The University of Reading

CSMDM16

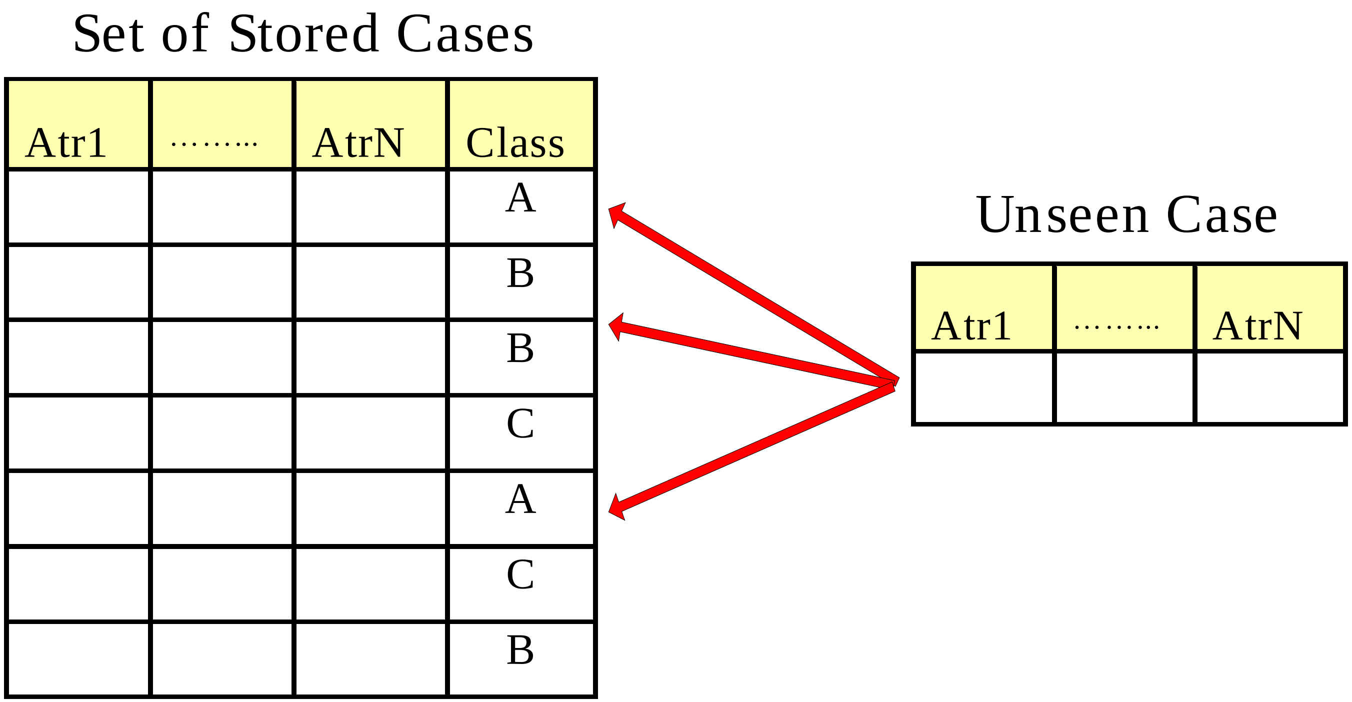
Classification and Model Evaluation

Dr. Giuseppe Di Fatta
Associate Professor
Department of Computer Science
G.DiFatta@reading.ac.uk

Overview

- Classification
  - Memory based reasoning
  - Decision Trees
  - Rule-based methods
  - Artificial Neural Networks
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
  - Classification by Regression
    - Multi-response linear regression
    - Pairwise classification
    - Logistic regression
- Model evaluation
  - Accuracy and error rate
  - Cross-validation
  - ROC

Instance-Based Classification



- Store the training records
- Use training records to predict the class label of unseen cases

Instance-based Classification

- Simplest form of learning: ***rote learning***
  - Training instances are searched for instance that most closely resembles new instance
  - Instance-based learning is *lazy* learning: the instances themselves represent the knowledge
  - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

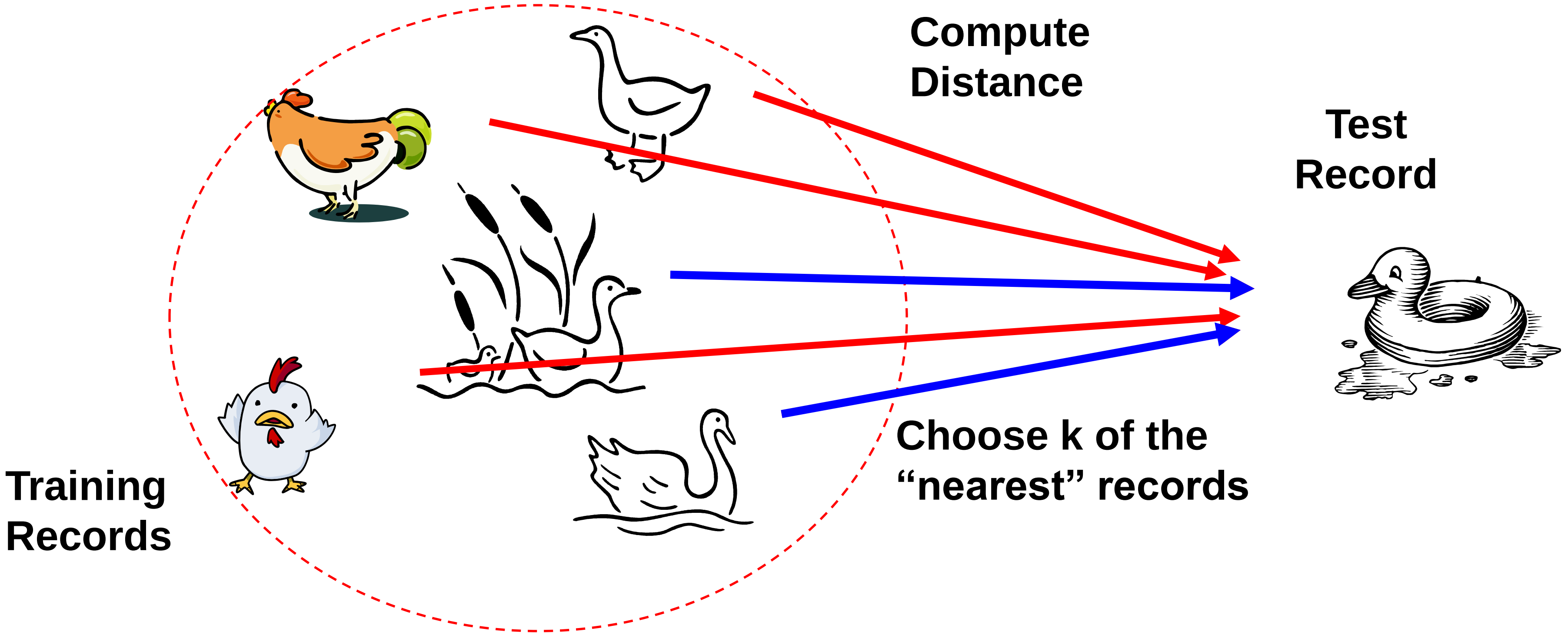
- Proximity measure (similarity/dissimilarity) defines what's "learned"

- **Nearest Neighbor**:
  - Uses "closest" points (nearest neighbors) for performing classification
    - *nearest-neighbor (1-NN)*
    - *k-nearest-neighbor (k-NN)*

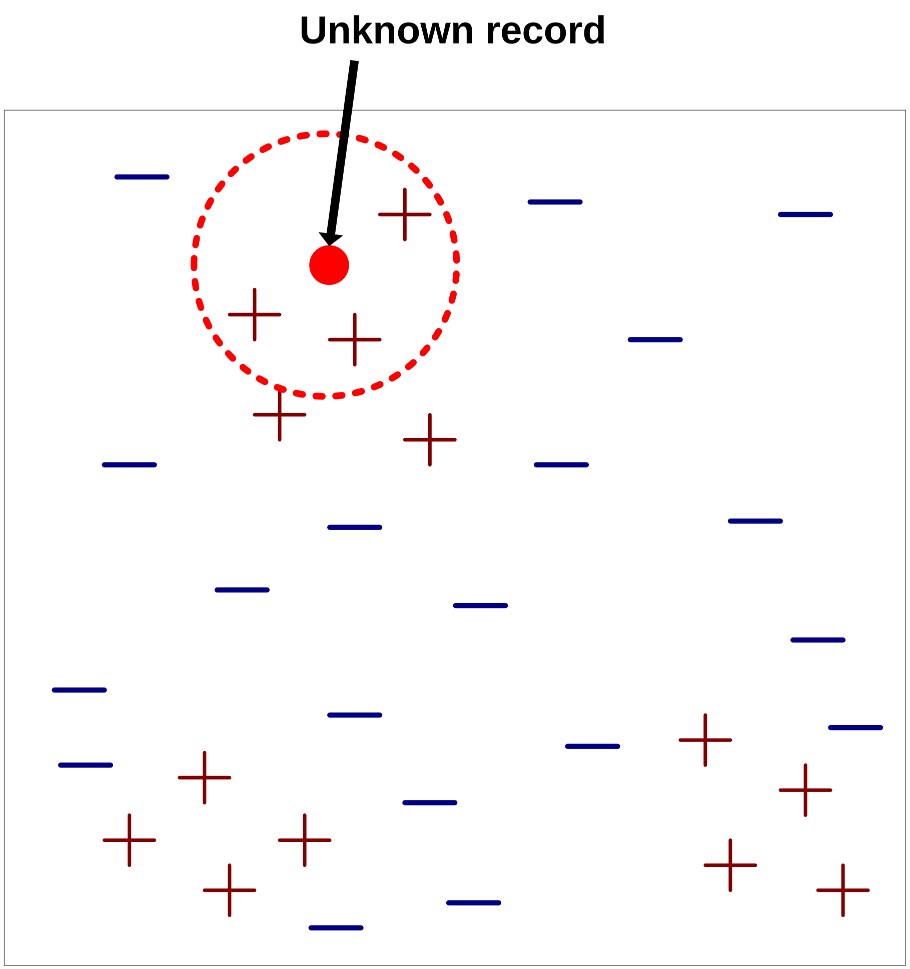
Nearest-Neighbor Classifiers

- **Nearest-Neighbor**:
  - Uses “closest” points (nearest neighbors) for performing classification

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



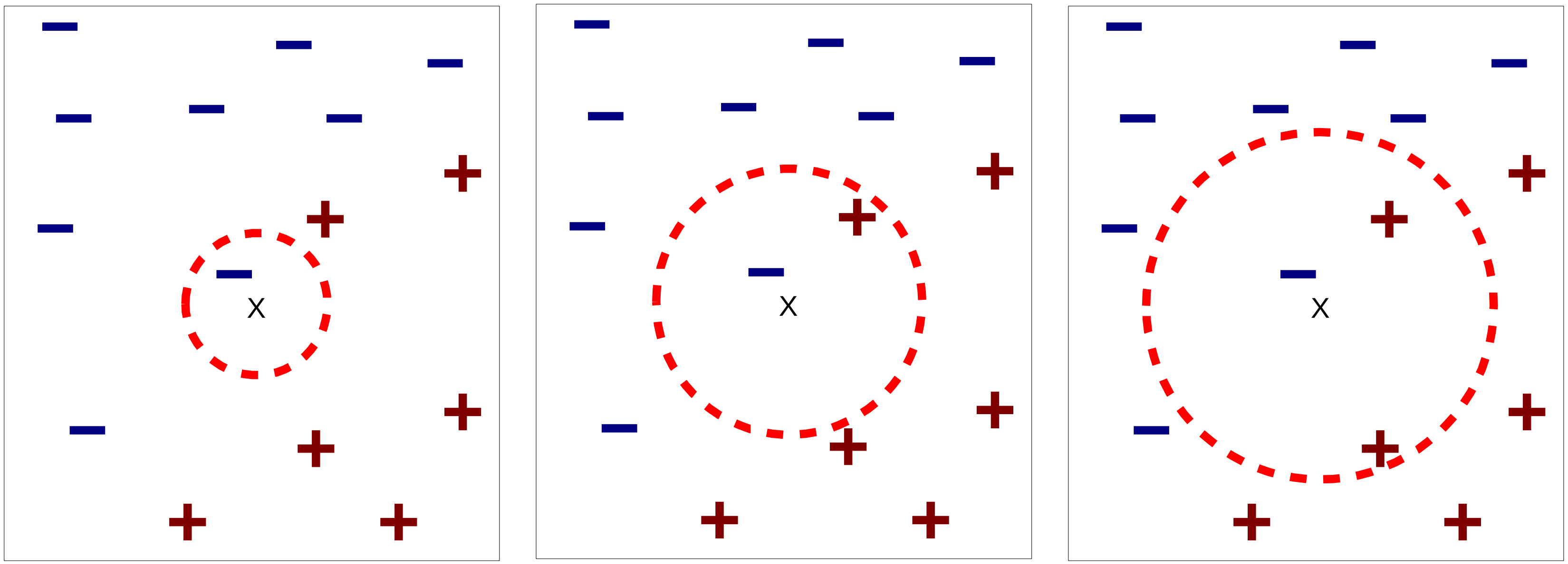
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - 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)

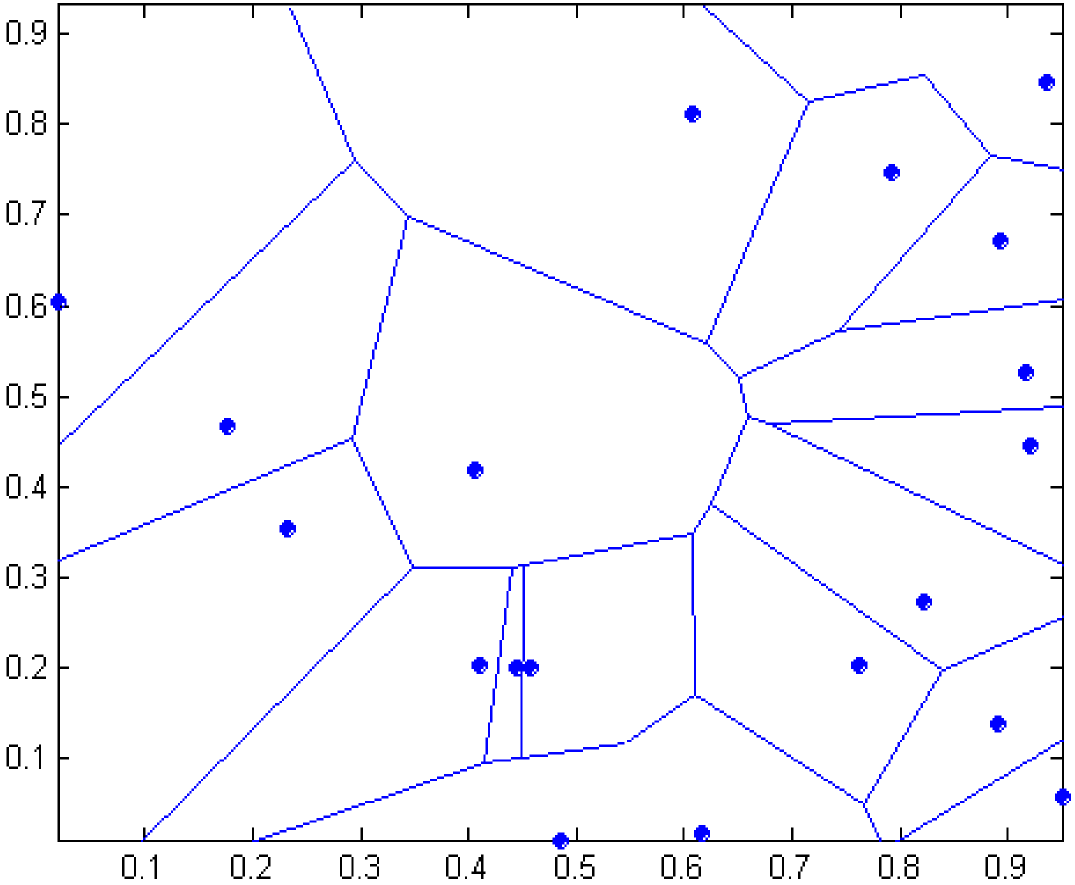
Nearest Neighbor



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

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

1-NN: Voronoi Diagram



Nearest-Neighbor Classification

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

b_i: 1 or 0 → feature [i] is/is not present

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

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

$d_E = 1.4142$

vs

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

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

$d_E = 1.4142$

Solutions:

- Normalize the vectors to unit length
- Use different metric (e.g. Tanimoto distance)

Nearest-Neighbor Classification

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records is relatively expensive

Discussion:

- Often very accurate and slow
  - simple version of 1-NN scans entire training data to derive a prediction
- Assumes all attributes are equally important
  - Remedy: attribute selection or weights
- Possible remedies against noisy instances:
  - Take a majority vote over the *k* nearest neighbors
  - Removing noisy instances from dataset (difficult!)
- Statisticians have used *k*-NN since early 1950s
  - If $n \to \infty$ and $k/n \to 0$, error approaches minimum

When to Consider NN Classifiers

- Instances map to points in R^N
- Less than 20 attributes per instance
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Do not loose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

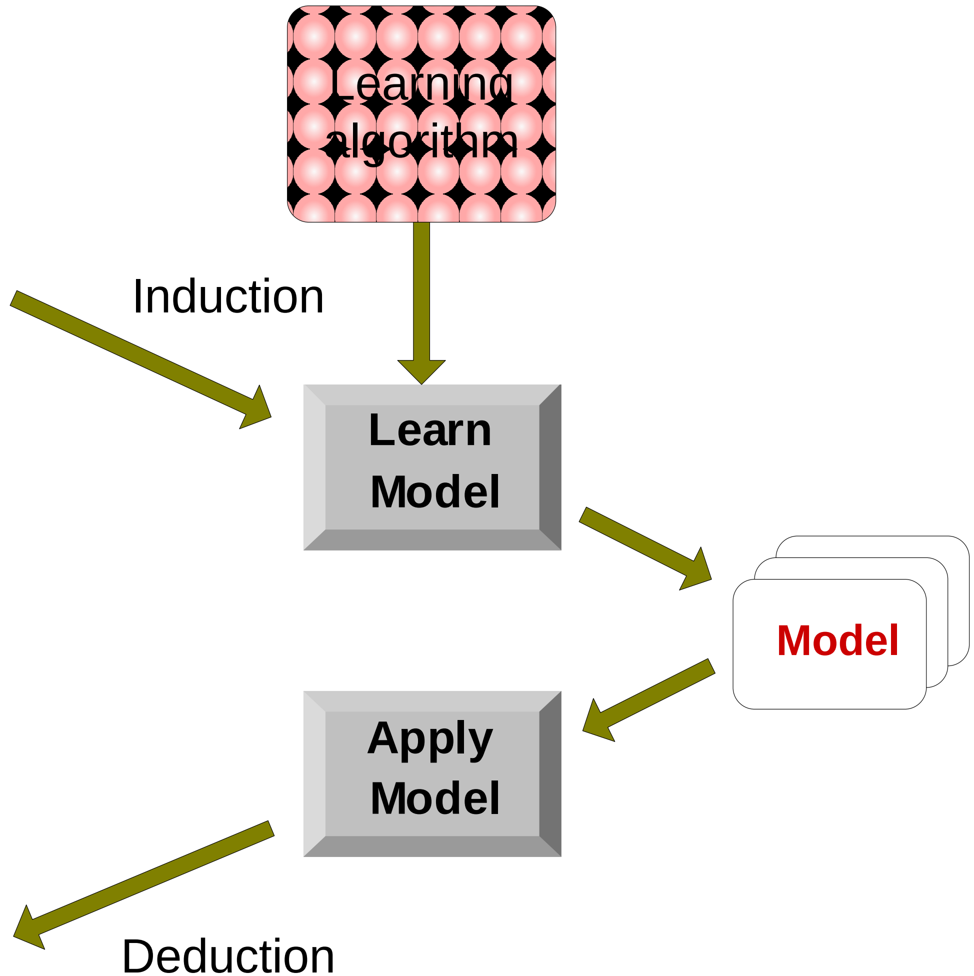
Classification: Learning Predictive Models

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

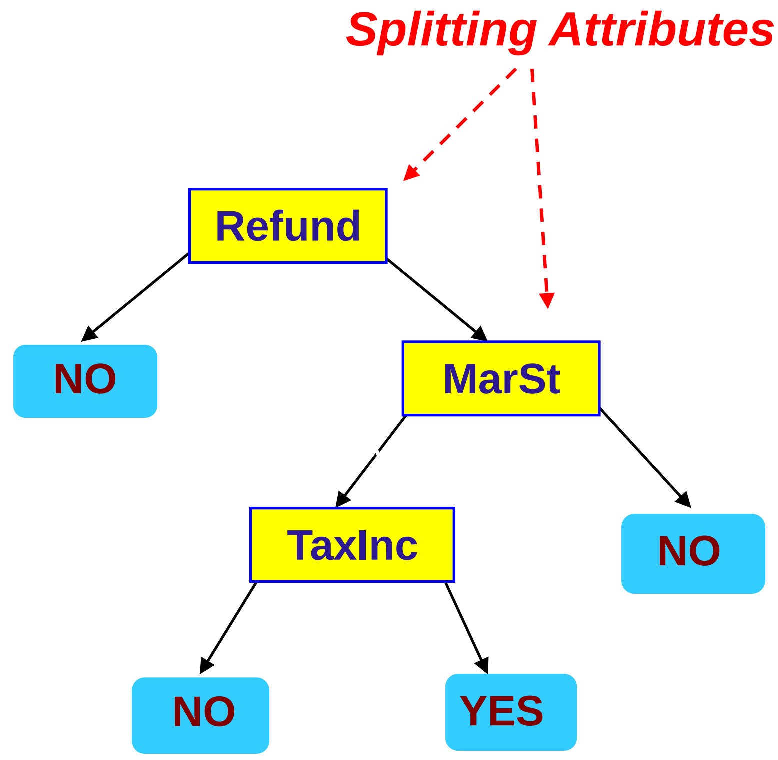


Example of a Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
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

Training Data

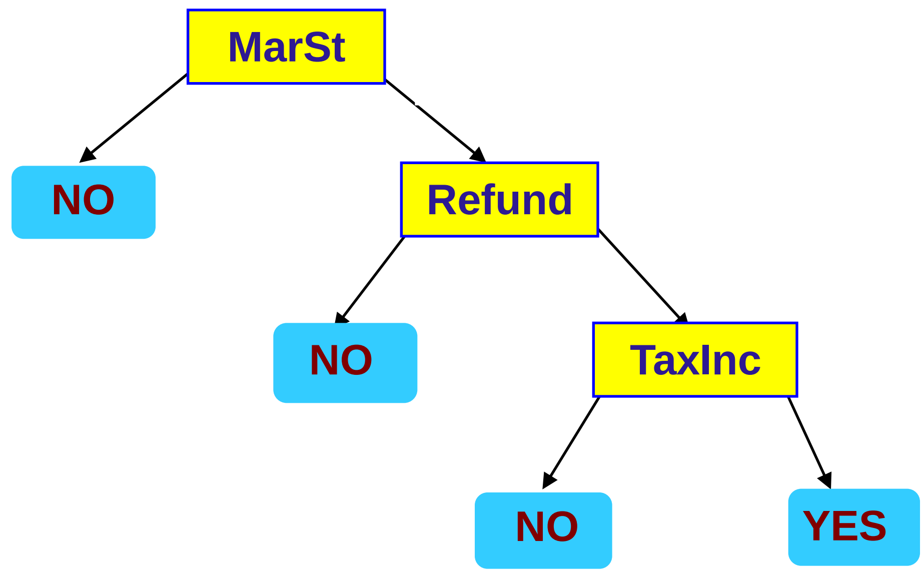


Model: Decision Tree

Another Example of Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
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

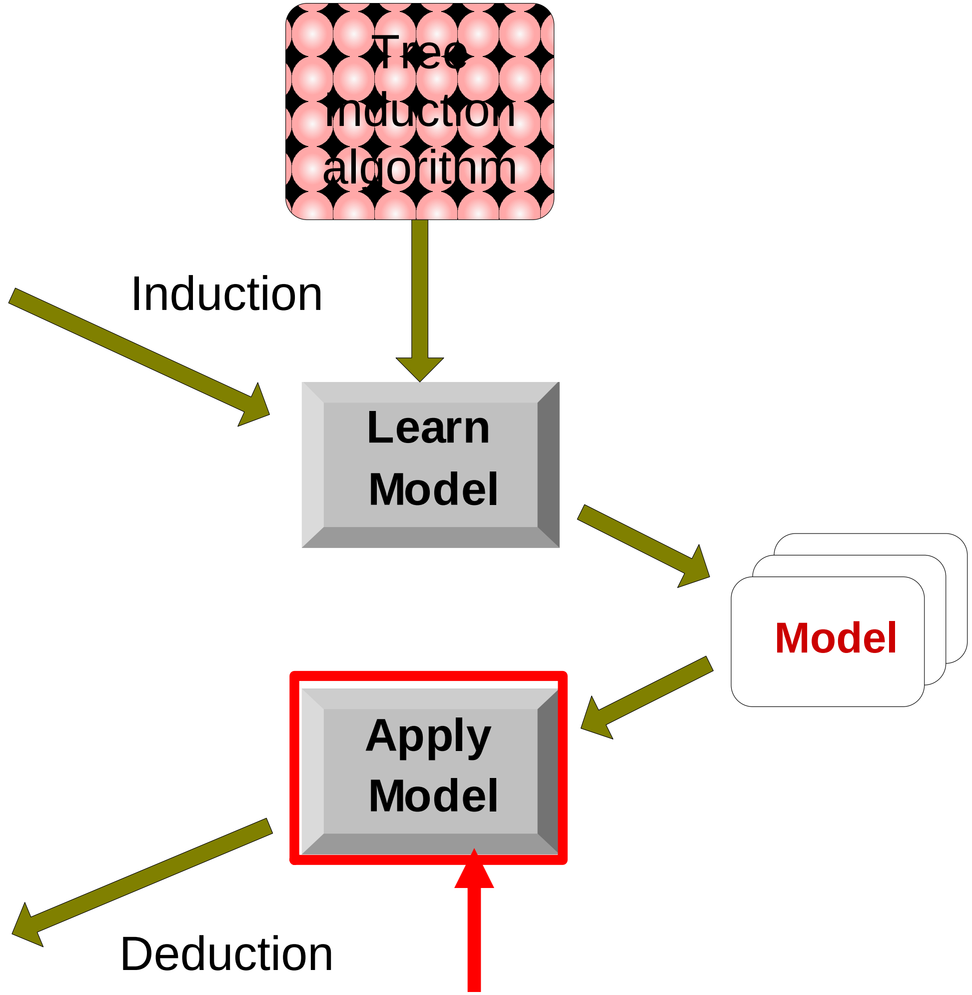


There could be more than one tree that fits the same data!

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

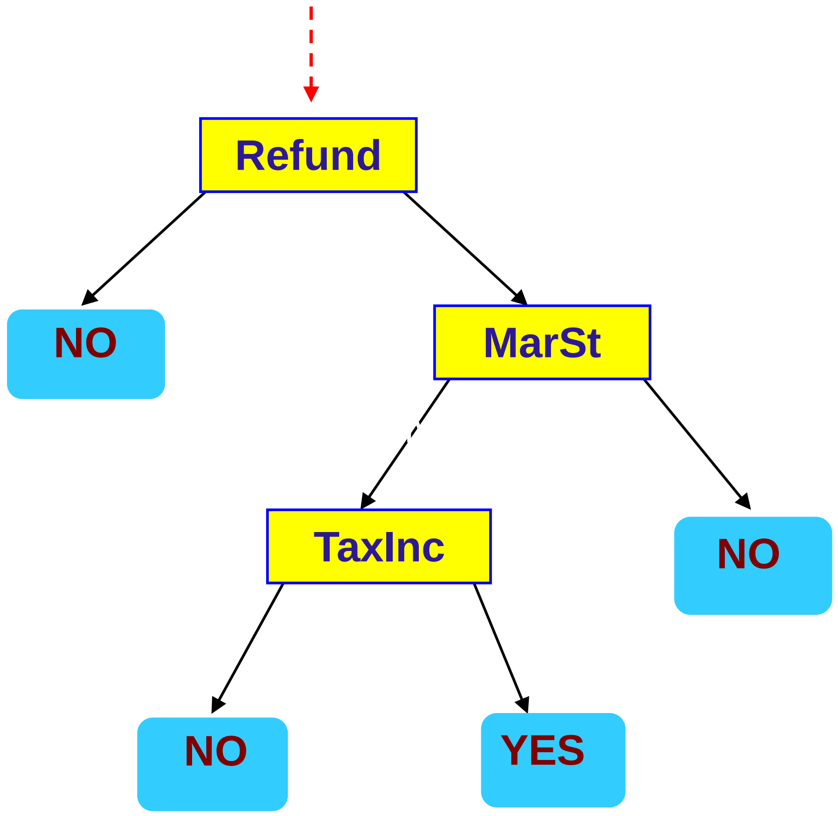
Training Set



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

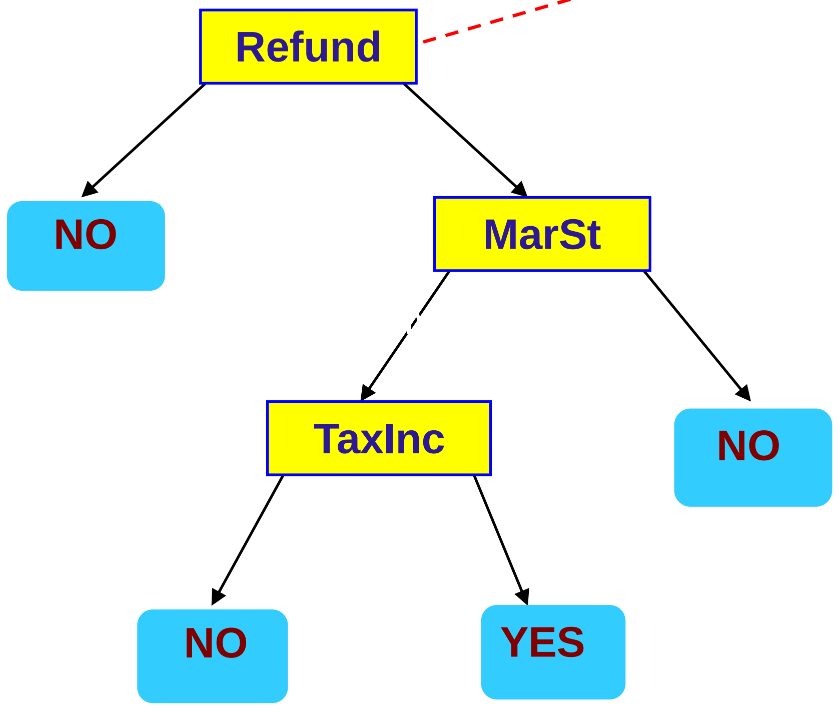
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

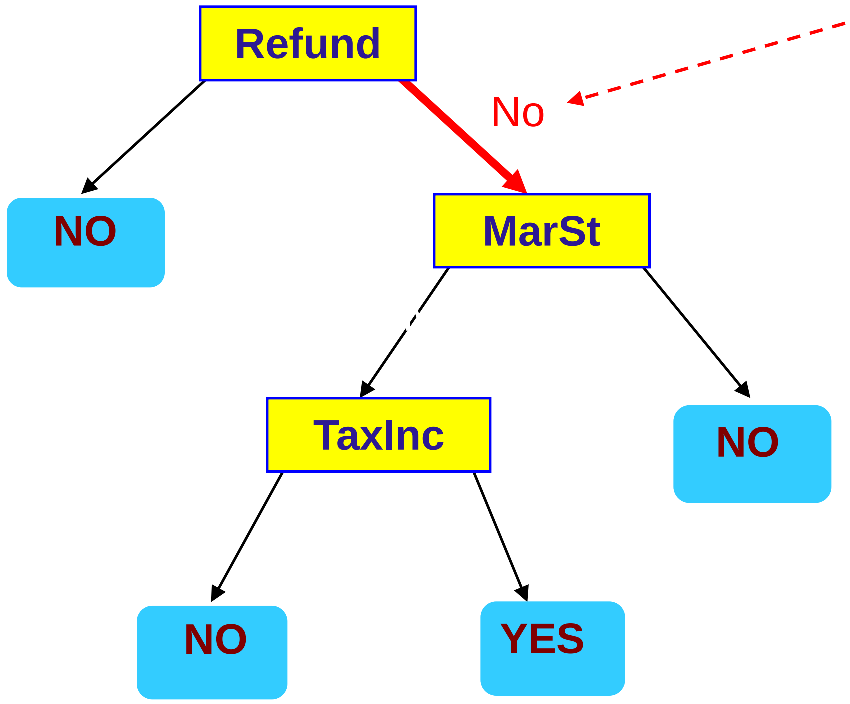
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

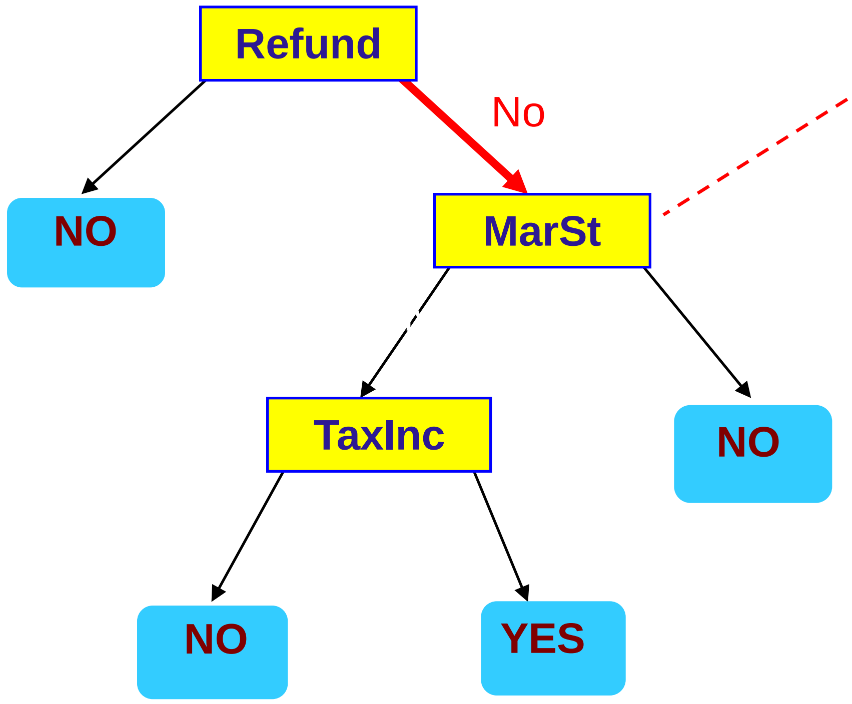
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

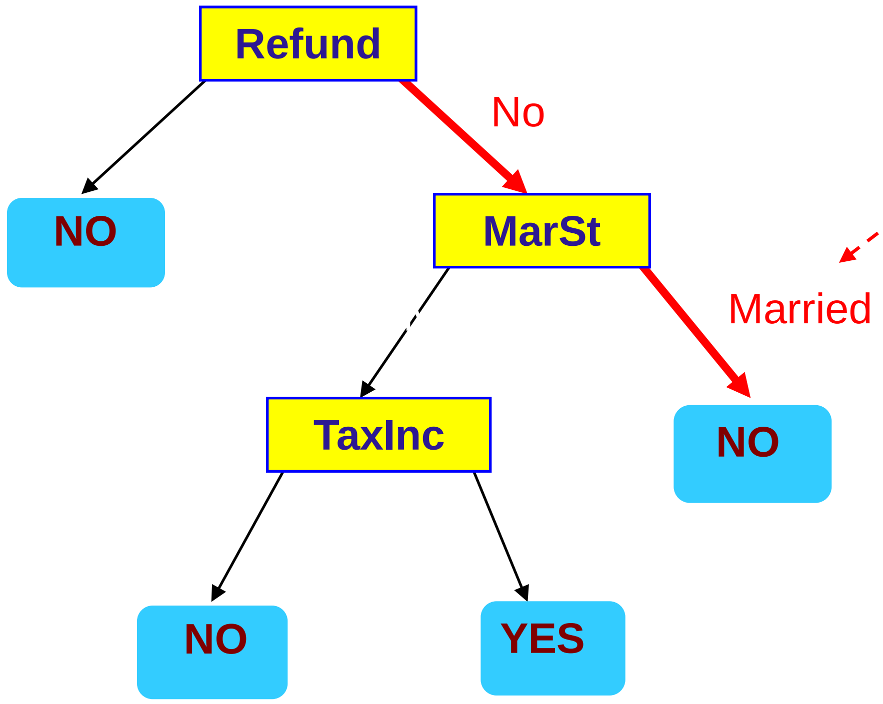
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

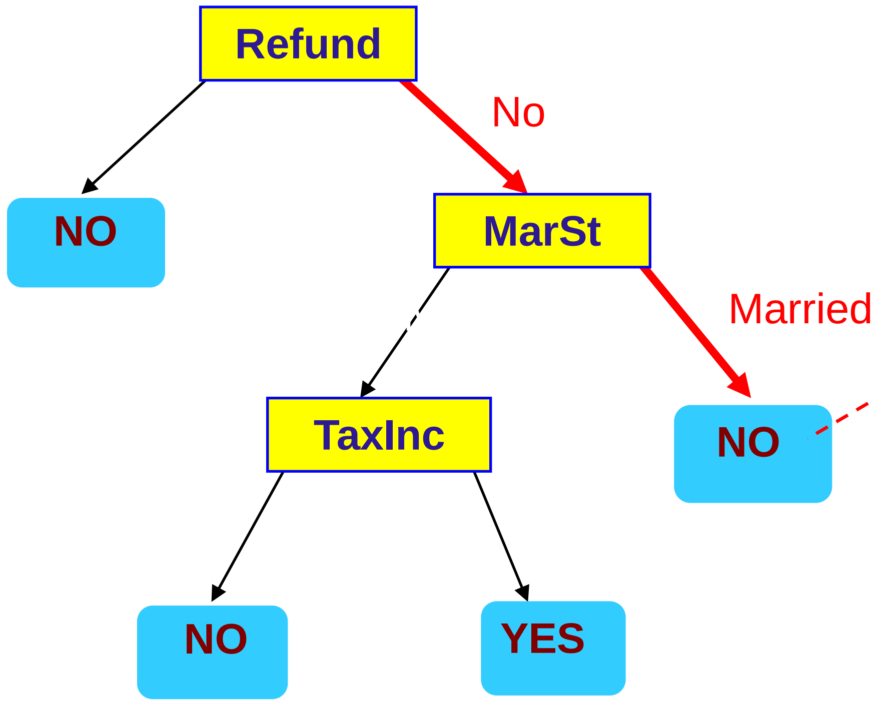
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



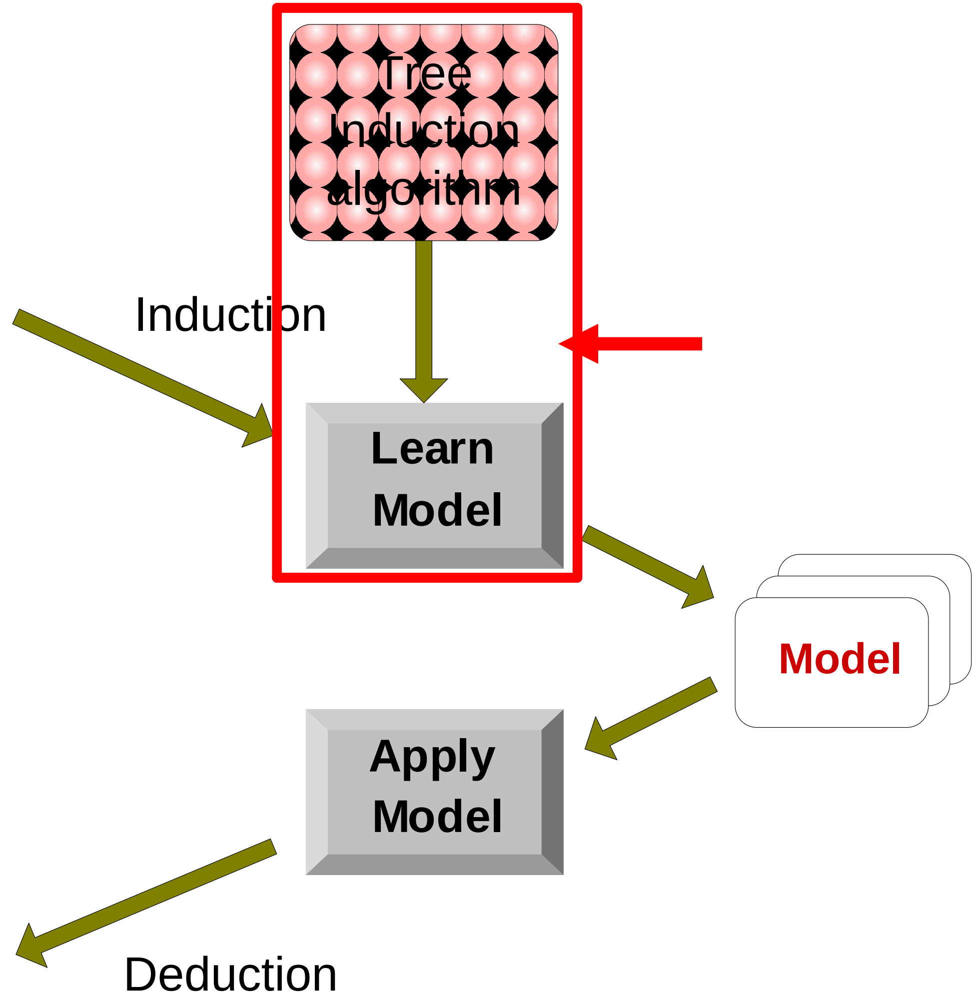
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Decision Tree Learning 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 Algorithms

Many Algorithms:

- Hunt’s Algorithm (one of the earliest)
- CART: Classification And Regression Tree
- ID3, C4.5, C5.0
- SLIQ
- SPRINT
- ...

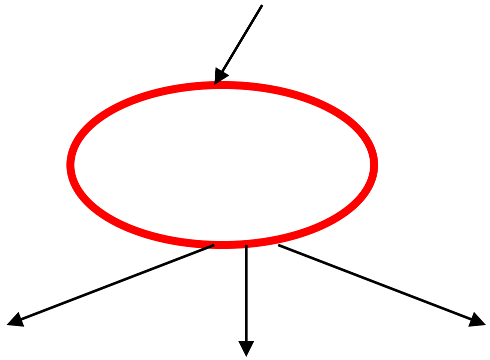
General Structure of Hunt's Algorithm

Let D_t be the set of training records that reach a node t

General recursive procedure:

- If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
- If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
- If D_t contains records that belong to more than one class:
  - Use an attribute test to split the data into smaller subsets (child nodes).
  - Recursively apply the procedure to each subset.

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
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**



Tree Induction

Greedy strategy

- Split the records based on an attribute test that optimizes a certain criterion.
  - Gini Index (CART)
  - Entropy and Information Gain (ID3, C4.5)
  - Misclassification error

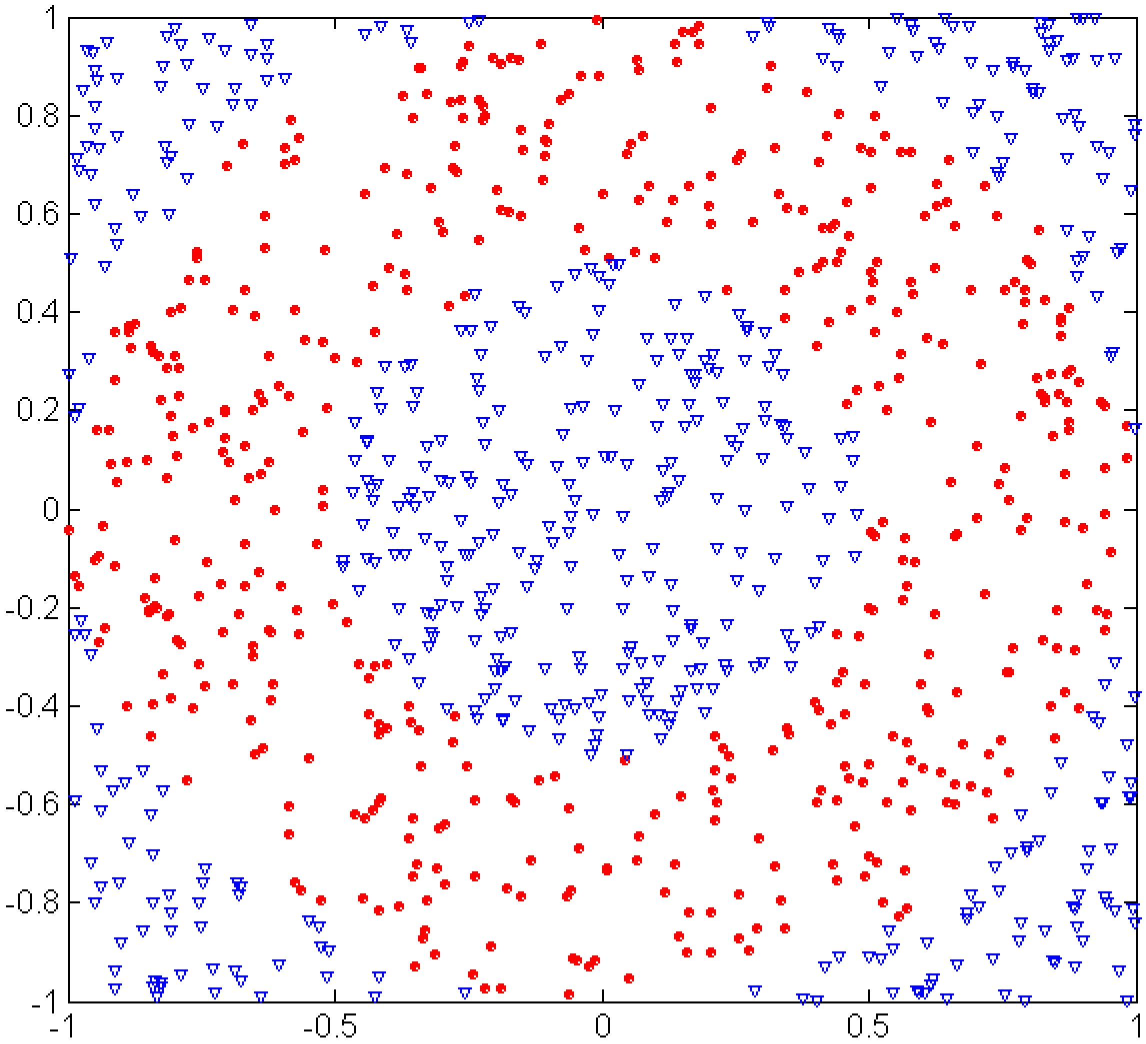
Issues

- Determine how to split the records
  - How to specify the attribute test condition?
  - How to determine the best split?
- Determine when to stop splitting

Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

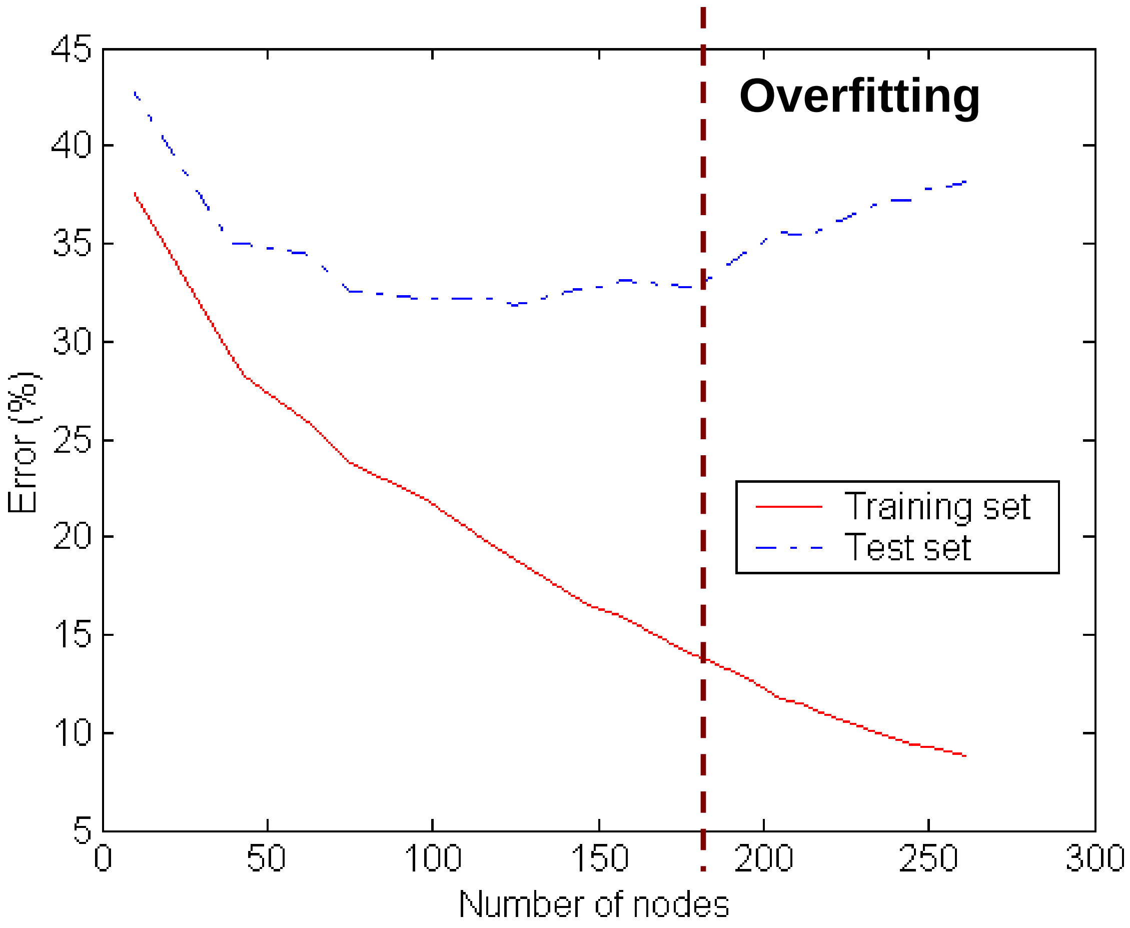
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

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

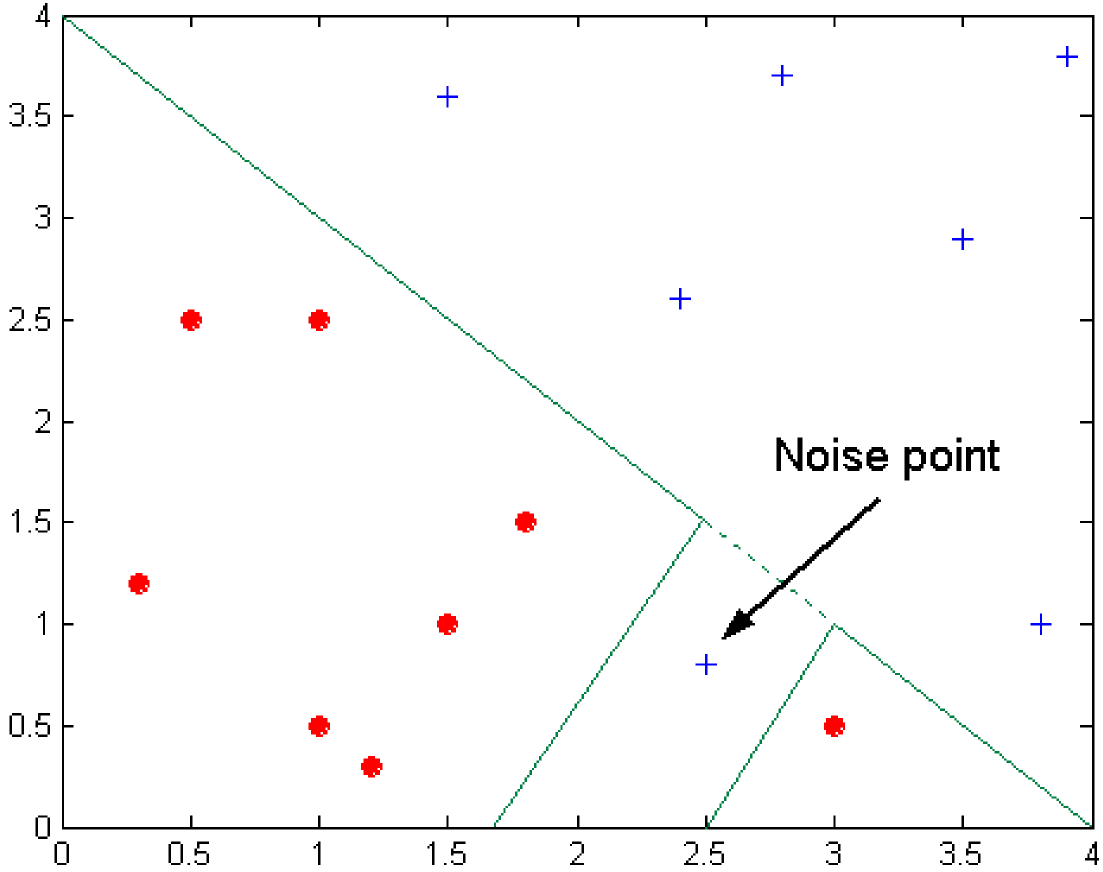
$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



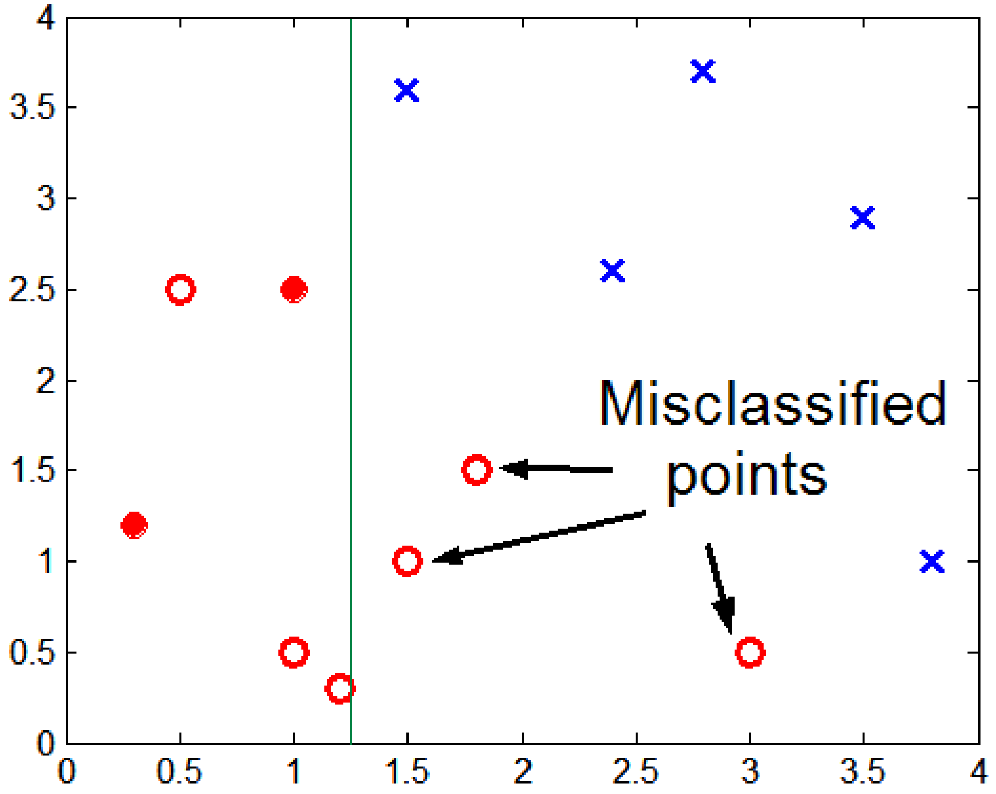
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



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

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

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
  - Pre-pruning
  - Post-pruning

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors
  - estimating generalization errors
  - model evaluation: cross-validation

Model Evaluation

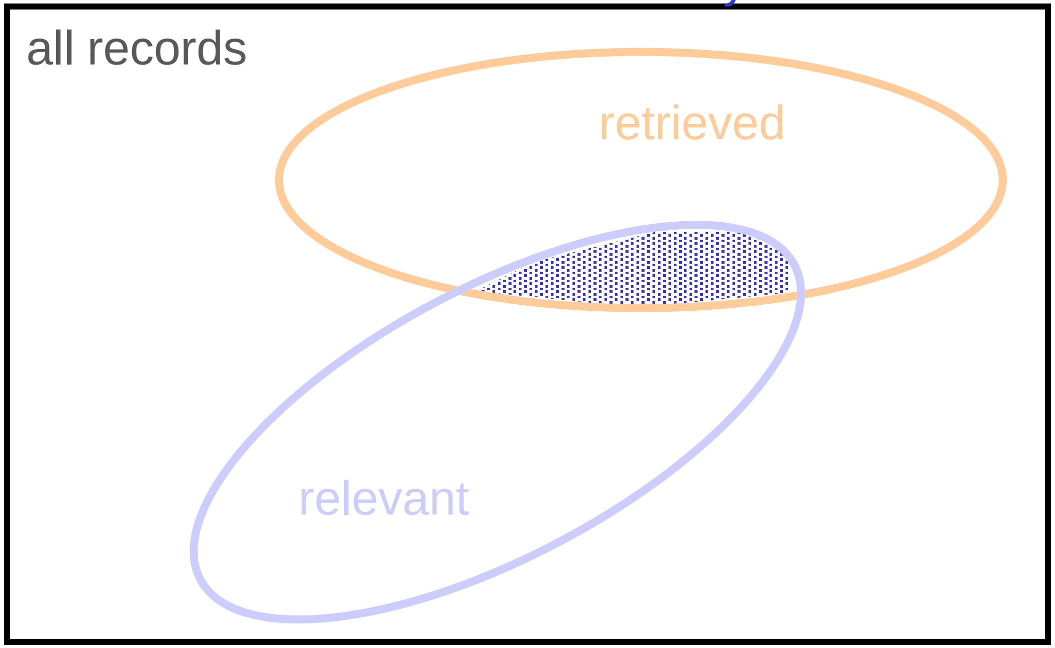
- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Evaluation of Predictive Tasks

For supervised classification we have a variety of measures to evaluate how good our model is

- Classification task
  - **Accuracy**: the fraction of classifications that are correct
  - **Error rate** = 1 - accuracy
- Information Retrieval task
  - **Recall**: proportion of relevant material actually retrieved
  - **Precision**: proportion of retrieved material actually relevant

Information Retrieval Systems



$$\text{Precision} = \frac{|\text{Relevant Retrieved}|}{|\text{Retrieved}|}$$

$$\text{Recall} = \frac{|\text{Relevant Retrieved}|}{|\text{Relevant in Collection}|}$$

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.

- Confusion Matrix:

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

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Accuracy

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

- Most widely-used metric:

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

$$\text{Error rate} = \frac{b+c}{a+b+c+d} = \frac{FN+FP}{TP+TN+FP+FN} = 1 - Accuracy$$

Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

Cost Matrix

	PREDICTED CLASS		
	C(i|j)	**Class=Yes**	**Class=No**
ACTUAL CLASS	**Class=Yes**	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

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

Cost	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if
1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$$\begin{aligned}\text{Cost} &= p\,(a + d) + q\,(b + c)\\ &= p\,(a + d) + q\,(N - a - d)\\ &= q\,N - (q - p)(a + d)\\ &= N\,[q - (q-p) \times \text{Accuracy}]\end{aligned}$$

Cost-Sensitive Measures

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

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

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

Model Evaluation

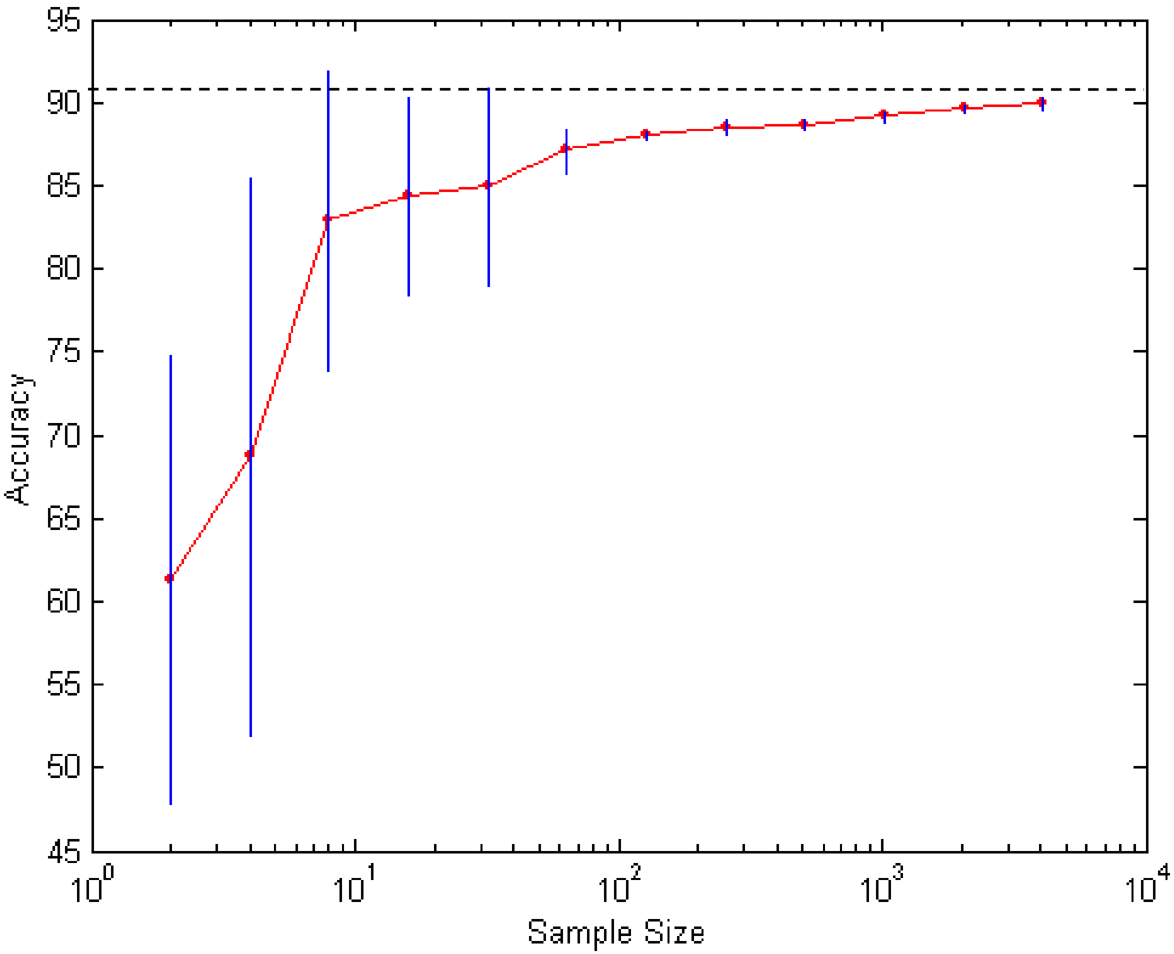
- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?

- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Learning Curve



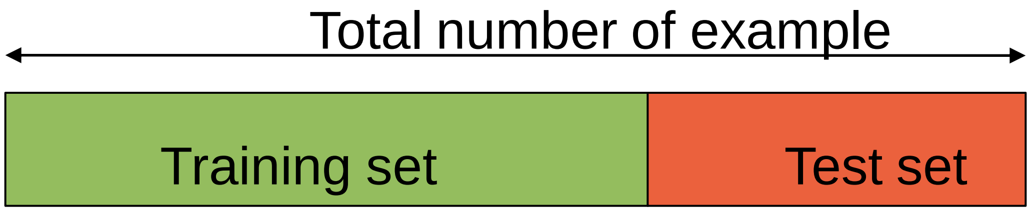
- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

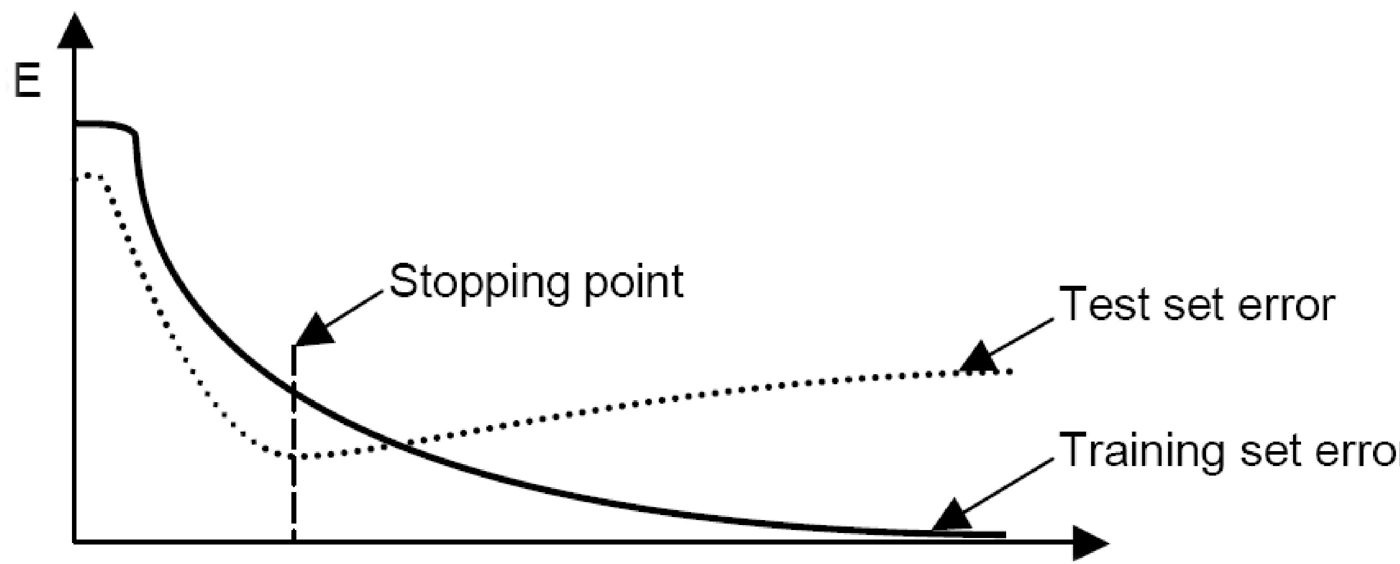
Effect of small sample size:

- Bias in the estimate
- Variance of estimate

The Holdout Method

- Split dataset into two groups
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier





The Holdout Method

- Holdout method
  - Original data is partitioned into two disjoint sets
  - E.g. 2/3 for training and 1/3 for testing, 50%-50%
  - Several issues

- Issues
  - Less data available for training
  - Bias on the training set
  - Training set and test set are not independent (e.g. there may be unbalanced classes in both)

Methods of Estimation

- Holdout
  - Original data is partitioned into two disjoint sets
- Random subsampling
  - Repeated holdout
  - Still some issues
    - Similar issues to holdout
    - No control over the selected samples (samples may be chosen multiple times or never)
- 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
- Bootstrap
  - Sampling with replacement
  - .632 bootstrap

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

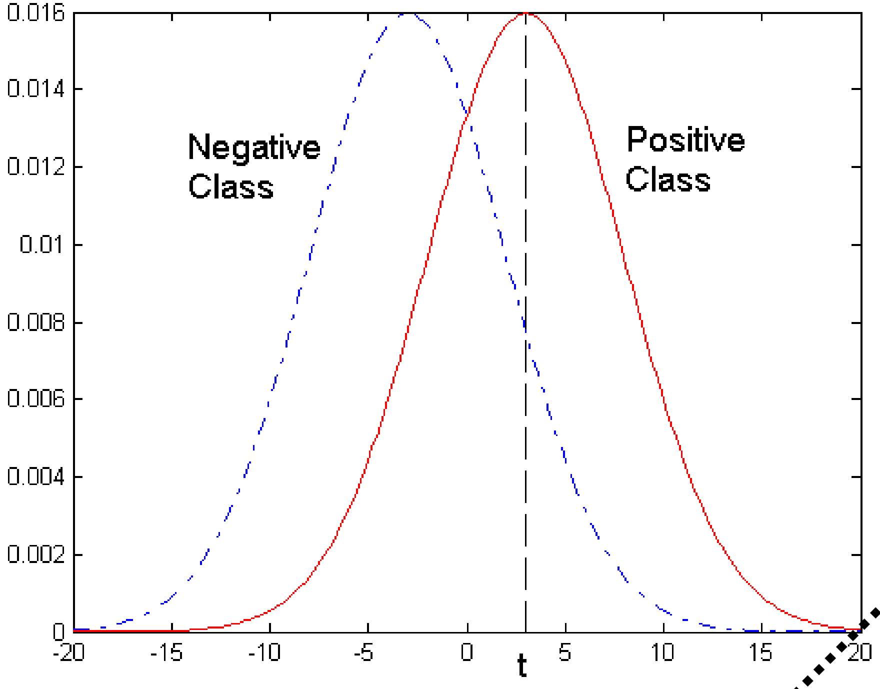
- Methods for Performance Evaluation
  - How to obtain reliable estimates?

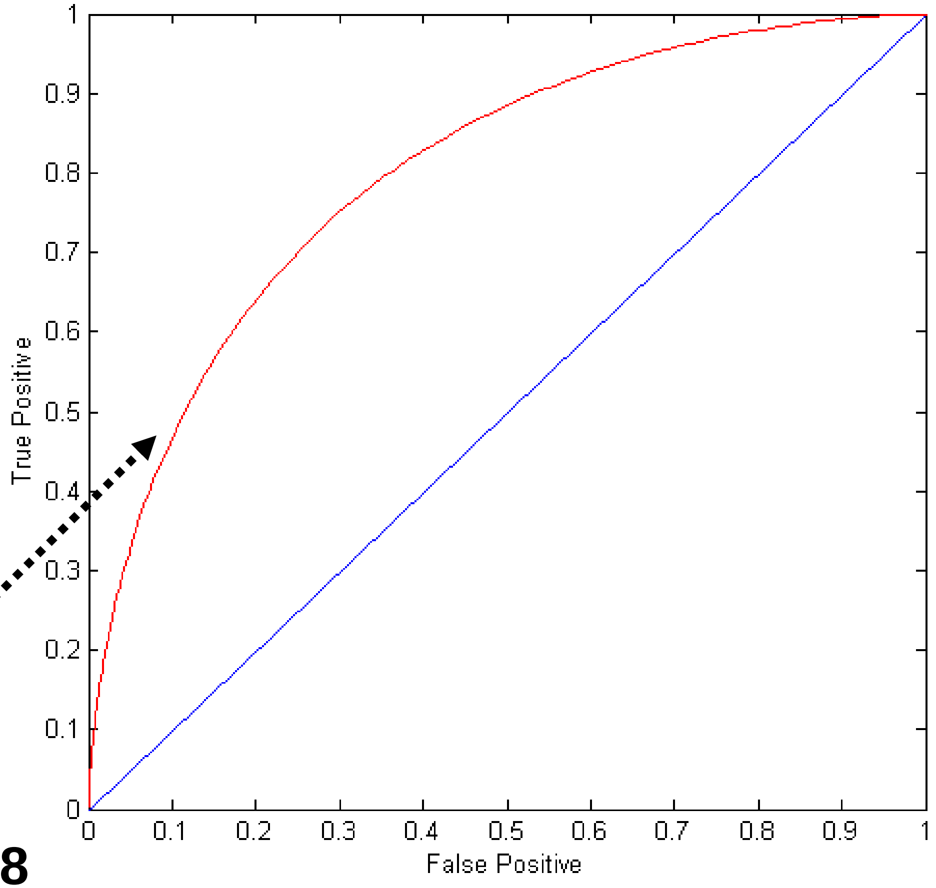
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at x > t is classified as positive





At threshold t:

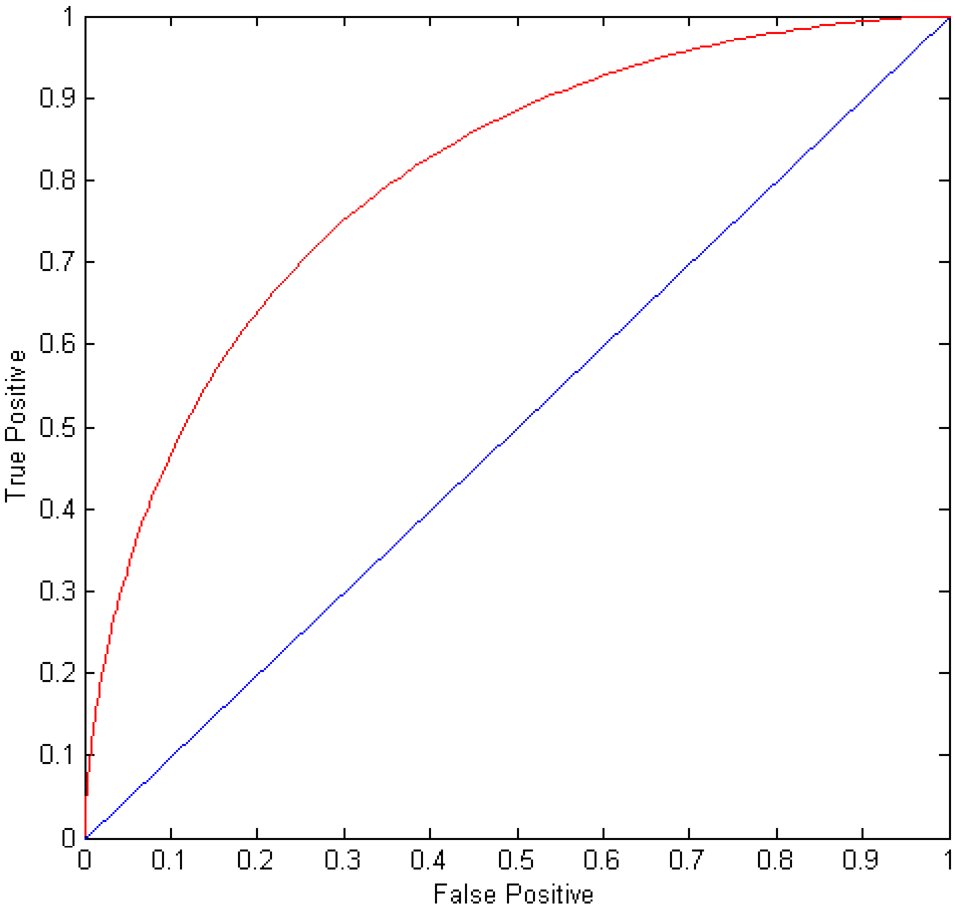
TP=0.5, FN=0.5, FP=0.12, FN=0.88

ROC Curve

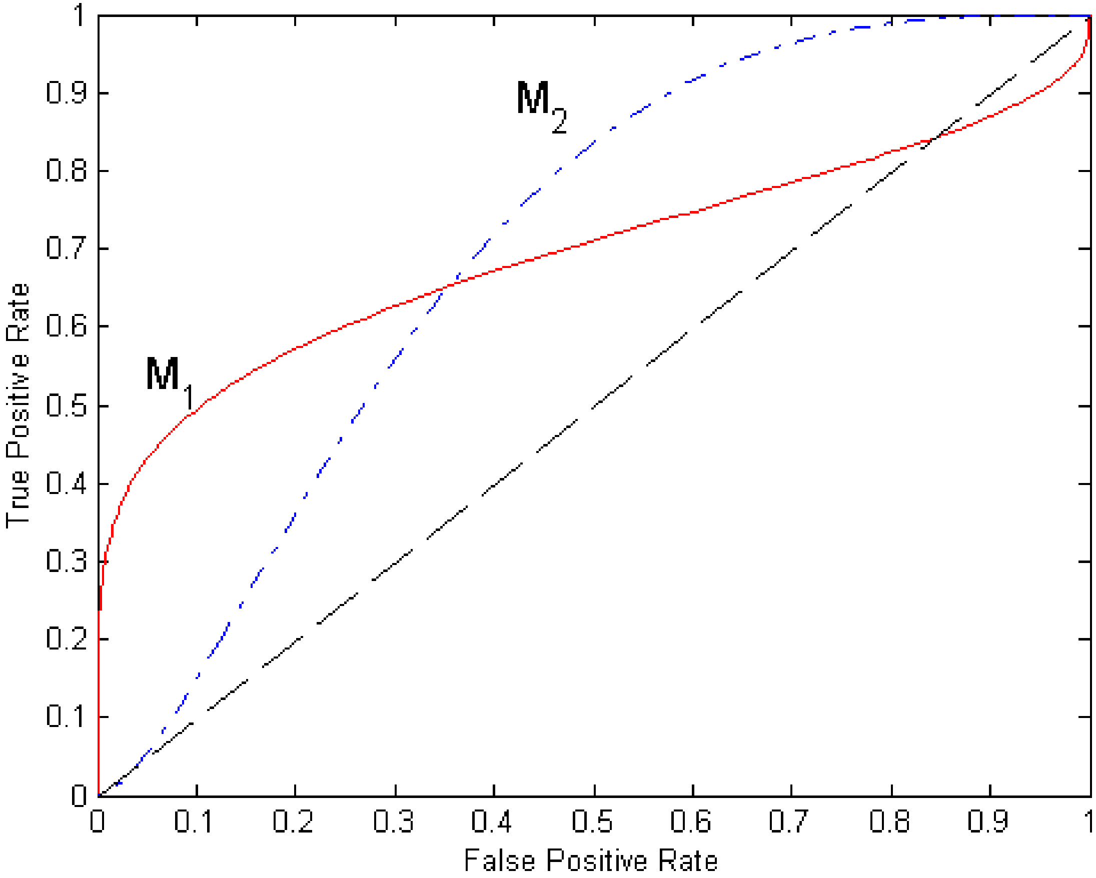
(TP,FP):

- (0,0): declare everything
  to be negative class
- (1,1): declare everything
  to be positive class
- (0,1): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



Using ROC for Model Comparison



- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5