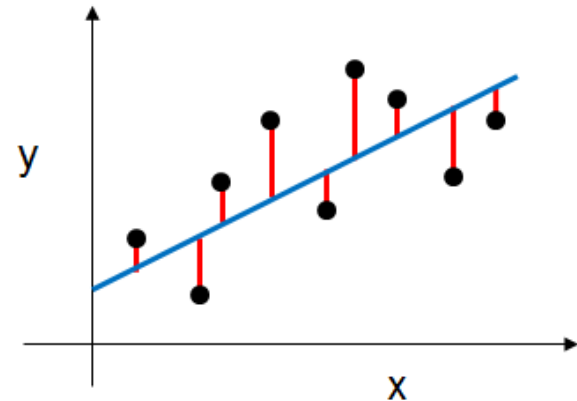
- Let $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$, $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$

- β_0 is the **intercept**

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

- β_1 is the **slope**

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}$$



Simple Linear Regression

- Given the table below, find the parameters of the linear regression model

X	Y
12.4	11.2
14.3	12.5
14.5	12.7
14.9	13.1
16.1	14.1
16.9	14.8
16.5	14.4
15.4	13.4
17.0	14.9
17.9	15.6
18.8	16.4
20.3	17.7
22.4	19.6
19.4	16.9
15.5	14.0
16.7	14.6
17.3	15.1
18.4	16.1
19.2	16.8
17.4	15.2
19.5	17.0
19.7	17.2
21.2	18.6

- Compute sums

$$\sum x_i = 401.7 \quad \sum y_i = 351.9$$

$$\sum x_i y_i = 6255.12 \quad \sum x_i^2 = 7143.97$$

- Compute betas

$$\beta_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} = 0.851144$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x} = \frac{1}{n} (\sum y_i - \beta_1 \sum x_i) = 0.434584$$

- Final regression equation

$$y = 0.434584 + 0.851144x$$

Simple Linear Regression

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import linear_model

# Get data
df = pd.read_csv(
    filepath_or_buffer='data/trucks.csv',
    header=None)
data = df.ix[:,:].values
X = data[:,0].reshape(-1, 1)
Y = data[:,1].reshape(-1, 1)

# Train the model using the training sets
regr = linear_model.LinearRegression()
regr.fit(X, Y)
slope = regr.coef_[0][0]
intercept = regr.intercept_

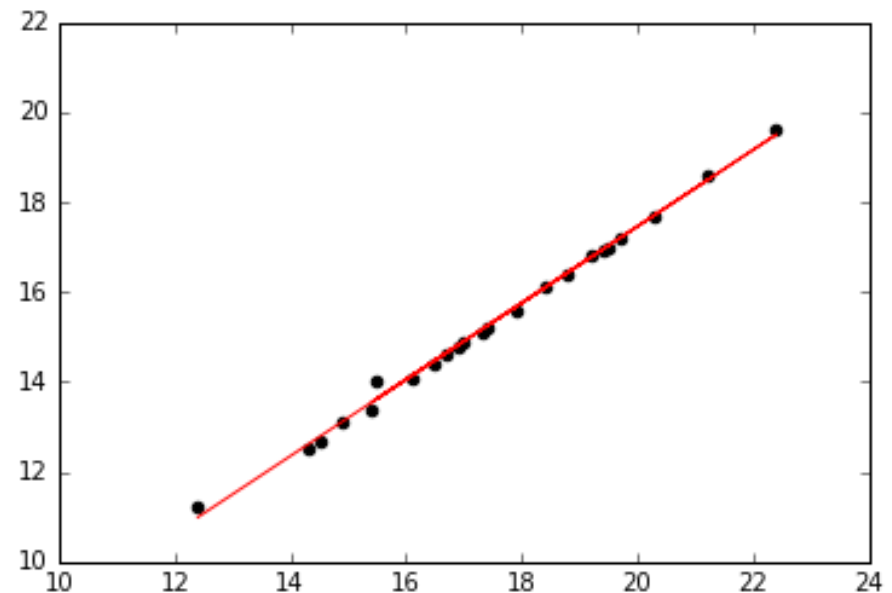
print("y = %f + %fx" %(intercept, slope))
print("Mean squared error: %f"
      % np.mean((regr.predict(X) - Y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %f' % regr.score(X, Y))

# Plot outputs
plt.scatter(X, Y, color='black')
plt.plot(X, regr.predict(X), color='red',
         linewidth=1)
plt.show()
```

$$y = 0.434585 + 0.851144x$$

Mean squared error: 0.011812

Variance score: 0.997083



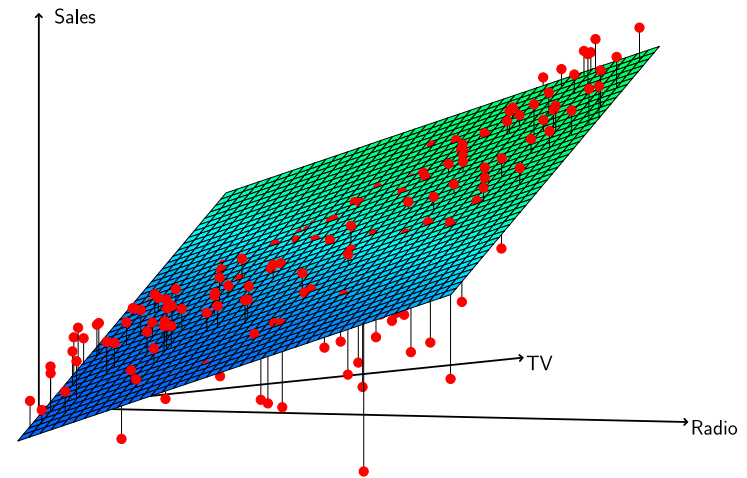
Multiple Linear regression

- What if we spend \$\$ on TV, radio and newspaper ads?

- Problem now has more coefficients:

$$\text{sales} \approx \beta_0 + \beta_1 \text{TV} + \beta_2 \text{radio} + \beta_3 \text{paper} + \varepsilon$$

- Predictor will be a hyperplane



Example with 2 predictors and one response

The Linear Regression Model

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

- β_0 is the intercept (i.e. the average value for y if all the x 's are zero), β_j is the slope for the j th variable x_j
- β_j is the average increase in y when x_j is increased by one and **all other x 's are held constant**.

Parameter Estimation

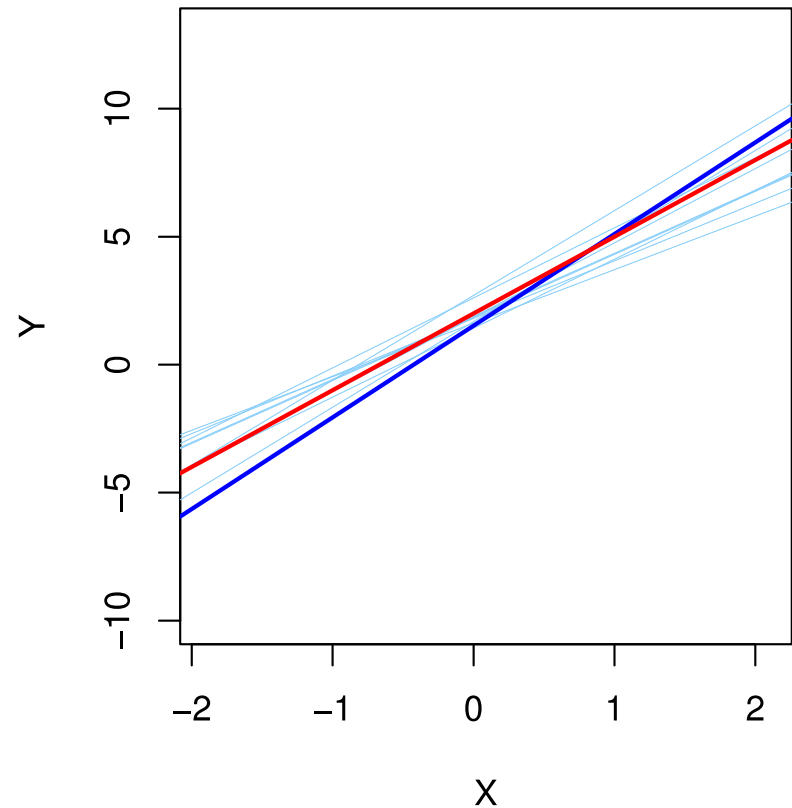
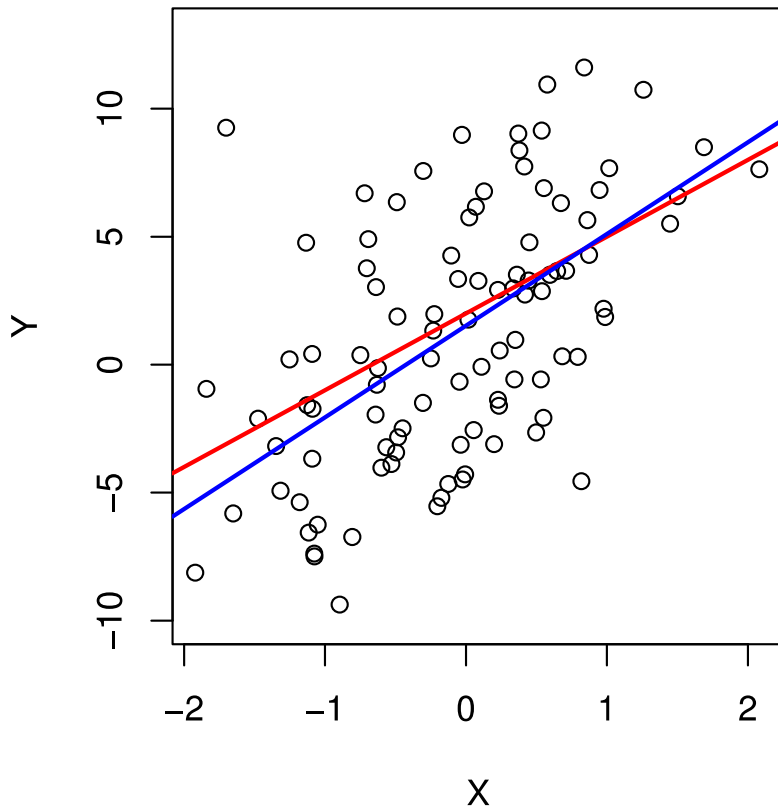
- We estimate the parameters using the least squares method to minimize **MSE (Mean Square Error)**

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_1 - \cdots - \hat{\beta}_p x_p)^2$$

* MSE/RMSE is a very popular method for estimating parameters and also evaluating accuracy of a model. We will use it again in the future.

Training vs. Real data

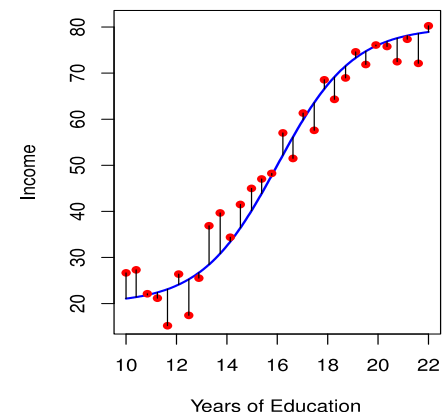
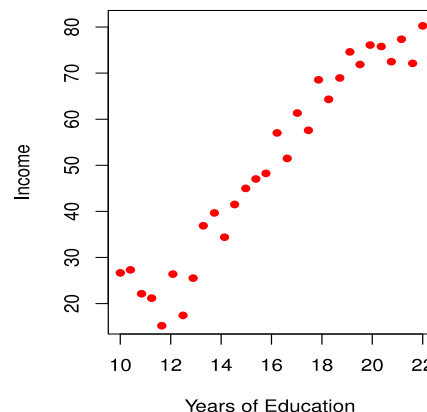
- In all data mining techniques, the input data affects (to a lesser or bigger extend) the output model.
- The key is to avoid overfitting or underfitting the training data and generate a model that accurately approximates the real data.
 - We will discuss how later...



- Red line: True (population) line - unobserved
- Blue line: Least squares line for training data
- Light blue lines: possible least square lines for different training datasets

Potential Problems

- Non-linear relationship between response-predictors
- Non-constant variance of error terms
- Collinearity
 - Two or more variables related to each other
- Correlation of error terms
- Outliers



Linear regression and normal equations

- Let \mathbf{X} be an $n \times m$ matrix whose rows are the records
- Let \mathbf{y} be the $n \times 1$ vector of the known response values, s.t. $\mathbf{y} = \langle 1, y_1, y_2, \dots, y_n \rangle$
- The solution to the linear regression problem is the vector $\boldsymbol{\beta}$ that minimizes

$$\min_{\boldsymbol{\beta}} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2$$

- The solution to the above problem is given by

$$\boldsymbol{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- However, this is not how we usually solve the problem...

Ridge Regression

- In order to prevent overfitting, we add a *regularization* penalty
- Estimate $\boldsymbol{\beta}$ as follows,

$$\min_{\boldsymbol{\beta}} (\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 + \lambda \|\boldsymbol{\beta}\|^2)$$

where λ is a user-supplied parameter that controls overfitting.

Regularization options

- In order to prevent overfitting, we add a *regularization* penalty

- Ridge regression:

$$\min_{\boldsymbol{\beta}} (\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2)$$

- Lasso regression:

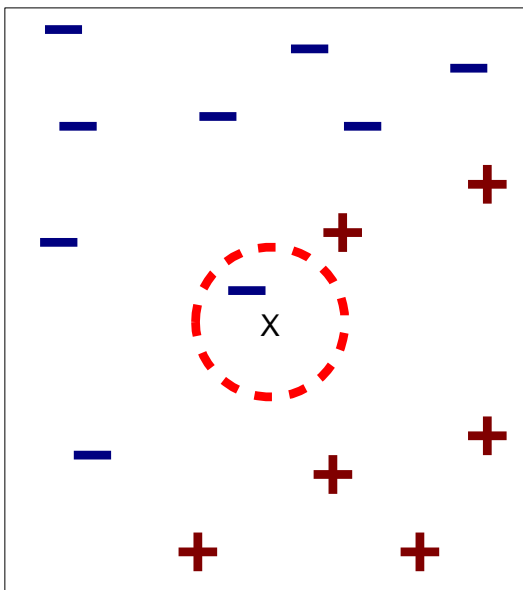
$$\min_{\boldsymbol{\beta}} (\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \gamma \|\boldsymbol{\beta}\|_1^2)$$

- Elastic Net:

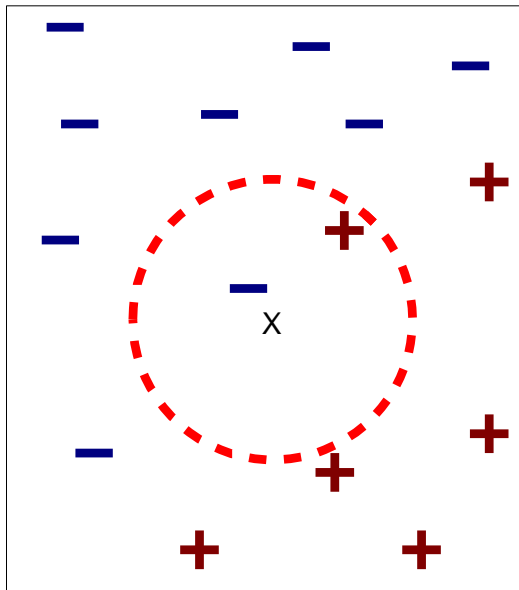
$$\min_{\boldsymbol{\beta}} (\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \gamma \|\boldsymbol{\beta}\|_1^2 + \lambda \|\boldsymbol{\beta}\|_2^2)$$

where λ and γ are user-supplied parameters that control overfitting.

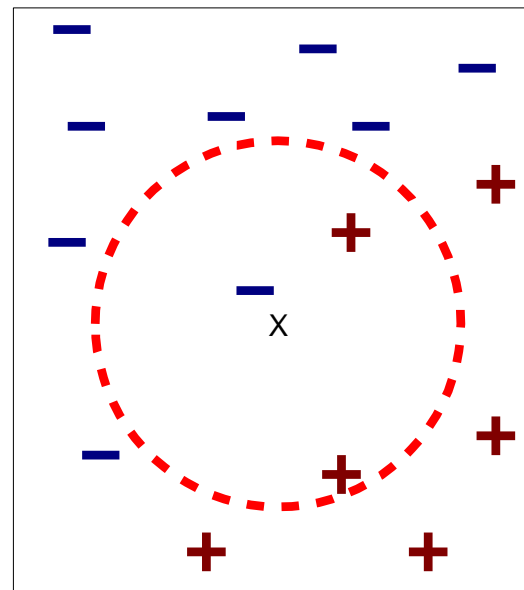
Nearest neighbor regression



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

- Average the y values of the found nearest neighbors.
- Optionally weigh the y values by the inverse of the object's distance to the target

Summary

- Applications of supervised learning are in almost any field or domain.
- There are still many more methods, e.g.,
 - Deep/Convolutional neural networks
 - Genetic algorithms
 - Fuzzy classification
 - Logistic regression

This large number of methods also show the importance of classification and its wide applicability.

- It remains an active research area.