Topics

- Introduction
- Decision Tree Induction
- **Model Overfitting**
- Evaluating the Performance of a Classifier
  - Metrics for Performance Evaluation
  - Methods for Performance Evaluation

Model Overfitting

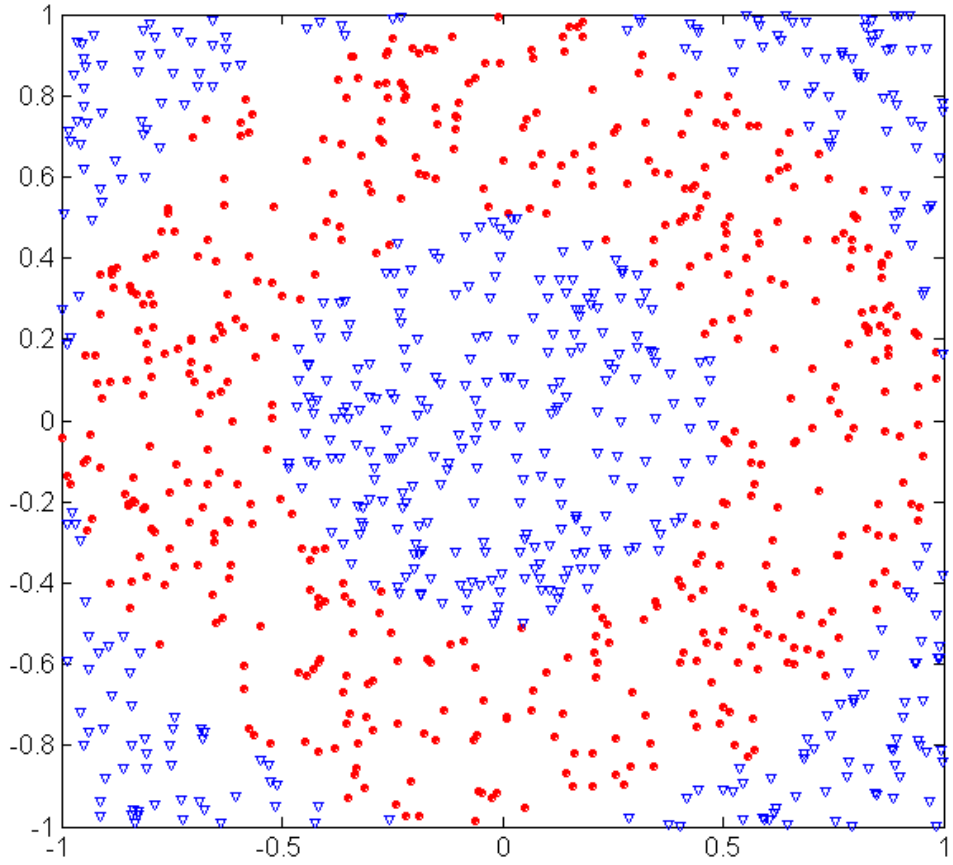
- **Two types of errors committed by a classification model:**

  1) Training errors (aka **resubstitution error** or **apparent error**) is the number of misclassification errors committed on training records.

  2) Generalization error is the expected test error of the model on previously unseen records.

- **A good model** must have low training error as well as low generalization error.
- A model that fits the training data too well (low training error) can have a poorer generalization error (high generalization error) than a model with a higher training error.
- **Overfitting** is the situation where a model has low training error and high generalization error.

Model Overfitting



500 circular and 500 triangular data points.

Circular points:

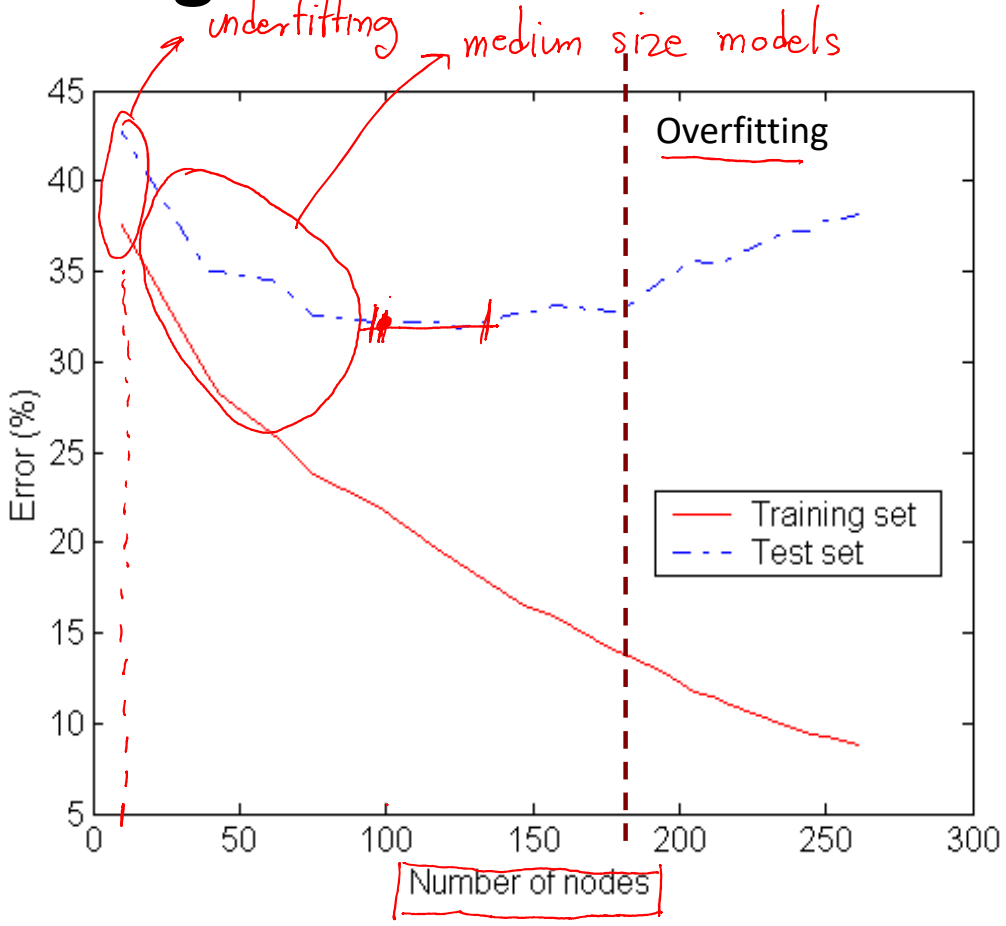
$$0.5 \leq \sqrt{x_1^2 + x_2^2} \leq 1$$

Triangular points:

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

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

Model Overfitting



Model Overfitting

- **Small size (low number of nodes)** ▸ The training and test error rates of the model are large. This situation is known as **model underfitting**. It occurs because the model has yet learn the true structure of the data ; in other words, the model is too simple.
- **Medium size (number of nodes increases)** ▸ The tree will have fewer training and generalization errors.
- **Large size (high number of nodes)** ▸ The generalization error rate begins to increase even though its training error rate continues to decrease. This situation is known as **model overfitting**
- Training error can be reduced by increasing the model complexity (number of nodes in a decision tree) but generalization error will increase when the model is too complex.

Causes of Model Overfitting

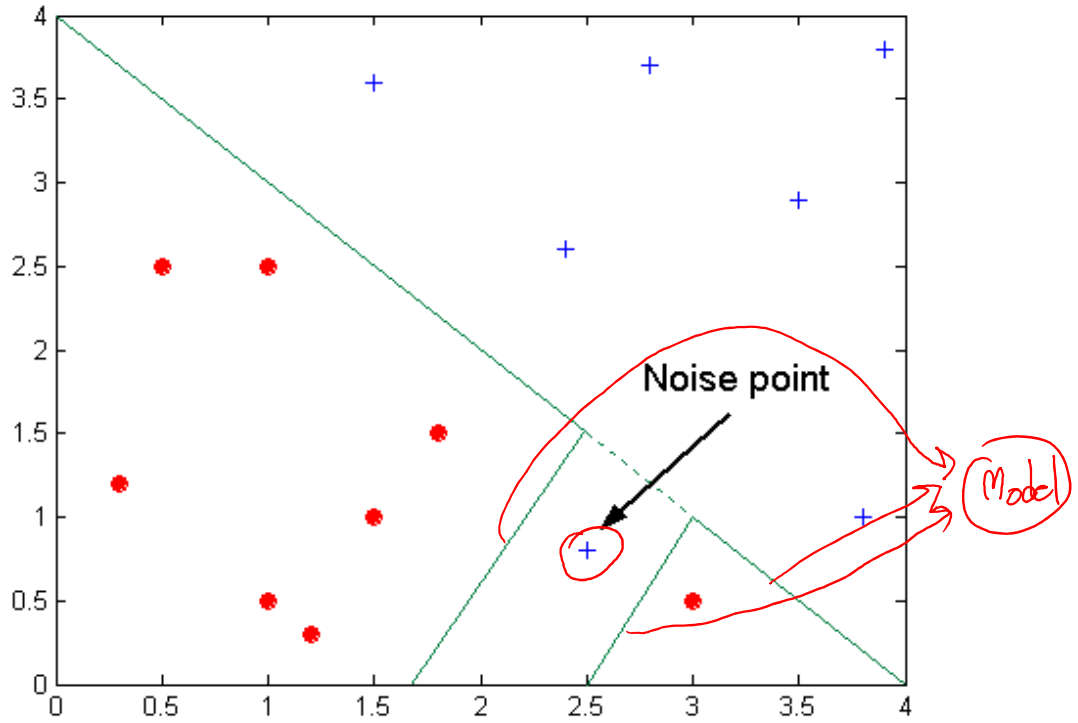
- **Model overfitting** occurs because

  1) The model fits the training data too well due to the presence of noise.

  2) The model is generated based on a small number of training records which leads to lack of representative samples (insufficiency).

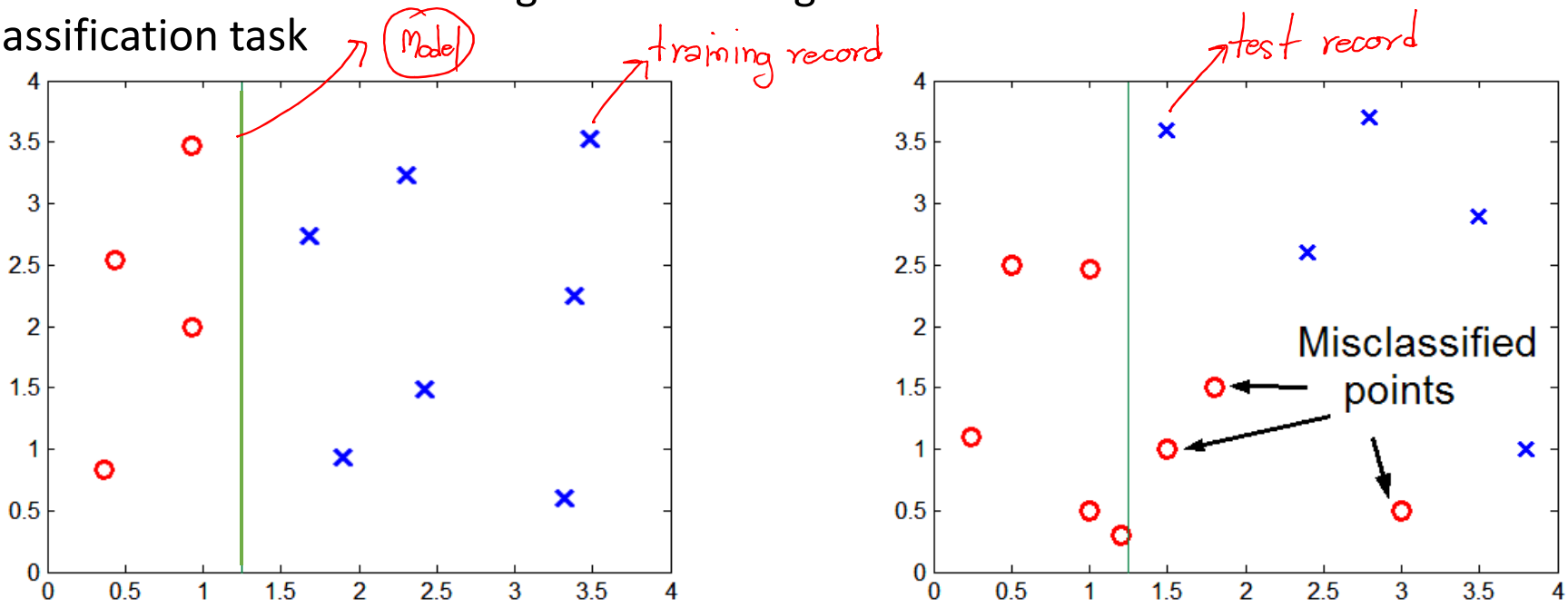
Overfitting due to Presence of Noise Example

- Decision boundary is distorted by noise point



Overfitting due to Lack of Representative Samples Example

- Lack of data records o in the right region which 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 records 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
- 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

Occam's Razor or The Principle of Parsimony

- We should prefer simpler models because the chance for model overfitting increases as the model becomes more complex.
- The additional components in a complex model stand a greater chance of being fitted purely by chance.

Occam's Razor Definition:

Given two models with the same generalization errors, the simpler model is preferred over the more complex model.

Estimation of Generalize Errors

- The complexity of a model has an impact on model overfitting.
- The ideal complexity is that of a model that produces the lowest generalization error.
- The problem is that the learning algorithm has access only to the training set but has no knowledge of the test set; thus, we do not know how well the tree will perform on records it has never seen before.
- The best it can do is to estimate the generalization error of the induced tree.

Estimation of Generalize Errors

Methods for estimating generalization errors:

1. Optimistic approach:

- This method uses the training error (resubstitution error) to provide generalization error.
- This is because it assumes that the training set is a good representation of the overall data.

For each leaf node: $e_g(t) = e(t)$

For total errors: $e(T) = \sum_{i=1}^{k} e(t_i)$

$e_g(T) = e(T)$

where

$e_g(t)$ = Generalization error of leaf node

$e(t)$ = Training error of leaf node

$e(T)$ = The overall training error of a decision tree

$e_g(T)$ = Generalization error of a decision tree

k = The number of leaf nodes

- A decision tree induction algorithm simply selects the model that produces the lowest training error rate as its final model.
- The training error is usually a poor estimate of generalization error.

Estimation of Generalize Errors

2) Pessimistic approach:

- This approach is similar to optimistic approach except that a penalty term for model complexity will be added to the equation.

For each leaf node:

$$e_g(t) = e(t) + \theta$$

For total errors:

$$e_g(T) = \frac{\sum_{i=1}^{k}[e(t_i) + \theta(t_i)]}{\sum_{i=1}^{k} n(t_i)} = \frac{e(T) + \theta(T)}{N}$$

where
$n(t)$ = The number of training records classified by node t
$e(t)$ = The number of misclassified records
$e(T)$ = The overall training error of a decision tree
k = Number of leaf nodes
N = Number of training records
$\theta(t_i)$ = The penalty term associated with each node t_i

- Example: For a tree with 30 leaf nodes and 10 errors on training (out of 1000 records), $\theta = 0.5$

Generalization error $e_g(T) = \frac{10+(30)(0.5)}{1000} = 0.025$

Estimation of Generalize Errors

3) Using a validation set:

- Instead of using the training set to estimate the generalization error, the original training data is divided into two smaller subsets.

  Subset 1 (the training set) is used for training.

  Subset 2 (the validation set) is used for estimating the generalization error.

- Typically, two-third of the training set is reserved for model building, while the remaining one-third is used for error estimation.

Handling Overfitting in Decision Tree Induction

Two strategies for avoiding model overfitting:

1. **Pre-Pruning (Early Stopping Rule):** It stops the algorithm before it becomes a fully-grown tree that perfectly fits the entire training data using a more restrictive stopping condition.
   - **Typical stopping conditions:**
     (1) Stop if all records belong to the same class
     (2) Stop if all the attribute values of records are the same
   - **More restrictive conditions:**
     (1) Stop if number of records is less than some user-specified threshold
     (2) Stop if expanding a leaf node when the gain in impurity measure (or improvement in the estimated generalization error) falls below a certain threshold.

Handling Overfitting in Decision Tree Induction

- **Advantage:** It avoids generating overly complex subtrees that overfit the training data.
- **Disadvantage:** It is difficult to choose the right threshold for early termination.
  - Too high threshold will result in underfitted models.
  - Too low threshold may not be sufficient to overcome the model overfitting problem.

Handling Overfitting in Decision Tree Induction

2. Post-pruning: It lets the decision tree grow to its maximum size first, then trim the fully grown tree in a bottom-up fashion.

- Trimming can be done by replacing a subtree with:

  (1) A new leaf node whose class label is determined from the majority class of records affiliated with the subtree.

  (2) The most frequently used branch of the subtree.

- Trimming process is terminated when no further improvement is observed.
- **Advantage:** This approach tends to give better results than pre-pruning because it makes pruning decision based on a fully grown tree.
- **Disadvantage:** The additional computations needed to grow the fully tree may be wasted when the subtree is pruned.

Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Before splitting

Training Error = 10/30

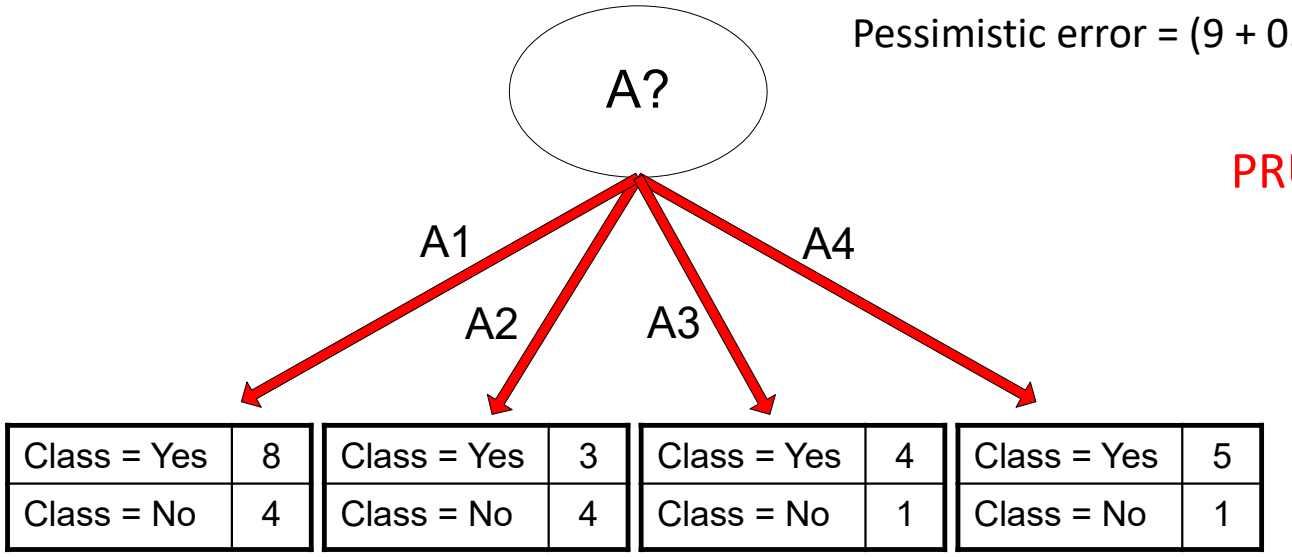
Pessimistic error = (10 + 0.5)/30 = 10.5/30 = 0.35

After splitting

Training Error = 9/30

Pessimistic error = (9 + 0.5(4))/30 = 11/30 = 0.37

PRUNE!



Topics

- Introduction
- Decision Tree Induction
- Model Overfitting
- **Evaluating the Performance of a Classifier**
  - Metrics for Performance Evaluation
  - Methods for Performance Evaluation

Evaluating the Performance of a Classifier

- In model overfitting section, several method for estimating the generalization error of a model during training are described.
- The estimated error helps the learning algorithm to do **model selection** (to find a model of the right complexity that is not susceptible to overfitting).
- Once the model has been constructed, it can be applied to the test set to predict the class labels of previously unseen records.
- It is useful to measure the performance of the model on the test set because such a measure provides an unbiased estimate of its generalization error.

Evaluating the Performance of a Classifier

Two components needed to be considered:

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

2. **Methods for Performance Evaluation**
   - How to obtain reliable estimates?

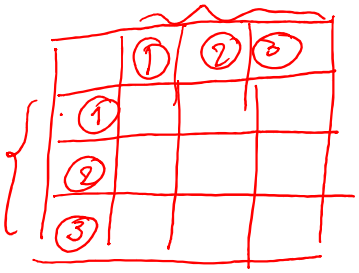
We need to select a metric and a method before evaluating the performance of a classifier.

Topics

- Introduction
- Decision Tree Induction
- Model Overfitting
- Evaluating the Performance of a Classifier
  - **Metrics for Performance Evaluation**
  - Methods for Performance Evaluation

Metrics for Performance Evaluation

- The **performance evaluation** of a classification model is based on the counts of test records predicted correctly or incorrectly by a classification model. These counts are tabulated in a table known as **a confusion matrix**.



- **Confusion Matrix for binary-class problem:**

		PREDICTED CLASS	
		+	-
ACTUAL CLASS	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

a = True Positive (TP)

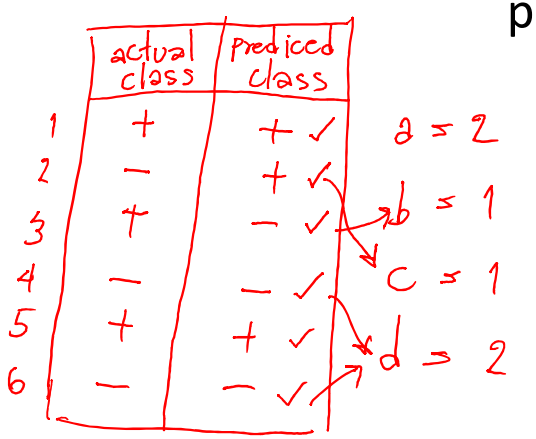
b = False Negative(FN)

c = False Positive (FP)

d = True Negative (TN)

Metrics for Performance Evaluation

- **True positive (TP)** corresponds to the number of positive examples correctly predicted by the classification model.
- **False negative (FN)** corresponds to the number of positive examples incorrectly predicted by the classification model.
- **False positive (FP)** corresponds to the number of negative examples incorrectly predicted by the classification model.
- **True negative (TN)** corresponds to the number of negative examples correctly predicted by the classification model.



	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

Metrics for Performance Evaluation

- **Two most widely-used metrics:**

1) Accuracy: It is a fraction of records that we predict correct.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

2) Error Rate: It is a fraction of records that we predict incorrect.

$$Error\ Rate = \frac{Number\ of\ wrong\ predictions}{Total\ number\ of\ predictions} = \frac{b+c}{a+b+c+d} = \frac{FP+FN}{TP+TN+FP+FN}$$

		PREDICTED CLASS	
		+	-
ACTUAL CLASS	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

Metrics for Performance Evaluation

The counts of confusion matrix can be expressed in terms of percentage.

- The **true positive rate (TPR)** or sensitivity is defined as the fraction of positive examples predicted correctly by the model.

$$TPR = \frac{a}{a+b} = \frac{TP}{TP+FN}$$

	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

- The **true negative rate (TNR)** or specificity is defined as the fraction of negative examples predicted correctly by the model.

$$TNR = \frac{d}{d+c} = \frac{TN}{TN+FP}$$

Metrics for Performance Evaluation

The counts of confusion matrix can be expressed in terns of percentage.

- The **false positive rate (FPR)** is the fraction of negative examples predicted as a positive class.

$$FPR = \frac{c}{c+d} = \frac{FP}{FP+TN}$$

	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

- The **false negative rate (FNR)** is the fraction of positive examples predicted as a negative class.

$$FNR = \frac{b}{b+a} = \frac{FN}{FN+TP}$$

Limitation of Accuracy

- Consider a 2-class problem of size 10,000 objects
  - Number of Class 0 examples = 9,990
  - 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

Class imbalance problem!

Handling the Class Imbalance Problem

1. **Cost of classification using cost matrix:** Compute total cost of prediction instead of accuracy or error rate.

Count	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	a	b
	-	c	d

cost matrix

Cost	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	C(+|+)	C(-|+)
	-	C(+|-)	C(-|-)

$$Cost = (a)(C(+|+)) + (b)(C(-|+)) + (c)(C(+|-)) + (d)(C(-|-))$$

- Usually cost C(+|+) = C(-|-) and C(+|-) = C(-|+)
- Select the model with the lowest cost

Handling the Class Imbalance Problem

Example: Computing cost of classification and accuracy of 2 models

C(+|+)=-1 means reward for making correct classification

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

← Missing a + case is really bad.

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

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

Accuracy = (150+250)/(150+40+60+250) = 80%

Cost = (-1)(150)+(100)(40)+(1)(60)+(0)(250) = 3,910

Accuracy = (250+200)/(250+45+5+200) = 90%

Cost = (-1)(250)+(100)(45)+(1)(5)+(0)(200) = 4,255

Handling the Class Imbalance Problem

2. Use cost-sensitive measures:

1) Precision determines the fraction of records that actually turns out to be positive in the group the classifier has declared as a positive class. It is biased towards C(+|+) & C(+|-)

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

2) Recall measures the fraction of positive examples correctly predicted by the classifier. It is biased towards C(+|+) & C(-|+)

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

	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

Handling the Class Imbalance Problem

3) F_1 measure summarizes precision and recall. It represents a harmonic mean between precision and recall.

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}}$$

$$F_1 = \frac{2rp}{r + p}$$

$$\boxed{F_1 = \frac{2a}{2a + b + c}}$$

	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

Note: A high value of F_1 measure ensures that both precision and recall are reasonably high.

Handling the Class Imbalance Problem

4) Kappa Statistics compares the accuracy of the classifier with a random classifier.

$$Kappa(\kappa) = \frac{p_o - p_e}{1 - p_e}$$

$$p_0 = \frac{a + d}{a + b + c + d}$$

$$p_e = p_{Yes} + p_{No}$$

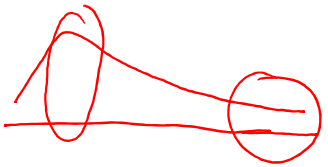
$$p_{Yes} = \frac{(a + b)(a + c)}{(a + b + c + d)^2}$$

$$p_{No} = \frac{(c + d)(b + d)}{(a + b + c + d)^2}$$

	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	a (TP)	b (FN)
	-	c (FP)	d (TN)

Handling the Class Imbalance Problem

3. Sampling-Based Approaches: This approach is used to modify the distribution of records so that the rare class is well represented in the training set.



Given that an original training set contains → 100 positive records

→ 1,000 negative records

1) Undersampling:

- A random sample of 100 negative records are chosen from a set of 1,000 to form the training along with all 100 positive records. The total number of records in the training set will be equal to 200 records.
- Some of useful negative records may not be chosen for training.
- The above problem can be address by performing undersampling multiple times to induce multiple classifiers (See Ensemble Method in Chapter 5)

Handling the Class Imbalance Problem

2) Oversampling:

- A random sample of 1,000 positive records are chosen from a set of 100 to form the training (sampling with replacement) along with all 1,000 negative records. The total number of records in the training set will be equal to 2,000 records.
- This approach may cause model overfitting for noisy data.

↗ in positive class

3) Hybrid approach:

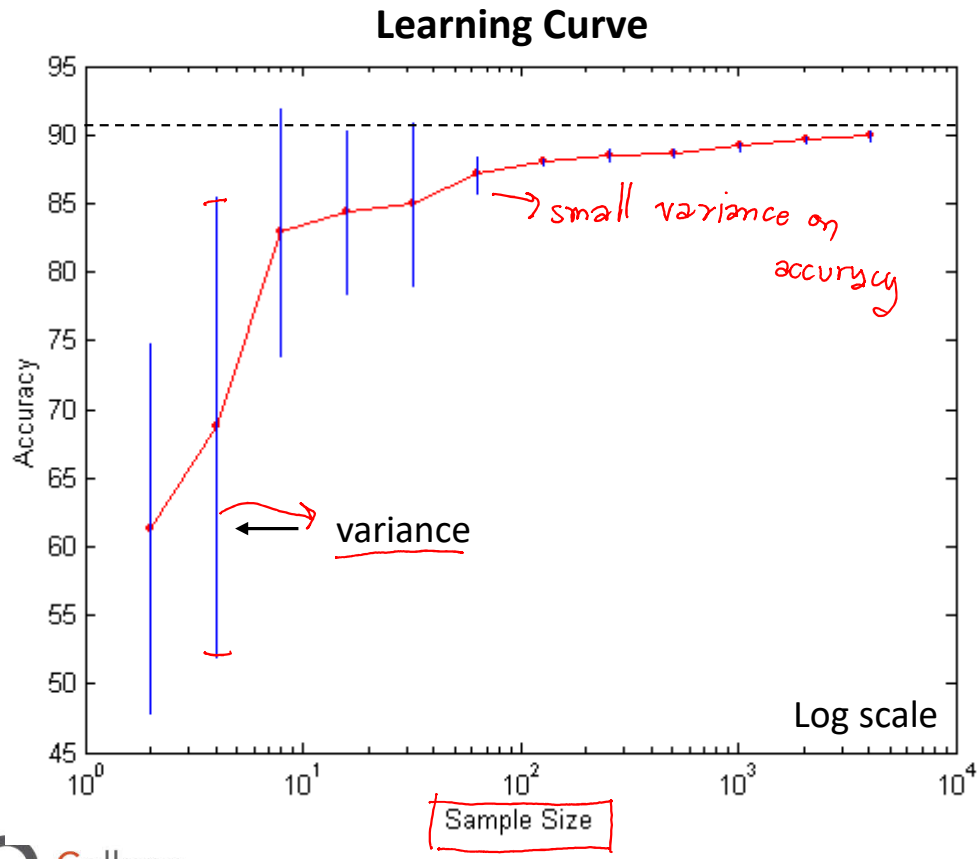
- This approach uses a combination of undersampling the majority class and oversampling the rare class to achieve uniform class distribution (equal number of both classes).
- A random sample of 500 negative records are chosen from a set of 1,000 and a random sample of 500 positive records are chosen from a set of 100 to form the training set. The total number of records in the training set will be equal to 1,000 records.

Topics

- Introduction
- Decision Tree Induction
- Model Overfitting
- Evaluating the Performance of a Classifier
  - Metrics for Performance Evaluation
  - **Methods for Performance Evaluation**

Methods for Performance Evaluation

- 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** shows how accuracy changes with varying sample size (training set size).
- Accuracy depends on the size of the training set.



Methods for Performance Evaluation

Six commonly used methods to evaluate the performance of a classifier:

1. Holdout:

- The known class attribute records is partitioned into two disjoint set called **training set** and **test set**.
- A classification model is induced from the training set and its performance is evaluated on the test set.
- The mostly used proportion for training and test set are 1:1 and 2:1.
- **Limitations:**

  (1) The model may not be as good as when all original data are used for training.

  (2) The model may be highly dependent on the composition of the training and test sets.

  Small training set: High variation in accuracy

  Large training set: Less reliable in accuracy computed from the smaller test set

  (3) The training and test sets are no longer independent of each other.

Methods for Performance Evaluation

2. Random Subsampling:

- This method repeats holdout method several times to improve the estimation of a classifier's performance.
- The overall accuracy is given by

$$acc_{sub} = \frac{\sum_{i=1}^{k} acc_i}{k}$$

where acc_{sub} = The model accuracy during the i^{th} iteration

k = Number of times

- **Limitation:** It has no control over the number of times each record is used for training and testing.

Methods for Performance Evaluation

3. Cross validation:

- Each record is used the same number of times for training and exactly once for testing.
- Two approaches for cross validation:

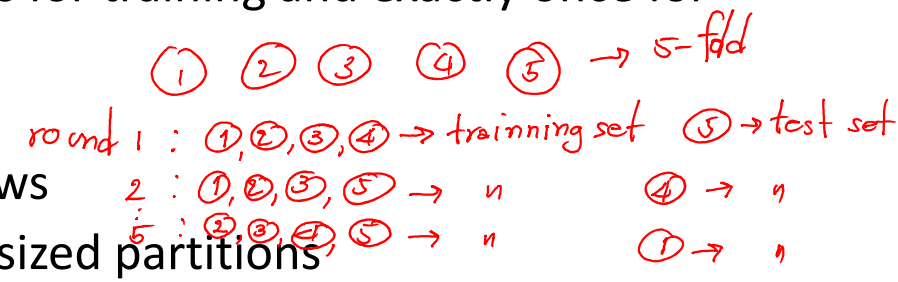
1) k-fold cross validation: steps are as follows

(1) Segment the N records into k equal-sized partitions

(2) Train on $k-1$ partitions, test on the remaining one partition

(3) Repeat (2) until each k partition is used for testing exactly once

(4) Compute total error by summing up the errors for N runs



2) Leave-one-out approach:

- It is a special case of k-fold cross validation where $k = N$ → number of records
- Each test set contains only one record

Methods for Performance Evaluation

4. Bootstrap:

- The training records are sampled with replacement.
- The model induced from the training set is then applied to the test set to obtain an estimate of the accuracy of the bootstrap sample (ϵ_i).
- The sampling procedure is then repeat b times to generate b bootstrap samples.
- **.632 bootstrap** is one of the most widely used variation

$$acc_{boot} = \frac{1}{b}\sum_{i=1}^{b}(0.632\epsilon_i + 0.368acc_s)$$

where acc_s = The accuracy computed from a training set that contains all the known class attribute records.

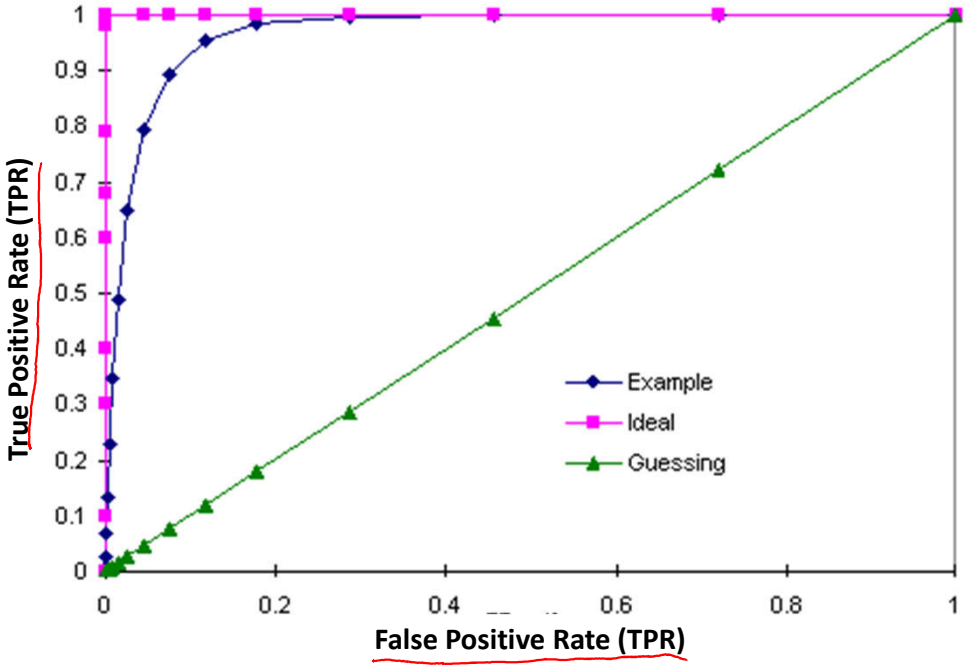
Methods for Performance Evaluation

5. **Receiver Operating Characteristic (ROC) Curve** is a graphical approach for displaying the tradeoff between true positive rate (TPR) (plotted on y axis) and false positive rate(FPR) (plotted on x axis) of a classifier where each point along the curve corresponds to one of the models induced by the classifier.

$$TPR = \frac{a}{a+b} = \frac{TP}{TP+FN}$$

$$FPR = \frac{c}{c+d} = \frac{FP}{FP+TN}$$

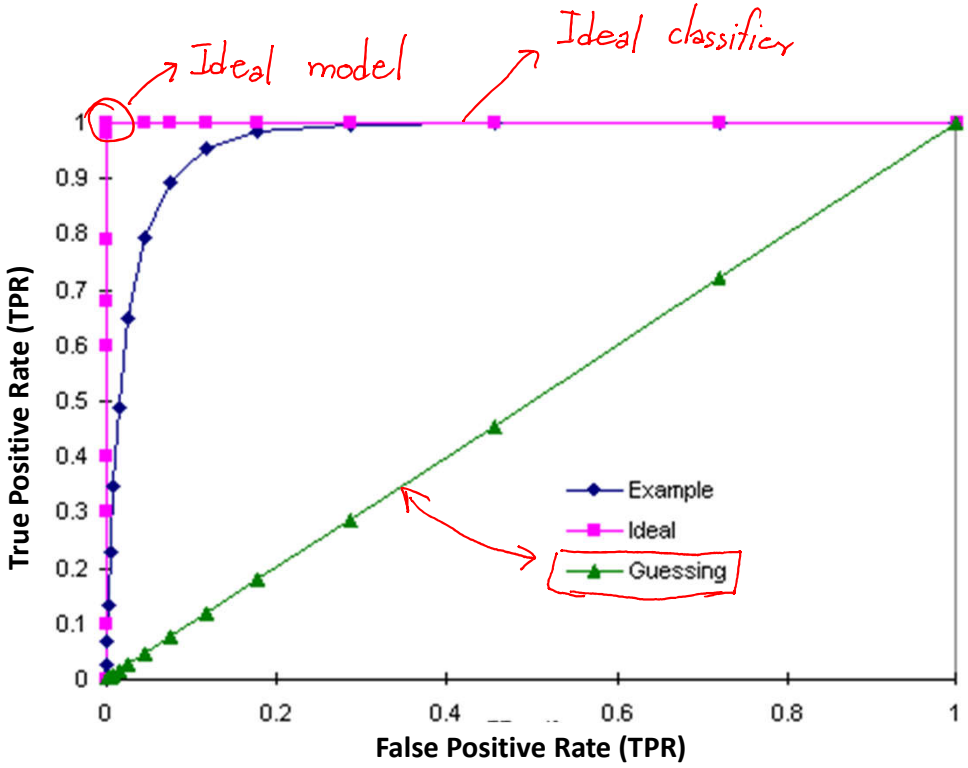
		PREDICTED CLASS	
		+	-
ACTUAL CLASS	+	a (TP)	b (FN)
	-	c (FP)	d (TN)



Methods for Performance Evaluation

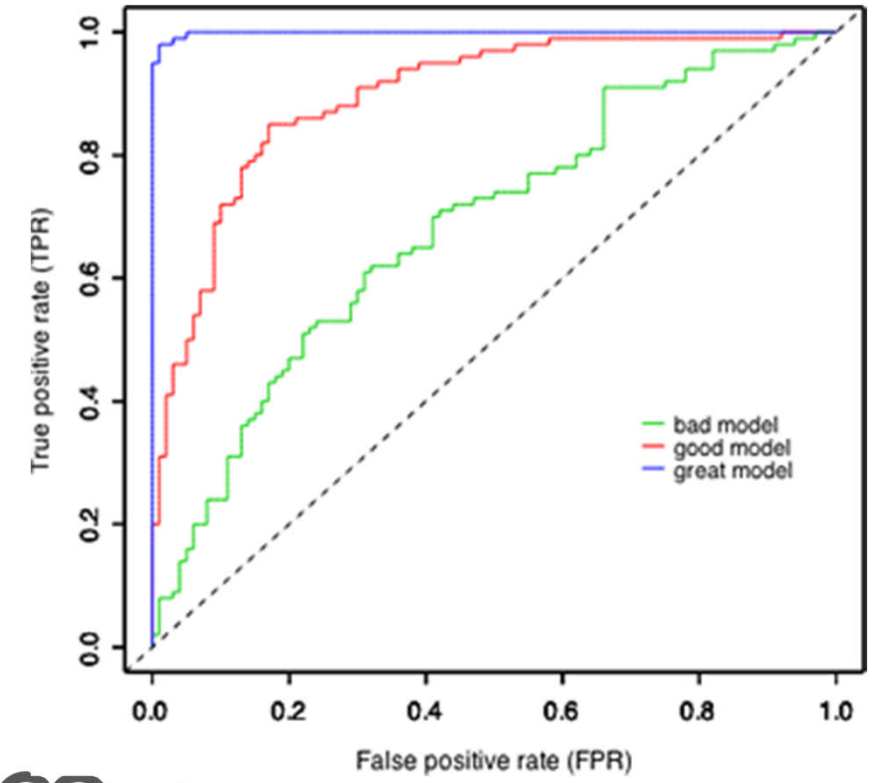
- **(TPR,FPR):**
  (0,0): Model predicts every records to be negative class
  (1,1): Model predicts every records to be positive class
  (1,0): The ideal model

- **Diagonal (green) line:**
  - Random guessing
  - Below diagonal line: prediction is opposite of the true class



Methods for Performance Evaluation

6. **Area under the ROC curve (AUC)** provides another approach for evaluating which ~~model~~ classifier is better on average. The best classifier has the largest area under the ROC curve.
   - The ideal ~~model~~ classifier : Area = 1
   - Random guessing : Area = 0.5



Methods for Performance Evaluation

- No model consistently outperform the other
  - M_1 is better than M_2 when FPR is less than 0.36
  - M_2 is better than M_1 when FPR is greater than 0.36