Classification

- **Model Evaluation and Selection**
- **Ensemble Methods: Increasing the Accuracy**

Model Evaluation and Selection

- ***Metrics for Performance Evaluation***
- *Methods for Performance Evaluation*
- *Methods for Model Comparison*

Model Evaluation and Selection

- ***Metrics for Performance Evaluation***
  - How to evaluate the performance of a model?
  - How can we measure **accuracy**? Other **metrics** to consider?
  - Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- ***Methods for Performance Evaluation***
  - How to obtain reliable estimates?
  - Methods for estimating a classifier's accuracy:
    - Holdout method, random subsampling
    - Cross-validation
    - Bootstrap
- ***Methods for Model Comparison***
  - How to compare the relative performance among competing models?
  - Comparing classifiers:
    - Confidence intervals
    - Cost-benefit analysis and ROC Curves

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\ Predicted class	C	¬ C	Total
C	True Positives (TP)	False Negatives (FN)	P
¬ C	False Positives (FP)	True Negatives (TN)	N
Total	P’	N’	P+N

True Positives (TP): Positive tuples correctly labeled by classifier.

True Negatives (TN): Negative tuples correctly labeled by classifier.

False Positives (FP): Negative tuples incorrectly labeled as positive by classifier.

False Negatives (FN): Positive tuples incorrectly labeled as negative by classifier.

Classifier Evaluation Metrics: Confusion Matrix

Actual class\ Predicted class	C	¬ C	Total
C	True Positives (TP)	False Negatives (FN)	P
¬ C	False Positives (FP)	True Negatives (TN)	N
Total	P'	N'	P+N

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

Classifier Evaluation Metrics:

Accuracy, Error Rate

Accuracy, ***recognition rate***: percentage of test set tuples that are correctly classified

Accuracy = (TP + TN) / (TP+FP+TN+FN)

Error Rate, ***misclassification rate***: 1 – Accuracy

Error Rate = (FP + FN) / (TP+FP+TN+FN)

Actual\ Predicted	C	¬ C
C	TP	FN
¬ C	FP	TN

Classifier Evaluation Metrics:

Sensitivity, Specificity

Class Imbalance Problem: *Main class of interest is rare.*

- Data set distribution reflects: majority ⇒ negative class and minority ⇒ positive class
  - In medical data, there may be a rare class, such as "cancer."
- If there is a class imbalance problem, ***Accuracy*** is NOT a good evaluation metric.
  - If only 3% of training tuples are actually cancer, 97% accuracy is NOT acceptable.
    - the classifier could be correctly labeling only the noncancer tuples, for instance, and misclassifying all the cancer tuples.
    - Instead, we need other measures, which access how well the classifier can recognize the positive tuples(cancer=yes) and how well it can recognize the negative tuple(cancer=no).

Sensitivity: ***True Positive recognition rate, Recall***

Sensitivity = TP / (TP+FN)

Specificity: ***True Negative recognition rate***

Specificity = TN / (TN+FP)

Actual\ Predicted	C	¬ C
C	TP	FN
¬ C	FP	TN

Classifier Evaluation Metrics:

Precision, Recall, F-measure

Precision: a **measure of exactness**

- what % of tuples that the classifier labeled as positive are actually positive

Precision = TP / (TP+FP)

Recall: a ***measure of completeness, Sensivity***

- what % of positive tuples did the classifier label as positive?

Recall = TP / (TP+FN)

Actual\ Predicted	C	¬ C
C	TP	FN
¬ C	FP	TN

- Inverse relationship between precision & recall

F-measure, F_1: harmonic mean of precision and recall,

F-measure = (2*Precision*Recall) / (Precision+Recall)

F_β: weighted measure of precision and recall

- assigns β times as much weight to recall as to precision

$F_\beta = (1+ \beta^2)*Precision*Recall) / (\beta^2 *Precision + Recall)$

Classifier Evaluation Metrics: Example

Actual Class\Predicted Class	cancer = yes	cancer = no	Total
cancer = yes	**90**	**210**	300
cancer = no	**140**	**9560**	9700
Total	230	9770	10000

Accuracy =

Error rate =

Sensivity =

Specificity =

Precision =

Recall

Classifier Evaluation Metrics: Example

Actual Class\Predicted Class	cancer = yes	cancer = no	Total
cancer = yes	**90**	**210**	300
cancer = no	**140**	**9560**	9700
Total	230	9770	10000

Accuracy = (90+9560)/10000 = 96.50%

Error rate = 1 – Accuracy = 3.50% = (140+210)/10000

Sensivity = 90 /300 = 30.00%

Specificity = 9560/9700 = 98.56%

Precision = 90/230 = 39.13%

Recall = 90/300 = 30.00%

Classifier Evaluation Metrics: Confusion Matrix

with more than two classes

Confusion Matrix:

Actual class\ Predicted class	C_1	C_2	…	C_m	Total
C_1	$CM_{1,1}$	$CM_{1,2}$	…	$CM_{1,m}$	AC_1
C_2	$CM_{2,1}$	$CM_{2,2}$	…	$CM_{2,m}$	AC_2
⋮					
C_m	$CM_{m,1}$	$CM_{m,2}$	…	$CM_{m,m}$	AC_m
Total	PC_1	PC_2		PC_m	AC_1+…+AC_m

- Given *m* classes, an entry $\mathbf{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in ***class i*** that were labeled by the classifier as ***class j.***
- For a classifier to have good accuracy, ideally most of the tuples would be represented along the diagonal of the confusion matrix, from entry $\mathbf{CM_{1,1}}$ to entry $\mathbf{CM_{m,m}}$, with the rest of the entries being zero or close to zero.

Classifier Evaluation Metrics: *with more than two classes*

Accuracy, Error Rate

Actual \ Predicted	C_1	C_2	…	C_m
C_1	$CM_{1,1}$	$CM_{1,2}$	…	$CM_{1,m}$
C_2	$CM_{2,1}$	$CM_{2,2}$	…	$CM_{2,m}$
⋮				
C_m	$CM_{m,1}$	$CM_{m,2}$	…	$CM_{m,m}$

Accuracy: Fraction of documents classified correctly

$$\textbf{Accuracy} = \frac{\sum_i CM_{i,i}}{\sum_j \sum_i CM_{i,j}}$$

Error Rate:

Error Rate = 1 – Accuracy

Classifier Evaluation Metrics: *with more than two classes*

Precision, Recall

Actual \ Predicted	C_1	C_2	...	C_m
C_1	$CM_{1,1}$	$CM_{1,2}$	...	$CM_{1,m}$
C_2	$CM_{2,1}$	$CM_{2,2}$	...	$CM_{2,m}$
⋮				
C_m	$CM_{m,1}$	$CM_{m,2}$	...	$CM_{m,m}$

Precision and Recall values for each class:

Precision: Fraction of tuples assigned class **i** that are actually about class **i**:

$$\textbf{Precision}_{C_i} = \frac{CM_{i,i}}{\sum_j CM_{j,i}}$$

Recall: Fraction of tuples in class **i** classified correctly:

$$\textbf{Recall}_{C_i} = \frac{CM_{i,i}}{\sum_j CM_{i,j}}$$

Classifier Evaluation Metrics: *with more than two classes*

Microaveraging and Macroaveraging

- In order to derive a single metric that tells us how well the system is doing, we can combine precision and recall values in two ways.

- In **macroaveraging**, compute performance for each class, and then average over classes.
- In **microaveraging**, collect decisions for all classes into a single confusion matrix, and then compute precision from that matrix.

Classifier Evaluation Metrics: *with more than two classes*

Macroaveraging

Macroaverage: compute performance for each class, and then average over classes.

Actual \ Predicted	C_1	C_2	...	C_m
C_1	$CM_{1,1}$	$CM_{1,2}$	...	$CM_{1,m}$
C_2	$CM_{2,1}$	$CM_{2,2}$	...	$CM_{2,m}$
⋮				
C_m	$CM_{m,1}$	$CM_{m,2}$	...	$CM_{m,m}$

Macroaveraging:

$$\text{Precision} = \frac{\sum_i \text{Precision}_{C_i}}{m}$$

$$\text{Recall} = \frac{\sum_i \text{Recall}_{C_i}}{m}$$

Classifier Evaluation Metrics: *with more than two classes*

Microaveraging

Microaverage: collect decisions for all classes into a single confusion matrix, and then compute precision from that matrix.

Actual \ Predicted	C_1	C_2	...	C_m
C_1	$CM_{1,1}$	$CM_{1,2}$	...	$CM_{1,m}$
C_2	$CM_{2,1}$	$CM_{2,2}$	...	$CM_{2,m}$
⋮				
C_m	$CM_{m,1}$	$CM_{m,2}$	...	$CM_{m,m}$

Confusion Matrix for C_i:

Actual \ Predicted	yes	no
yes	$TP_i = CM_{i,i}$	$FN_i = \sum_j CM_{i,j} - CM_{i,i}$
no	$FP_i = \sum_j CM_{j,i} - CM_{i,i}$	$TN_i = \sum_j \sum_i CM_{i,j} - FP_i - FN_i + CM_{i,i}$

Confusion Matrix for all classes:

Actual \ Predicted	yes	no
yes	$TP = \sum_i TP_i$	$FN = \sum_i FN_i$
no	$FP = \sum_i FP_i$	$TN = \sum_i TN_i$

Classifier Evaluation Metrics:

with more than two classes - Example

Confusion matrix for a three-class classification for e-mail classification task

Accuracy = 140/200 = 70%

Actual \ Predicted	urgent	normal	spam
urgent	30	7	3
normal	10	40	20
spam	5	15	70

Confusion Matrix for urgent:

Actual \ Predicted	yes	no
yes	30	10
no	15	145

Confusion Matrix for normal:

Actual \ Predicted	yes	no
yes	40	30
no	22	108

Confusion Matrix for spam:

Actual \ Predicted	yes	no
yes	70	20
no	23	87

Confusion Matrix for all classes:

Actual \ Predicted	yes	no
yes	140	60
no	60	340

Classifier Evaluation Metrics:

with more than two classes - Example

Confusion Matrix for urgent:

Actual \ Predicted	yes	no
yes	30	10
no	15	145

Confusion Matrix for normal:

Actual \ Predicted	yes	no
yes	40	30
no	22	108

Confusion Matrix for spam:

Actual \ Predicted	yes	no
yes	70	20
no	23	87

$Precision_{urgent}$=30/45=.67

$Precision_{normal}$=40/62=.65

$Precision_{spam}$=70/93=.75

Macroaverage Precision = (.67+.65+.75) / 3 = .69

Confusion Matrix for all classes:

Actual \ Predicted	yes	no
yes	140	60
no	60	340

Microaverage Precision = 140 / 200 = .70

Predictor Error Measures

- If *a predictor returns a continuous value* rather than a categorical label, it is difficult to say exactly whether the predicted value is correct or not.
  - Instead of focusing on whether the predicted value is an "exact" match with the correct value, we look at how far off the predicted value is from the actual value.
- **Error Functions**: $\mathbf{y_i}$ is the actualvalue, $\mathbf{y_i}$**'** is the predicted value

$$\text{Mean absolute error}: \quad \frac{\sum_{i=1}^{d} |y_i - y_i'|}{d}$$

$$\text{Mean squared error}: \quad \frac{\sum_{i=1}^{d} (y_i - y_i')^2}{d}$$

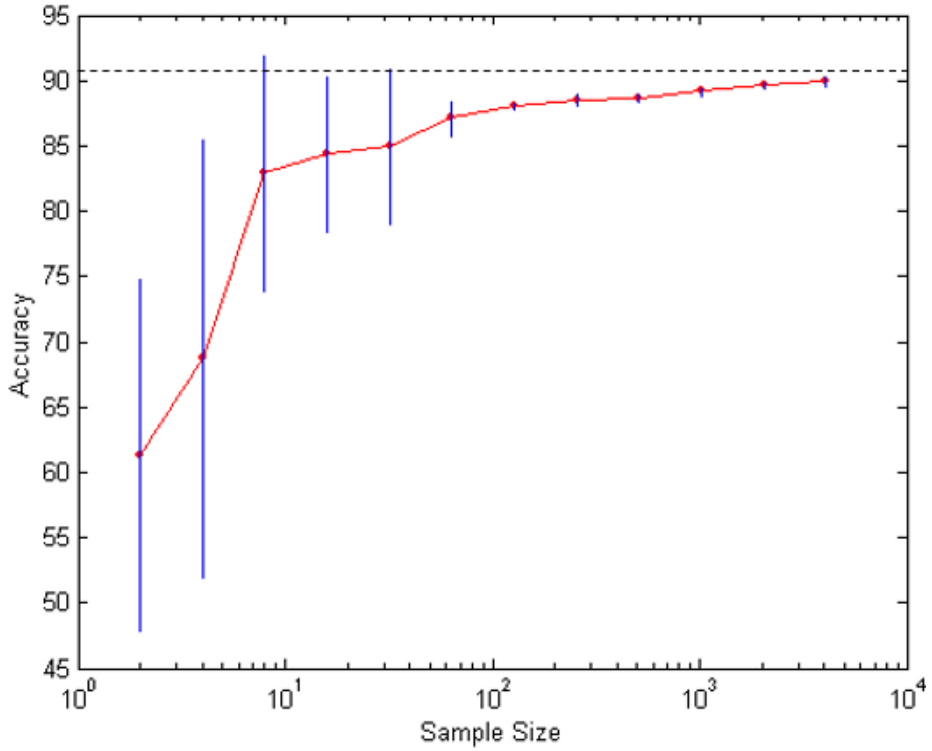
- The mean squared error exaggerates the presence of outliers, while the **mean absolute error** does not.
- If we were to take the square root of the mean squared error, the resulting error measure is called the **root mean squared error**.

Model Evaluation and Selection

- *Metrics for Performance Evaluation*
- ***Methods for Performance Evaluation***
- *Methods for Model Comparison*

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.



shows how **learning curve** accuracy changes with varying sample size

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

Evaluating Classifier Performance: Holdout

- The purpose of Evaluating Classifier Performance is to estimate the performance of a classifier on previously unseen data (test set)

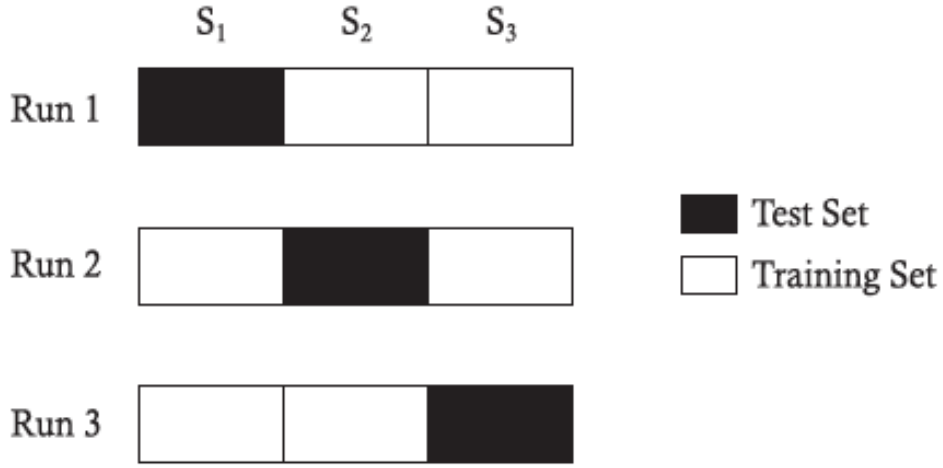
- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation

- **Random sampling**: a variation of holdout
  - Repeat holdout k times,
  - accuracy = avg. of the accuracies obtained

Evaluating Classifier Performance: Cross-Validation

- **Cross-validation** (k-fold, where k = 10 is most popular)
  - Randomly partition the data into ***k mutually exclusive*** subsets, each approximately equal size
  - At **i-th** iteration, use D_i as test set and other k-1 subsets as training set
  - The **accuracy estimate** is the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data.

- ***Example: 3-fold cross-validation***



Evaluating Classifier Performance: Cross-Validation

Variations of Cross-Validation:

- **Leave-one-out**:
  - **k** folds where **k = # of tuples**, for small sized data
  - i.e. If there are n tuples in data set, one tuple is used as test data and the rest n-1 tuples are used as training data in each iteration.

- **Stratified cross-validation**:
  - Folds are **stratified** so that class distribution in each fold is approximately the same as that in the initial data
  - Example:
    - Initial data contains 3000 tuples, and 600 tuples are positive (2400 tuples are negative).
    - In 3-fold cross validation, each subset will randomly get 200 positive tuples and 800 negative tuples.

Evaluating Classifier Performance: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Tuples are randomly selected from initial data to create training data.
    - Each time a tuple is selected, it is equally likely to be selected again and re-added to training set
- There are several bootstrap methods, and a common one is **.632 Boostrap**
  - A data set with **d** tuples is sampled **d** times, with replacement.
  - The data tuples that did not make it into the training set end up forming the test set.
  - About **63.2%** of the original data end up in the bootstrap (training data), and the remaining 36.8% form the test set.
    - Each tuple has a probability of **1/d** of being selected, so the probability of not being chosen is 1-1/d. Since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure **k** times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

Model Evaluation and Selection

- *Metrics for Performance Evaluation*
- *Methods for Performance Evaluation*
- ***Methods for Model Comparison***

Model Selection using Statistical Tests of Significance

- Suppose we have 2 classifiers, M_1 and M_2, which one is better?
- Use 10-fold cross-validation to obtain ***mean error rates*** of models M_1 and M_2 :
- These mean error rates are just *estimates* of error on the true population of *future* data cases
  - Although the mean error rates obtained for M_1 and M_2 may appear different, that difference may NOT be statistically significant.
- What if the difference between the 2 error rates is just attributed to ***chance***?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation.
  - Each partitioning is independently drawn.
  - Average 10 error rates obtained for M_1 and M_2, are their mean error rates.
- Assume samples follow a **t-distribution with k–1 degrees of freedom** (where k=10)
  - Individual error rates calculated in cross-validations may be considered as independent samples from a probability distribution.
  - In general, they follow a ***t-distribution with k-1 degrees of freedom*** where k=10.
- Use significance test **t-test** (or **Student's t-test**) to see the difference between two models is statistically significant or not.
- **Null Hypothesis**: M_1 & M_2 are the same
- If we can *Reject* **null hypothesis**, then
  - Conclude that the difference between M_1 & M_2 is **statistically significant**
  - Chose model with lower error rate

Estimating Confidence Intervals: t-test

- If only 1 test set available: **pairwise comparison**
  - For i^{th} round of 10-fold cross-validation, the same cross partitioning is used to obtain $\mathbf{err(M_1)_i}$ and $\mathbf{err(M_2)_i}$
  - Average over 10 rounds to get $\overline{\mathbf{err}}(\mathbf{M_1})$ and $\overline{\mathbf{err}}(\mathbf{M_2})$
  - **t-test** computes **t-statistic** with **k-1 degrees of freedom:** (k is 10 here)

$$\mathbf{t} = \frac{\overline{\mathbf{err}}(\mathbf{M_1}) - \overline{\mathbf{err}}(\mathbf{M_2})}{\sqrt{\mathbf{var}(\mathbf{M_1} - \mathbf{M_2})/\mathbf{k}}}$$

$$\mathbf{var}(\mathbf{M_1} - \mathbf{M_2}) = \frac{\mathbf{1}}{\mathbf{k}} \sum_{\mathbf{i=1}}^{\mathbf{k}} [(\mathbf{err}(\mathbf{M_1})_\mathbf{i} - \mathbf{err}(\mathbf{M_2})_\mathbf{i}) - (\overline{\mathbf{err}}(\mathbf{M_1}) - \overline{\mathbf{err}}(\mathbf{M_2}))]^\mathbf{2}$$

Estimating Confidence Intervals: Statistical Significance

- Are M_1 & M_2 **significantly different**?
  - Compute **t**.
  - Select **significance level** (e.g. *significance = 5%)*
  - Consult **table for t-distribution**:
    - Find t value corresponding to k-1 degrees of freedom (here, 9)
    - **t-distribution** is *symmetric*: typically upper % points of distribution shown

    $\Rightarrow$ look up the **value for confidence limit *z*** with **significance */2*** (here, 0.025) in table for t-distribution
  - If **(t > z or t < -z)**, then **t value lies in rejection region**:
    - **Reject null hypothesis** that mean error rates of M_1 & M_2 are same
      - Conclude: **statistically significant** difference between M_1 & M_2
    - **Otherwise**, conclude that any difference is **chance**

Estimating Confidence Intervals: Table for t-distribution

- Significance level, e.g., significance = 0.05 or 5% means M_1 & M_2 are significantly different for 95% of population
- Confidence limit z = significance / 2

Percentage Points $t_{\alpha,\nu}$ of the t Distribution

ν \ α	.40	.25	.10	.05	.025	.01	.005	.0025	.001	.0005
1	.325	1.000	3.078	6.314	12.706	31.821	63.657	127.32	318.31	636.62
2	.289	.816	1.886	2.920	4.303	6.965	9.925	14.089	23.326	31.598
3	.277	.765	1.638	2.353	3.182	4.541	5.841	7.453	10.213	12.924
4	.271	.741	1.533	2.132	2.776	3.747	4.604	5.598	7.173	8.610
5	.267	.727	1.476	2.015	2.571	3.365	4.032	4.773	5.893	6.869
6	.265	.718	1.440	1.943	2.447	3.143	3.707	4.317	5.208	5.959
7	.263	.711	1.415	1.895	2.365	2.998	3.499	4.029	4.785	5.408
8	.262	.706	1.397	1.860	2.306	2.896	3.355	3.833	4.501	5.041
9	.261	.703	1.383	1.833	2.262	2.821	3.250	3.690	4.297	4.781
10	.260	.700	1.372	1.812	2.228	2.764	3.169	3.581	4.144	4.587
40	.255	.681	1.303	1.684	2.021	2.423	2.704	2.971	3.307	3.551
60	.254	.679	1.296	1.671	2.000	2.390	2.660	2.915	3.232	3.460
120	.254	.677	1.289	1.658	1.980	2.358	2.617	2.860	3.160	3.373
∞	.253	.674	1.282	1.645	1.960	2.326	2.576	2.807	3.090	3.291

ν = degrees of freedom.

Estimating Confidence Intervals: Statistical Significance - Example

- Results of 10-fold cross validations

	err(M1)	err(M2)
1	0.090	0.050
2	0.130	0.100
3	0.120	0.070
4	0.150	0.110
5	0.080	0.030
6	0.120	0.080
7	0.060	0.020
8	0.140	0.120
9	0.180	0.150
10	0.170	0.140
avg	**0.124**	**0.087**

var(m1-m2)	**0.000**
t	**13.000**

$$\mathbf{var(M_1 - M_2)} = \frac{1}{k}\sum_{i=1}^{k}[(\mathbf{err(M_1)_i - err(M_2)_i}) - (\overline{\mathbf{err}}\mathbf{(M_1)} - \overline{\mathbf{err}}\mathbf{(M_2)})]^2$$

$$\mathbf{t} = \frac{\overline{\mathbf{err}}\mathbf{(M_1)} - \overline{\mathbf{err}}\mathbf{(M_2)}}{\sqrt{\mathbf{var(M_1 - M_2)/k}}}$$

- **significance=5%,** M_1 & M_2 are *significantly different* for 95% of population
- **value for confidence limit *z* with significance/*2*** from ***table for t-distribution***
  ***z* = 2.262**

t > z (13 > 2.262)

➔ **statistically significant** difference between M_1 & M_2

Estimating Confidence Intervals: Statistical Significance - Example

- Results of 10-fold cross validations

	err(M1)	err(M2)
1	0.050	0.060
2	0.100	0.090
3	0.070	0.060
4	0.110	0.100
5	0.030	0.050
6	0.080	0.090
7	0.020	0.010
8	0.120	0.110
9	0.150	0.130
10	0.140	0.150
avg	**0.087**	**0.085**

var(m1-m2)	**0.002**
t	**0.154**

$$\mathbf{var(M_1 - M_2)} = \frac{\mathbf{1}}{\mathbf{k}} \sum_{\mathbf{i=1}}^{\mathbf{k}} [(\mathbf{err(M_1)_i} - \mathbf{err(M_2)_i}) - (\overline{\mathbf{err}}\mathbf{(M_1)} - \overline{\mathbf{err}}\mathbf{(M_2)})]^{\mathbf{2}}$$

$$\mathbf{t} = \frac{\overline{\mathbf{err}}\mathbf{(M_1)} - \overline{\mathbf{err}}\mathbf{(M_2)}}{\sqrt{\mathbf{var(M_1 - M_2)/k}}}$$

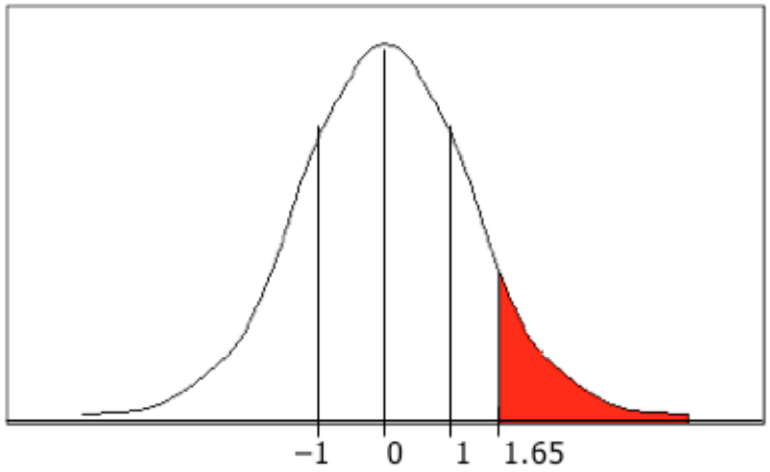
- **significance=5%,** M_1 & M_2 are *significantly different* for 95% of population
- **value for confidence limit *z* with significance/*2*** from ***table for t-distribution***
  ***z* = 2.262**

t ⊁ z (0.154 ⊁ 2.262)

➔ **no difference** between M_1 & M_2

Confidence Limits

- Confidence limits for the normal distribution with 0 mean and a variance of 1:



Pr[X ≥ z]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

- Thus, a confidence limit for 10% significance (i.e. significance/2 = 5%)

  P(-1.65 ≤ X ≤ 1.65) = 90%

  – This means that 90% of the population in this range.

Confidence Interval for Accuracy

standard normal distribution of accuracy

- For large data sets (N>100), the **accuracy has a normal distribution** with **mean p** and **variance p(1-p)/N.**

- Transformed value for the accuracy **acc:** $\frac{\mathbf{acc}}{\sqrt{\mathbf{p(1-p)/N}}}$
  - in order to have the normal distribution with 0 mean and a variance of 1:

- Confidence limits for **significance α:** $\mathbf{P(-Z_{\alpha/2} < \frac{acc}{\sqrt{p(1-p)/N}} < Z_{\alpha/2}) = 1-\alpha}$

- Solving the equation above for p yields **Confidence Interval for p**:

$$\mathbf{p = \frac{2*N*acc + Z^2 \pm \sqrt{Z^2 + 4*N*acc - 4*N*acc^2}}{2*(N+Z^2)}}$$

where $\mathbf{Z}$ is $\mathbf{Z_{\alpha/2}}$

Confidence Interval for Accuracy - Example

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
  - N=100, accuracy acc=0.8
  - Let 1-α=0.95 (95% confidence)
  - From probability table $\mathbf{Z_{\alpha/2}=1.96}$ (where significance α = 5%)

N	100	500	1000	5000
Mean p lower	0.747	0.779	0.785	0.794
Mean p upper	0.830	0.817	0.812	0.806

Comparing Performance of 2 Models

- Given two models, say, M1 and M2, which one is better?
  - M1 is tested on data set D1 (size=n_1), found error rate = e_1 .
  - M2 is tested on data set D2 (size=n_2), found error rate = e_2 .
  - Assume D1 and D2 are independent.
  - If n_1 and n_2 are sufficiently large, then

$$\mathbf{e_1 \sim NormalDist(\mu_1, \sigma_1)}$$

$$\mathbf{e_2 \sim NormalDist(\mu_2, \sigma_2)}$$

  - Approximate variance: $\hat{\sigma}_i^2 = \frac{e_i(1-e_i)}{n_i}$

Comparing Performance of 2 Models

- To test if performance difference is significant: $\mathbf{d = e_1 - e_2}$
  - $\mathbf{d \sim NormalDist(d_t, \sigma_t)}$ where $\mathbf{d_t}$ is the true difference
  - Since D1 and D2 are independent, their variance adds up:

$$\boldsymbol{\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_2^2}$$

$$\boldsymbol{\hat{\sigma}_t^2 = \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}}$$

  - At (1-α) confidence level, true difference range: $\mathbf{d_t = d \pm Z_{\alpha/2} \hat{\sigma}_t}$

Comparing Performance of 2 Models - Example

- Given: M1: n1=30 e1=0.15
  M2: n2=5000 e2=0.25
- d = |e2-e1| = 0.1

$$\hat{\sigma}_t^2 = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

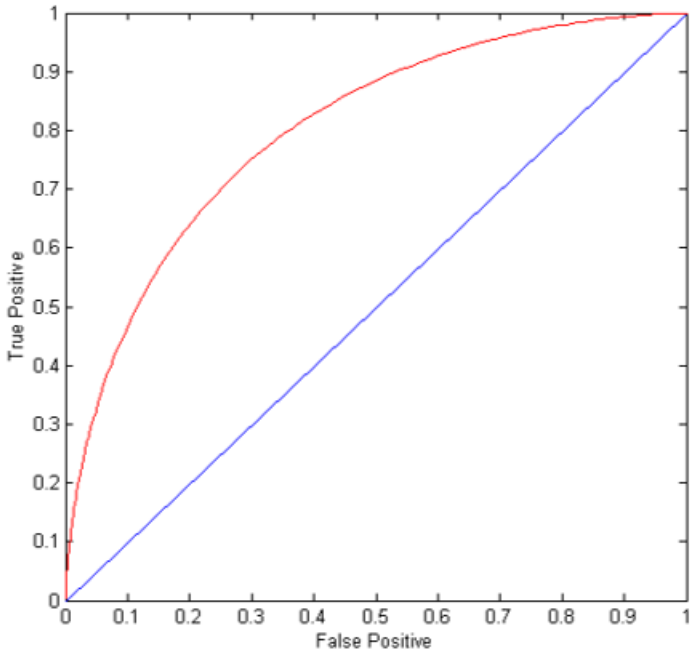
- At 95% confidence level: $Z_{\alpha/2}$=1.96

$$d_t = 0.1 \pm 1.96 * \sqrt{0.0043} = 0.1 \pm 0.128$$

➔ **Since interval contains 0, difference may NOT be statistically significant.**

Model Selection: ROC Curves
ROC (Receiver Operating Characteristics)

- **ROC curves**: for visual comparison of classification models
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- ROC curve characterizes the trade-off between positive hits and false alarms
- ROC curve plots TP rate (on the y-axis) against FP rate (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve

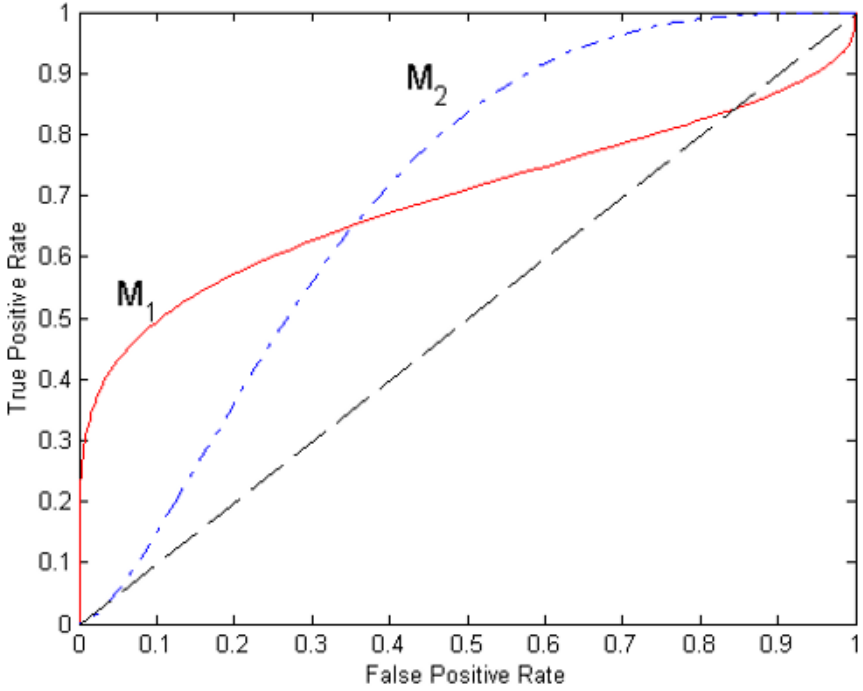


(TP,FP):

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

- Diagonal line: Random guessing

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

How to Construct an ROC curve

Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

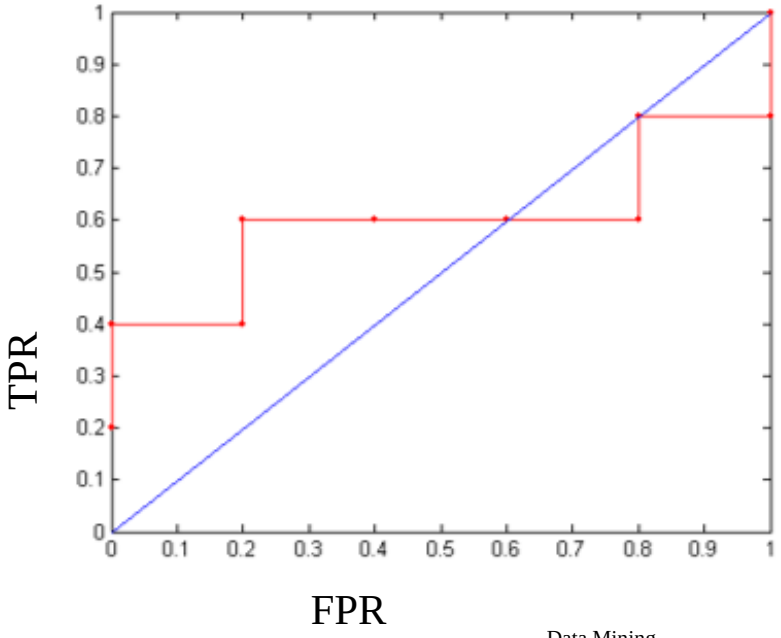
- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP+TN)

How to Construct an ROC curve

	Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=		0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
	TP	5	4	4	3	3	3	3	2	2	1	0
	FP	5	5	4	4	3	2	1	1	0	0	0
	TN	0	0	1	1	2	3	4	4	5	5	5
	FN	0	1	1	2	2	2	2	3	3	4	5
→	TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
→	FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

TPR = TP/(TP+FN)
FPR = FP/(FP+TN)

ROC Curve:



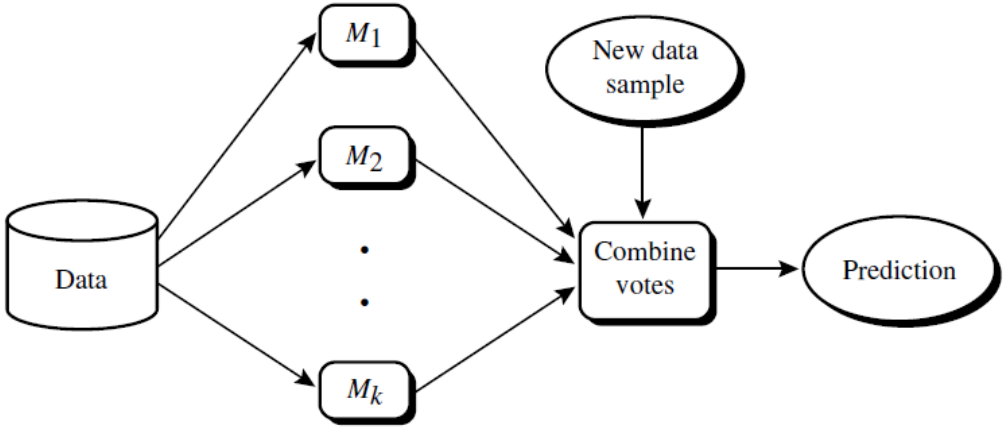
Issues Affecting Model Selection

- **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
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Ensemble Methods: Increasing the Accuracy

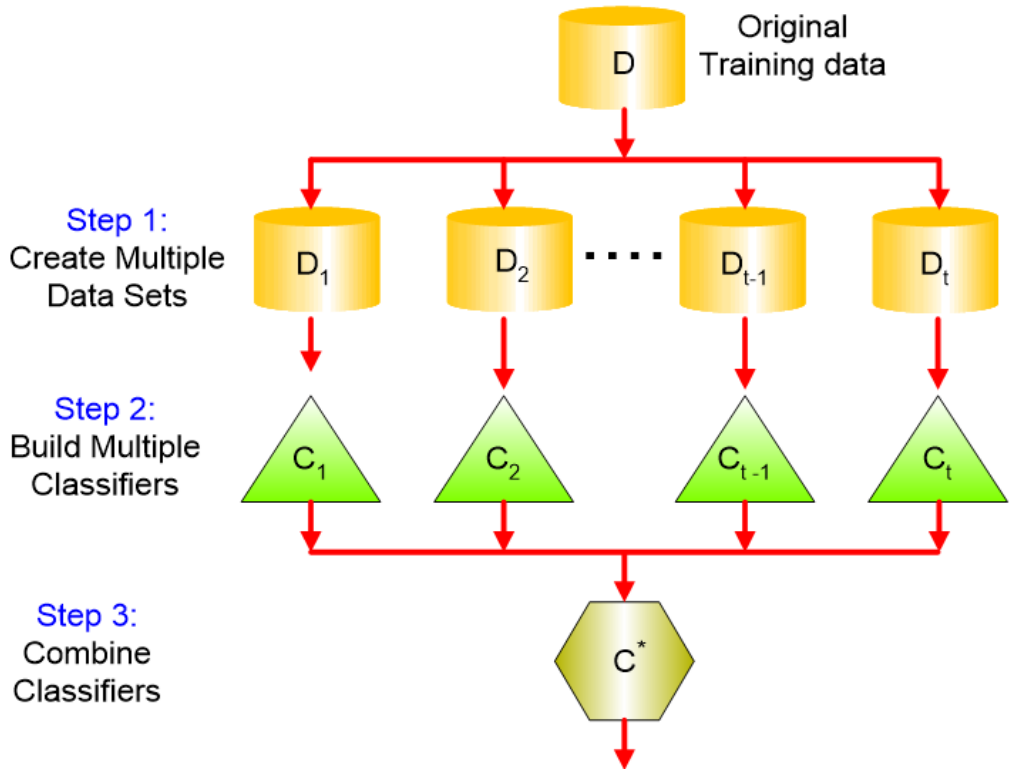
Ensemble Methods: Increasing the Accuracy

- ***Ensemble methods***
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - **Bagging**: averaging the prediction over a collection of classifiers
  - **Boosting**: weighted vote with a collection of classifiers
  - **Ensemble**: combining a set of heterogeneous classifiers



Ensemble Methods

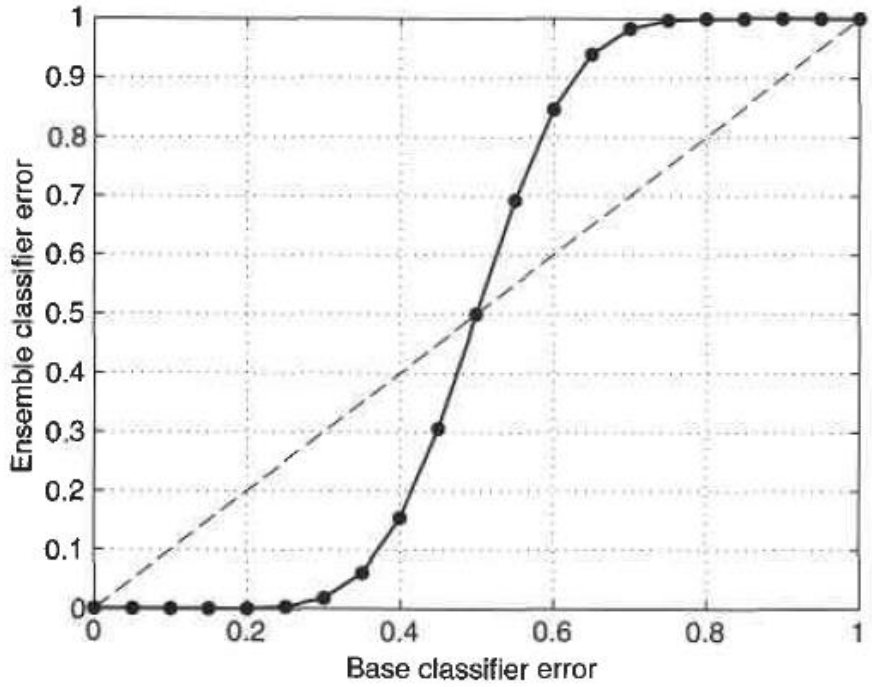
- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers



Why does Ensemble Classifier work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$



Methods for Constructing an Ensemble Classifier

- The ensemble of classifiers can be constructed in many ways:
- ***By manipulating the training set.***
  - Multiple training sets are created by resampling the original data.
  - A classifier is then built from each training set using a particular learning algorithm.
  - **Bagging** and **boosting** are two examples of ensemble methods that manipulate their training sets
- ***By manipulating the input features.***
  - A subset of input features is chosen to form each training set.
  - This approach works well with data sets that contain highly redundant features.
  - **Random forest** is an ensemble method that manipulates its input features and uses decision trees as its base classifiers.

Bagging: Boostrap Aggregation

- ***Analogy***: Diagnosis based on multiple doctors' majority vote
- ***Training***
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- ***Classification***: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- ***Prediction***: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- ***Accuracy***
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Bagging

- **Bagging** is a technique that repeatedly samples (with replacement) from a data set according to a uniform probability distribution.
- Each bootstrap sample has the same size as the original data.
  - Because each sample has a probability $1-(1-1/N/)^N$ of being selected in each Di, a sample Di contains approximately 63% of the original training data.
  - After training the k classifiers, a test instance is assigned to the class that receives the highest number of votes.
- Bagging improves generalization error by reducing the variance of the base classifiers.
  - The performance of bagging depends on the stability of the base classifier.
  - If a base classifier is unstable, bagging helps to reduce the errors associated with random fluctuations in the training data.
  - If a base classifier is stable, bagging may not be able to improve the performance of the base classifiers.

Bagging - Example

- We create a single inner node decision tree classifiers for the following training data. That tree is also known as a ***decision stump***.

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	−1	−1	−1	−1	1	1	1

- Any decision tree with a single inner node can have maximum %70 accuracy for this training set.
  - Attribute >35 produces %70 accuracy
  - Attribute >75 produces %70 accuracy

Bagging - Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

- 10 decision trees with single inner nodes are generated.
- Each bootstrap sample has the same size as the original data and randomly generated from the original data.

Bagging - Example

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

The performance is increased by bagging approach

Boosting

- ***Analogy***: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost

- In the AdaBoost algorithm, the importance of a base classifier C_i depends on its error rate, which is defined as

$$\epsilon_i = \frac{1}{N}\left[\sum_{j=1}^{N} w_j \, I\left(C_i(\mathbf{x}_j) \neq y_j\right)\right]$$

where I(p) = 1 if the predicate p is true, and 0 otherwise.

- The importance of a classifier Ci is given by the following parameter,

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$$

Adaboost

- Given a set of **N** class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_N}$, y_N)
- Initially, all the weights of tuples are set the same (**1/N**)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$\epsilon_i = \frac{1}{N}\left[\sum_{j=1}^{N} w_j \, I\left(C_i(\mathbf{x}_j) \neq y_j\right)\right]$$

- The importance of classifier M_i's vote:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$$

- Weight update mechanism:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(\mathbf{x_i}) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(\mathbf{x_i}) \neq y_i \end{cases}$$

  - where Z_j is the normalization factor used to ensure that $\sum_i w_i^{(j+1)} = 1$

Adaboost -Example

Training records

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	−1	−1	−1	−1	1	1	1

Training records chosen during boosting

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

Weights of training records

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

Random Forest

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - **Forest-RI (*random input selection*):** Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - **Forest-RC (*random linear combinations*):** Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Empirical Comparison among Ensemble Methods

Data Set	Number of (Attributes, Classes, Records)	Decision Tree (%)	Bagging (%)	Boosting (%)	RF (%)
Anneal	(39, 6, 898)	92.09	94.43	95.43	95.43
Australia	(15, 2, 690)	85.51	87.10	85.22	85.80
Auto	(26, 7, 205)	81.95	85.37	85.37	84.39
Breast	(11, 2, 699)	95.14	96.42	97.28	96.14
Cleve	(14, 2, 303)	76.24	81.52	82.18	82.18
Credit	(16, 2, 690)	85.8	86.23	86.09	85.8
Diabetes	(9, 2, 768)	72.40	76.30	73.18	75.13
German	(21, 2, 1000)	70.90	73.40	73.00	74.5
Glass	(10, 7, 214)	67.29	76.17	77.57	78.04
Heart	(14, 2, 270)	80.00	81.48	80.74	83.33
Hepatitis	(20, 2, 155)	81.94	81.29	83.87	83.23
Horse	(23, 2, 368)	85.33	85.87	81.25	85.33
Ionosphere	(35, 2, 351)	89.17	92.02	93.73	93.45
Iris	(5, 3, 150)	94.67	94.67	94.00	93.33
Labor	(17, 2, 57)	78.95	84.21	89.47	84.21
Led7	(8, 10, 3200)	73.34	73.66	73.34	73.06
Lymphography	(19, 4, 148)	77.03	79.05	85.14	82.43
Pima	(9, 2, 768)	74.35	76.69	73.44	77.60
Sonar	(61, 2, 208)	78.85	78.85	84.62	85.58
Tic-tac-toe	(10, 2, 958)	83.72	93.84	98.54	95.82
Vehicle	(19, 4, 846)	71.04	74.11	78.25	74.94
Waveform	(22, 3, 5000)	76.44	83.30	83.90	84.04
Wine	(14, 3, 178)	94.38	96.07	97.75	97.75
Zoo	(17, 7, 101)	93.07	93.07	95.05	97.03

- Empirical results obtained when comparing the performance of a decision tree classifier against bagging, boosting, and random forest.
- The base classifiers used in each ensemble method consist of fifty decision trees.
- The classification accuracies reported in this table are obtained from ten-fold cross-validation.
- Notice that the ensemble classifiers generally outperform a single decision tree classifier on many of the data sets.

Summary

- ***Classification*** is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for ***decision tree induction***, ***Naive Bayesian classification***, ***rule-based classification***, and many other classification methods.
- ***Evaluation metrics*** include: accuracy, sensitivity, specificity, precision, recall, *F* measure, and $F_{ß}$ measure.
- ***Stratified k-fold cross-validation*** is recommended for accuracy estimation.

Summary

- ***Bagging and boosting*** can be used to increase overall accuracy by learning and combining a series of individual models.
- ***Significance tests*** are useful for model selection.
- There have been numerous comparisons of the different classification methods;
  - The matter remains a research topic
  - No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method