Machine Learning

Prof. Dr. Stefan Kramer
Johannes Gutenberg-Universität Mainz

Acknowledgements

- E. Frank
- I. Witten
- T. Mitchell

Outline

- Bias-Variance Decomposition
- Evaluation and Validation

Bias-Variance Decomposition for Regression

Assume $Y = f(X) + \varepsilon$, where $E(\varepsilon) = 0$, then the expression for the expected *prediction error* of a regression fit with squared-error loss is

$$PE(f(.,T)) = E_{X,Y}(Y - f(X,T))^2$$

$$PE^*(f) = \mathrm{E}_T[PE(f(.,\mathrm{T}))] =$$

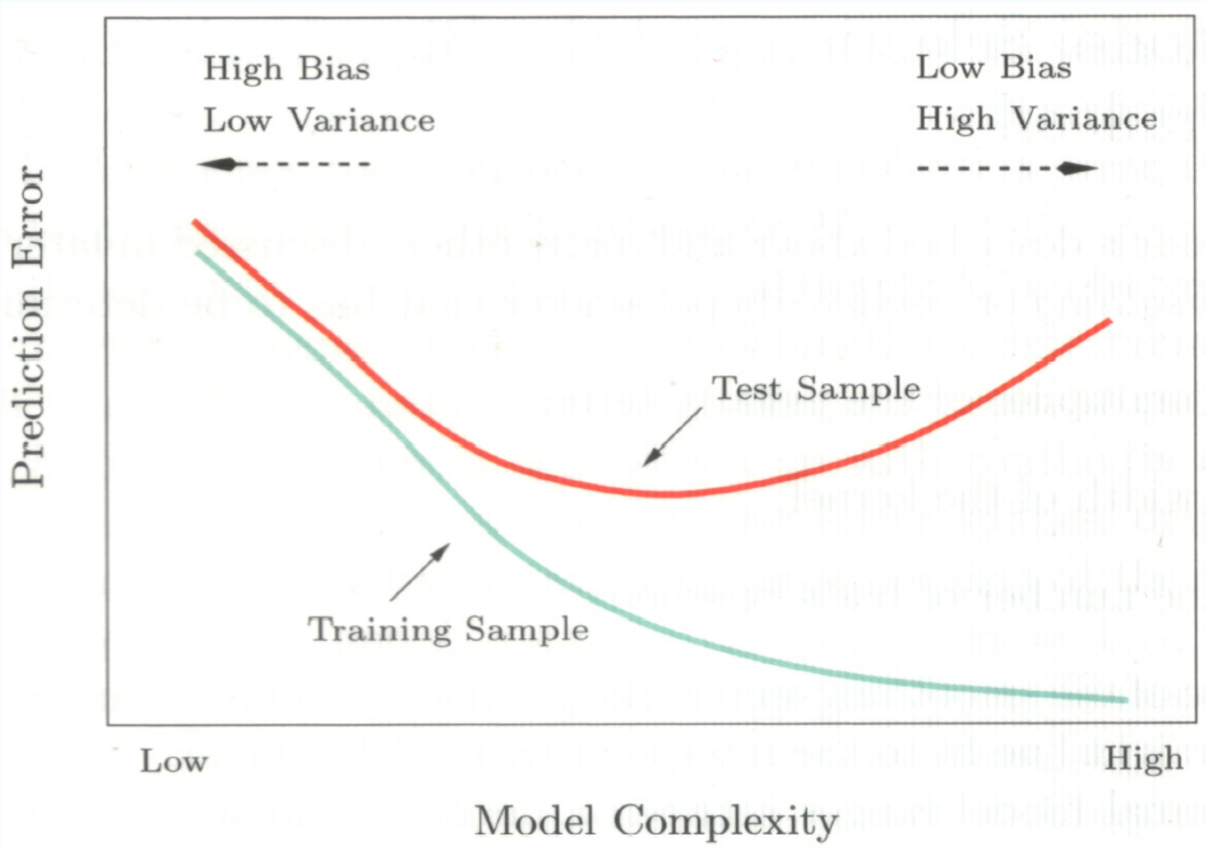
$$= E\varepsilon^2 + E_X[f^*(X) - \hat{f}(X)]^2 + E_{X,T}[f(X,T) - \hat{f}(X)]^2$$

$$= E\varepsilon^2 + \mathrm{Bias}(\hat{f}(X))^2 + \mathrm{Var}(\hat{f}(X))$$

$$= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}$$

T is a randomly drawn training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

Bias and Variance



Behavior of test sample and training sample error as the model complexity is varied

Conclusion

- Ensembles: mostly motivated by theory
- Comprehensibility suffers
- Standard methods for improving performance of relatively simple basic models

Outline

- Evaluation and validation in predictive mining
- ROC and recall-precision curves

Evaluation and Validation

Problem Setting

Training Set

c:c-c:c			c:(6)-N	c:c-N	
1					
0					
1					
…					

Test Set

c:c-c:c	c:(6)-c(6)	Br-C	Br	C-O-c:(6)-N	C-O-c:c-N	
1	0	1	1	1	1	+1
1	1	0	0	0	1	+1
0	0	0	0	0	1	-1
…	…	…	…	…	…	…

Evaluation: Key to Success

- How predictive is the model we learned?
- Error on the training data is not a good indicator of performance on future data
- Otherwise 1-NN (*rote learning*) would be the optimum classifier!
- Simple solution that can be used if lots of (labeled) data is available:
  - split data into training and test set
  - however: (labeled) data is often limited
  - more sophisticated techniques need to be used

Training and Testing

- Natural *performance measure* for classification problems is the *error rate*
  - success: instance's class is predicted correctly
  - error: instance's class is predicted incorrectly
  - error rate: proportion of errors made over the whole set of instances
- *Resubstitution error*: error rate on the training data
- Resubstitution error is *hopelessly optimistic* (think of rote learning)

Training and Testing

- Test set: set of independent instances that have played no part in formation of classifier
- Assumption: both training data and test data are *representative samples of the same underlying distribution* (i.i.d. assumption - *independent and identically distributed*)
- In practice, test and training data may differ in nature
- Example: classifiers built using measured data from two different labs A and B – to estimate performance of the classifier from lab A in completely new lab, test it on data from B

Training and Testing

- *If lots of data available, then evaluation and comparison of classifiers not a problem*
- Large, representative training and test sets
- Simple statistical significance tests can be used: McNemar and others

McNemar

n_{00} = number of examples misclassified by both f_A and f_B	n_{01} = number of examples misclassified by f_A but not by f_B
n_{10} = number of examples misclassified by f_B but not by f_A	n_{11} = number of examples misclassified by neither f_A nor f_B

- Expected frequencies under null hypothesis (classifiers f_A and f_B equally well):

n_{00}	$(n_{01} + n_{10})/2$
$(n_{01} + n_{10})/2$	n_{11}

- Test statistic: $\frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}$
- Approximately χ^2 distributed with one degree of freedom

A Note on Parameter Tuning

- It is important that the test data is not used in any way to create the classifier
- Some learning schemes operate in two stages:
  - stage 1: *learn the structure (structure learning)*
  - stage 2: *learn (tune) the parameters (parameter learning)*
- It is ***not allowed*** to use the test data for ***parameter tuning***!
- Proper procedure uses three sets: training data, validation data, and test data
- Validation data is used to optimize parameters (also used by some "pruning" methods)

Making the Most of the Data

- **Once the evaluation (estimation of error) is complete, *all the data* can be used to build the final classifier**
- The larger the training data, the better the classifier (law of diminishing returns)
- The larger the test data, the more accurate the error estimate
- Holdout procedure: method of splitting original data into training and test set
- Dilemma: ideally we want both, a large training and a large test set

Hold-Out Estimation and Stratification

- Usual procedure: use two thirds for training, one third for testing
- Problem: the samples might not be representative
- Example: class might be missing in the test data
- Use *stratification*: ensures that each class is represented with approximately equal proportions in both datasets

Repeated Hold-Out

- Holdout estimate can be made more reliable by repeating the process with different subsamples
- In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
- The error rates on the different iterations are averaged to yield an overall error rate
- This is called the *repeated holdout method*
- Still not optimal: the different test sets *overlap*
- Can we do without overlapping datasets?

Cross-Validation

- First step: data is split into k subsets of equal size
- Second step: each subset in turn is used for testing and the remainder for training
- This procedure is called *k-fold cross-validation*
- Recommended: *stratified k-fold crossvalidation*
- The error estimates are averaged to yield an overall error estimate
- Even better: *record prediction for each instance* and compare with its actual class

Cross-Validation

- Standard method: *stratified ten-fold cross-validation*
- Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate
- Stratification reduces the *variance* of the error estimate
- Best practice: repeated stratified cross-validation, for instance *ten times ten-fold stratified crossvalidation*

Leave-One-Out Cross-Validation

- Leave-one-out cross-validation is a particular form of cross-validation with the number of folds equal to the number of training instances
  - i.e., a classifier has to be built n times, where n is the number of training instances
- Measurement at a different point of the learning curve
- Leave-one-out makes maximum use of the data
- Obviously, no random subsampling involved
- However, computationally very expensive (except for some classification schemes as k-Nearest Neighbor and decision tables)

Fooling Leave-One-Out

- Extreme example: completely random dataset with two classes and equal proportions for both of them
- Best inducer predicts majority class (results in 50% predictive accuracy on fresh data from this domain)
- *However, LOO-CV* ***error*** *estimate for this inducer will be 100%*

Bootstrap

- CV uses sampling without replacement
- In contrast, the *bootstrap* uses sampling *with replacement* to form training set
- A dataset of n instances is sampled n times with replacement to form a new dataset of n instances (same size, duplicates likely) – used as training set
- Instances from original dataset *not occurring in* this ***bootstrap sample*** are used for testing

0.632 Bootstrap

- This method is also called the 0.632 bootstrap
- A particular instance has a probability of 1-1/n of not being picked
- Thus, its probability of ending up in the test data is:
$$\left(1-\frac{1}{n}\right)^n \approx e^{-1} = 0.368$$
- This means the training data will contain approximately 63.2% of the instances

Estimating Error with the Bootstrap

- The overall error estimate by the bootstrap is then a combination of
  - the (pessimistic: only ~63% of the instances seen) error on the test data and
  - the (optimistic) error on the training data

$$err = 0.632 \cdot e_{\text{test instances}} + 0.368 \cdot e_{\text{training instances}}$$

- Process is repeated several time, with different bootstrap samples, and the results averaged
- Together with LOO, best method for small datasets

Fooling the Bootstrap

- Consider again the random dataset and *rote learning*
- True expected error: 50%
- Bootstrap estimate for this classifier: ~32%
- *Why?*

ROC and Recall-Precision Curves

Counting the Costs

- In practice, different types of classification errors often incur different costs: false positive costs vs. false negative costs (vs. abstention costs)
- Examples:
  - promotional mailing
  - medical diagnosis
  - predicting toxic substance as not toxic, or vice versa

Taking Into Account Costs

Confusion matrix

		Predicted class	
		Yes	No
Actual class	Yes	True positive	False negative
	No	False positive	True negative

Lift Charts

- In practice, costs are rarely known
- Decisions are usually made by comparing possible *costs scenarios*
- A *lift chart* allows for a *visual comparison*

Story

- Result of promotional mailout: 1,000,000 households contacted, 1,000 responded
- Then a *classifier* is given (from *somewhere*), outputting for each instance (household) the probability of being positive (responsive)
- Instances are sorted according to their predicted probability of being *positive*

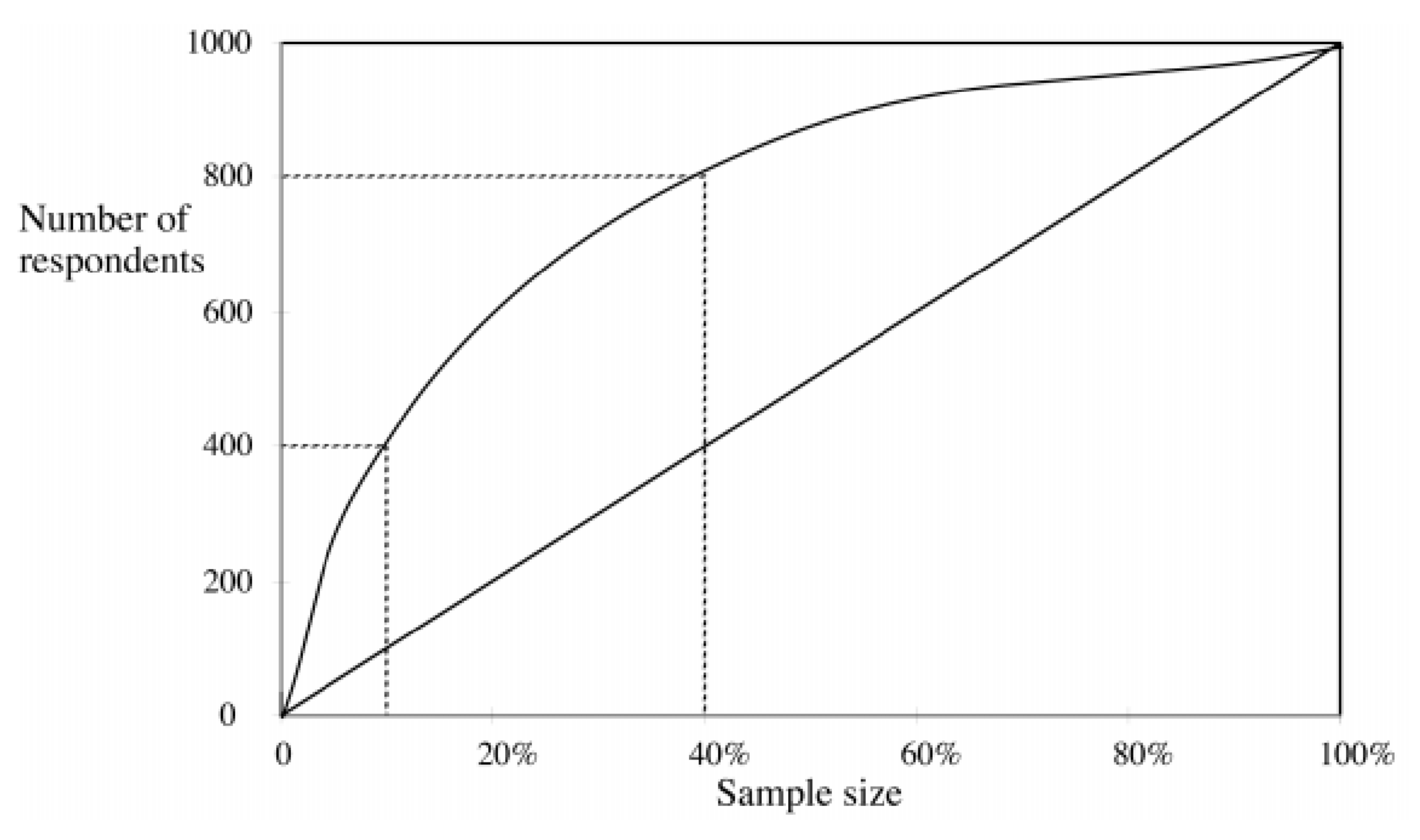
Rank	Predicted probability	Actual class
1	0.95	Yes
2	0.93	Yes
3	0.93	No
4	0.88	Yes
…	…	…

predicted positive

Generating a Lift Chart

- Then we can calculate what would happen if only the top k percent of instances in this ranking *were predicted positive* (i.e., a hypothetical mailout would go to them): what is the fraction of *real respondents* we obtain?
- In lift chart, x-axis is *sample size (percentage of complete dataset)*, and y-axis is *number of true positives*

A Hypothetical Lift Chart



Story

- 1,000,000 households contacted
- 1,000 responded
- *Situation 1*: classifier predicts that 100,000 households are responsive, and actually gets 400 actual respondents
- *Situation 2*: classifier predicts that 400,000 households are responsive, and actually gets 800 out of the possible 1000 real respondents

ROC Curves

- “ROC” stands for “receiver operating characteristic”
- Used in signal detection to show tradeoff between hit rate and false alarm rate over a noisy channel
- x axis shows percentage of false positives in sample (rather than sample size)
- y axis shows percentage of true positives in sample (rather than absolute number)

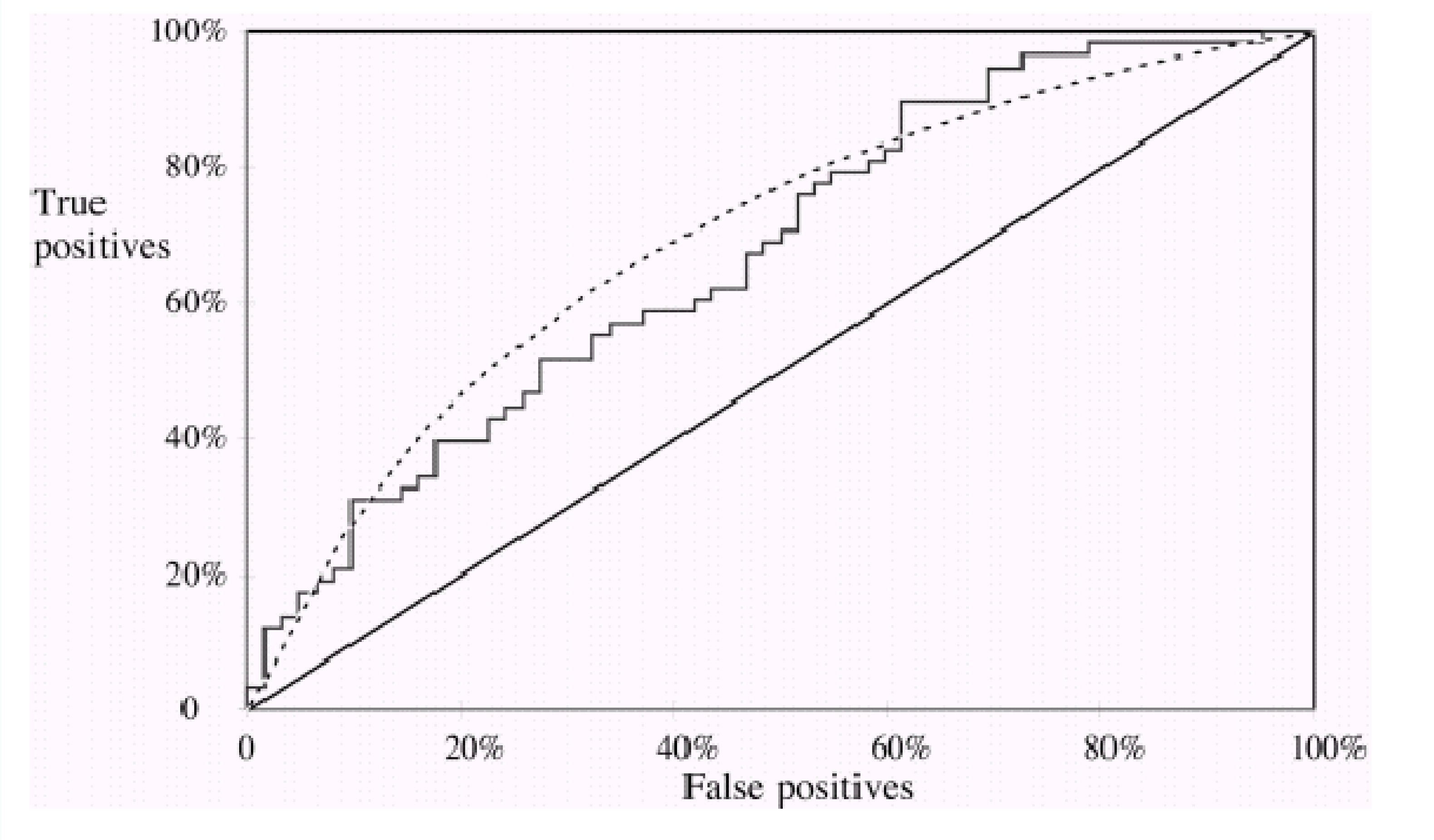
Measures and Curves

	Predicted Class			
Actual Class		Pos	Neg	
	Pos	TP	FN	P
	Neg	FP	TN	N
		PP	PN	

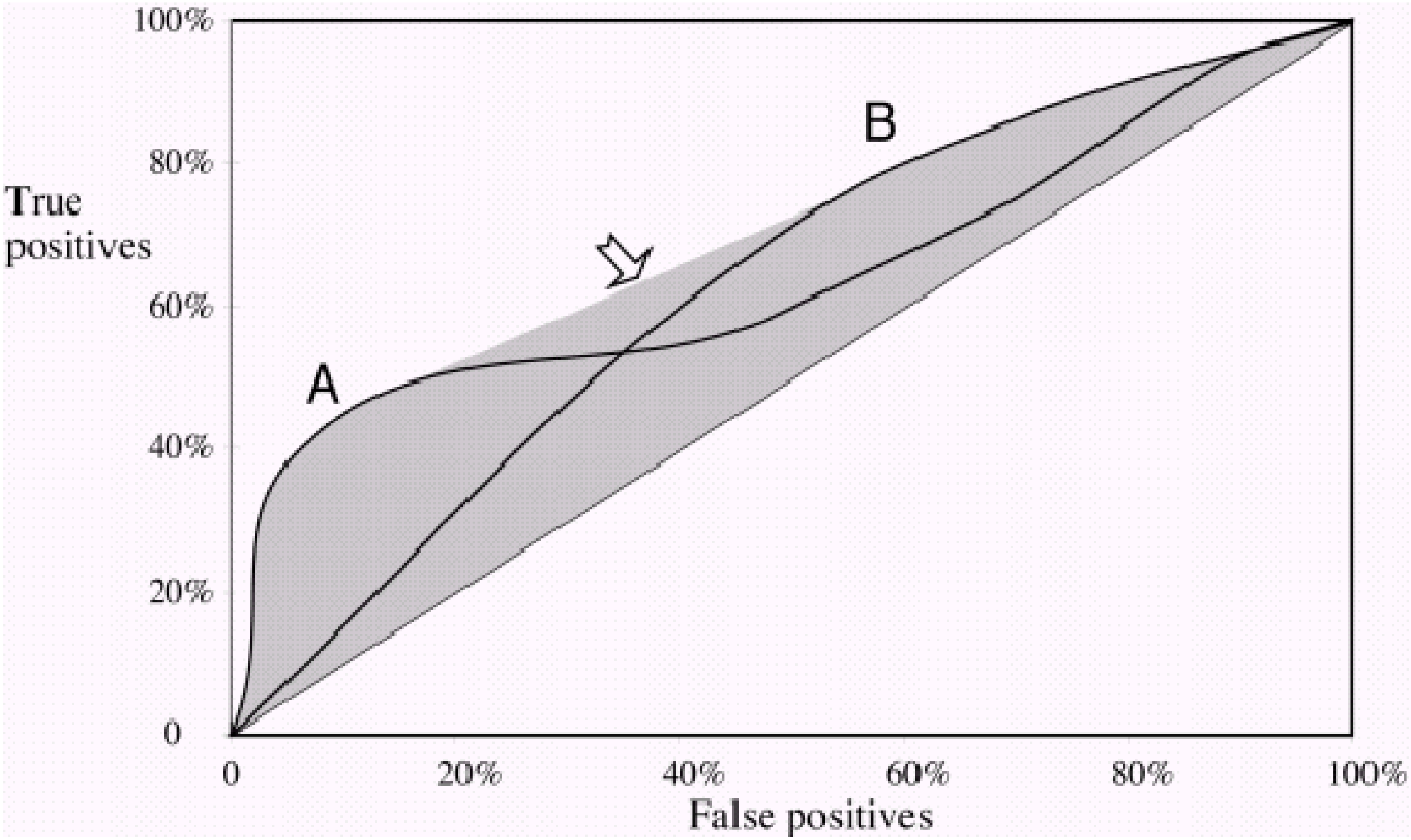
ROC Curves:
x-axis: False Positive Rate = FP /(FP + TN) = FP / N
y-axis: True Positive Rate = TP /(TP + FN) = TP / P

Recall-Precision Curves:
x-axis: Recall = TP / (TP + FN) = TP / P = TPR
y-axis: Precision = TP / (TP + FP) = TP / PP

A Sample ROC Curve



ROC Curves for Two Schemes



Area Under ROC (AUC or AUROC)

- Sometimes, the area under the ROC curve is taken as a measure of quality
- AUROC can also be represented as the ratio of the number of correct pairwise rankings vs. the number of all possible pairs, a quantity known as called the *Wilcoxon-Mann-Whitney (WMW)* statistic

Cross-Validation and ROC

- Simple method of getting a ROC curve using cross-validation:
  - collect probabilities for instances in test folds
  - sort instances according to probability of being positive
- This is the method implemented, for instance, in the WEKA workbench