Machine Learning (CS 567)

Fall 2008

Time: T-Th 5:00pm - 6:20pm
Location: GFS 118

Instructor: Sofus A. Macskassy (macskass@usc.edu)
Office: SAL 216
Office hours: by appointment

Teaching assistant: **Cheol Han (cheolhan@usc.edu)**
Office: SAL 229
Office hours: M 2-3pm, W 11-12

Class web page:
http://www-scf.usc.edu/~csci567/index.html

Administrative – 599 Spring 2009

- I teach a 599 Seminar next semester (Spring 2009)
  - Style is seminar: weekly readings, class discussion
- Title: Advanced Topics in Machine Learning: Statistical Relational Learning
  - This is not the same as what Prof. Fei Sha is teaching, which is a different advanced topics in ML seminar
- Focus of the course is on relational learning
  - Standard ML considers instances to be independent
  - What if they are not? Such as in relational databases, social networks, other graphical data such as the web or hypertext?
  - Topics include issues such as collective inference, relational inference, search space, bias, graphical models and more.
- Preliminary syllabus is posted on the csci567 page

Administrative – Project Grading

- Attendance: 5%
- Presentation: 20%
  - Clarity of presentation (what the problem is, what you did, what your results were and what they mean); did you include necessary elements.
- Paper clarity: 25%
  - Clarity of writing and are the required elements present. Also, after reading the paper, can a reader understand it well enough to replicate your work and results.
- Research: 50%
  - The amount of work you did, the quality of the work, the thoroughness of your evaluation and whether you clearly have applied what you have learned in class in the appropriate manner.

Evaluation of Classifiers

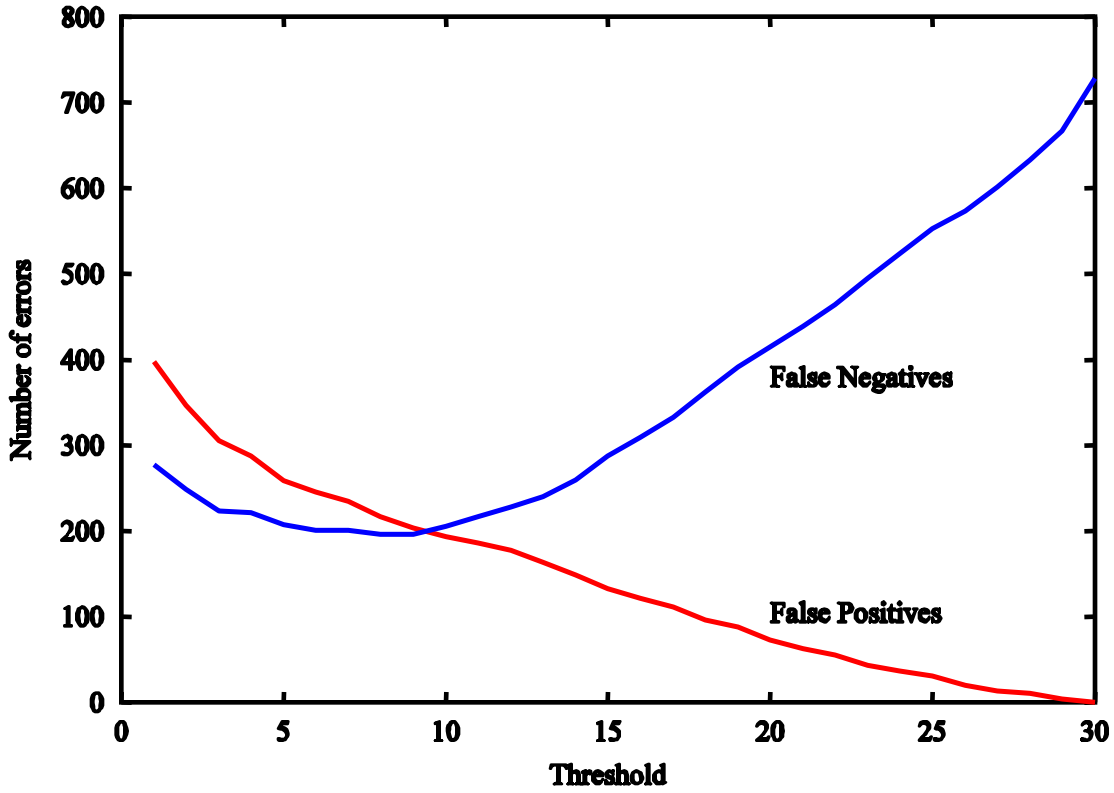
- ROC Curves
- Reject Curves
- Precision-Recall Curves
- Statistical Tests
  - Estimating the error rate of a classifier
  - Comparing two classifiers
  - Estimating the error rate of a learning algorithm
  - Comparing two algorithms

Cost-Sensitive Learning

- In most applications, false positive and false negative errors are not equally important. We therefore want to adjust the tradeoff between them. Many learning algorithms provide a way to do this:
  - probabilistic classifiers: combine cost matrix with decision theory to make classification decisions
  - discriminant functions: adjust the threshold for classifying into the positive class
  - ensembles: adjust the number of votes required to classify as positive

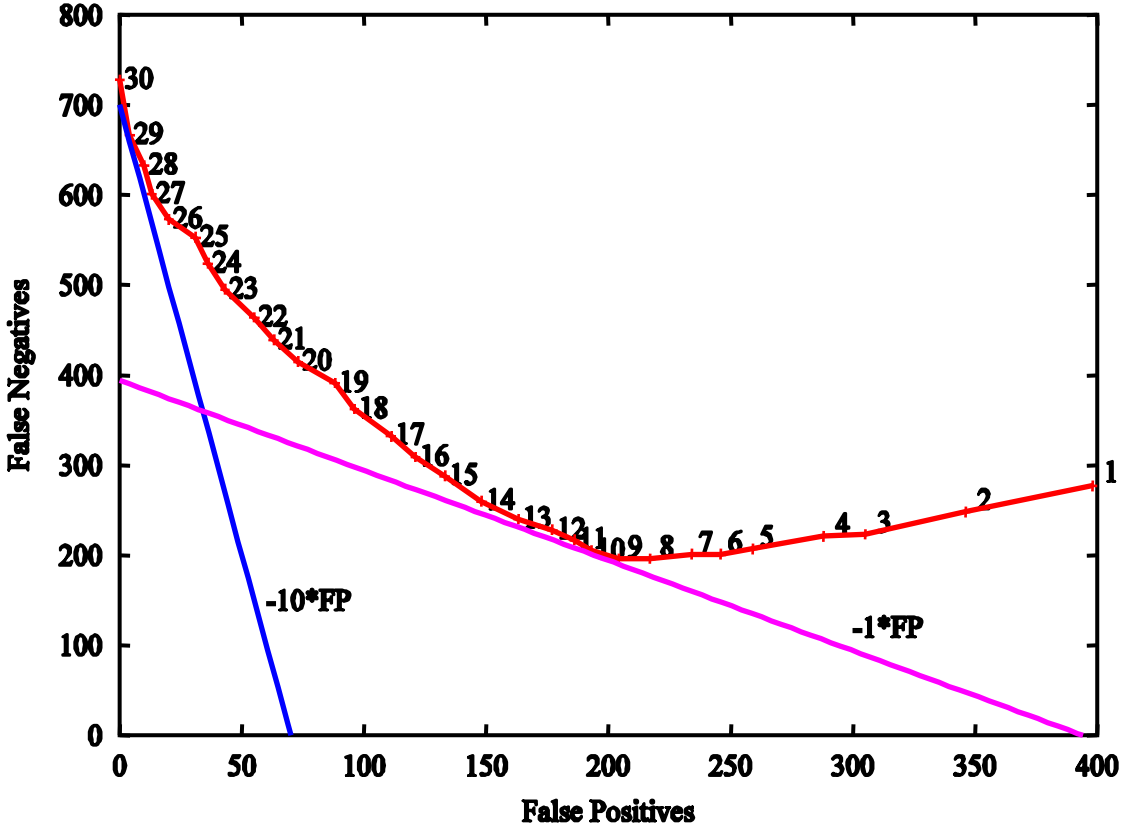
Example: 30 trees constructed by bagging

- Classify as positive if K out of 30 trees predict positive. Vary K.



Directly Visualizing the Tradeoff

- We can plot the false positives versus false negatives directly.
- If $R \cdot L(0,1) = L(1,0)$ (i.e., a FP is R times more expensive than a FN), then total errors = $L(0,1)*FN(\theta) + L(1,0)*FP(\theta) \propto FN(\theta) + R*FP(\theta)$; $FN(\theta) = -R*FP(\theta)$
- The best operating point will be tangent to a line with a slope of –R

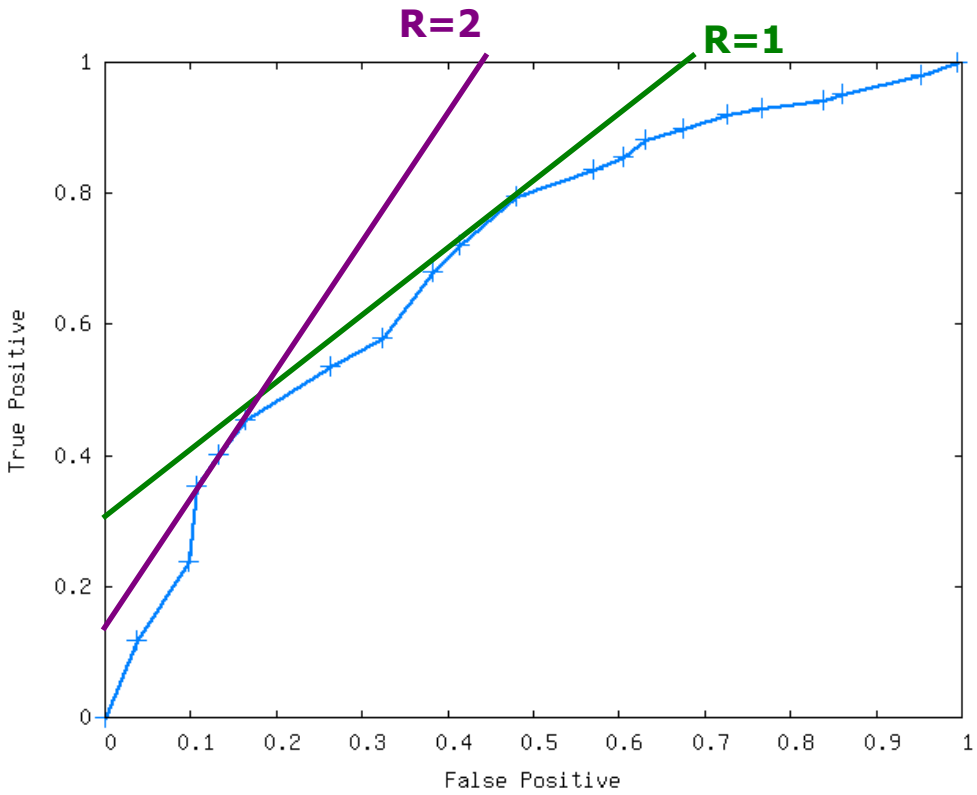


If R=1, we should set the threshold to 10.

If R=10, the threshold should be 29

Receiver Operating Characteristic (ROC) Curve

- It is traditional to plot this same information in a normalized form with True Positive Rate plotted against the False Positive Rate.



The optimal operating point is tangent to a line with a slope of R

Generating ROC Curves

- Linear Threshold Units, Sigmoid Units, Neural Networks
  - adjust the classification threshold between 0 and 1
- K nearest neighbor
  - adjust number of votes (between 0 and k) required to classify positive
- Naïve Bayes, Logistic Regression, etc.
  - vary the probability threshold for classifying as positive
- Support vector machines
  - require different margins for positive and negative examples

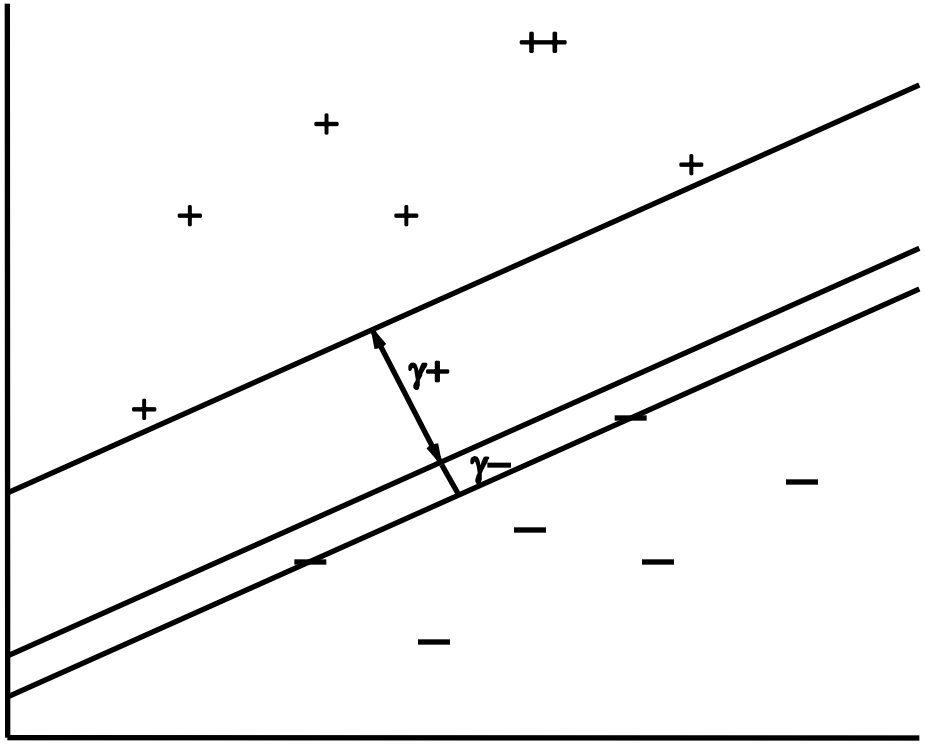
SVM: Asymmetric Margins

Minimize $||w||^2 + C \sum_i \xi_i$

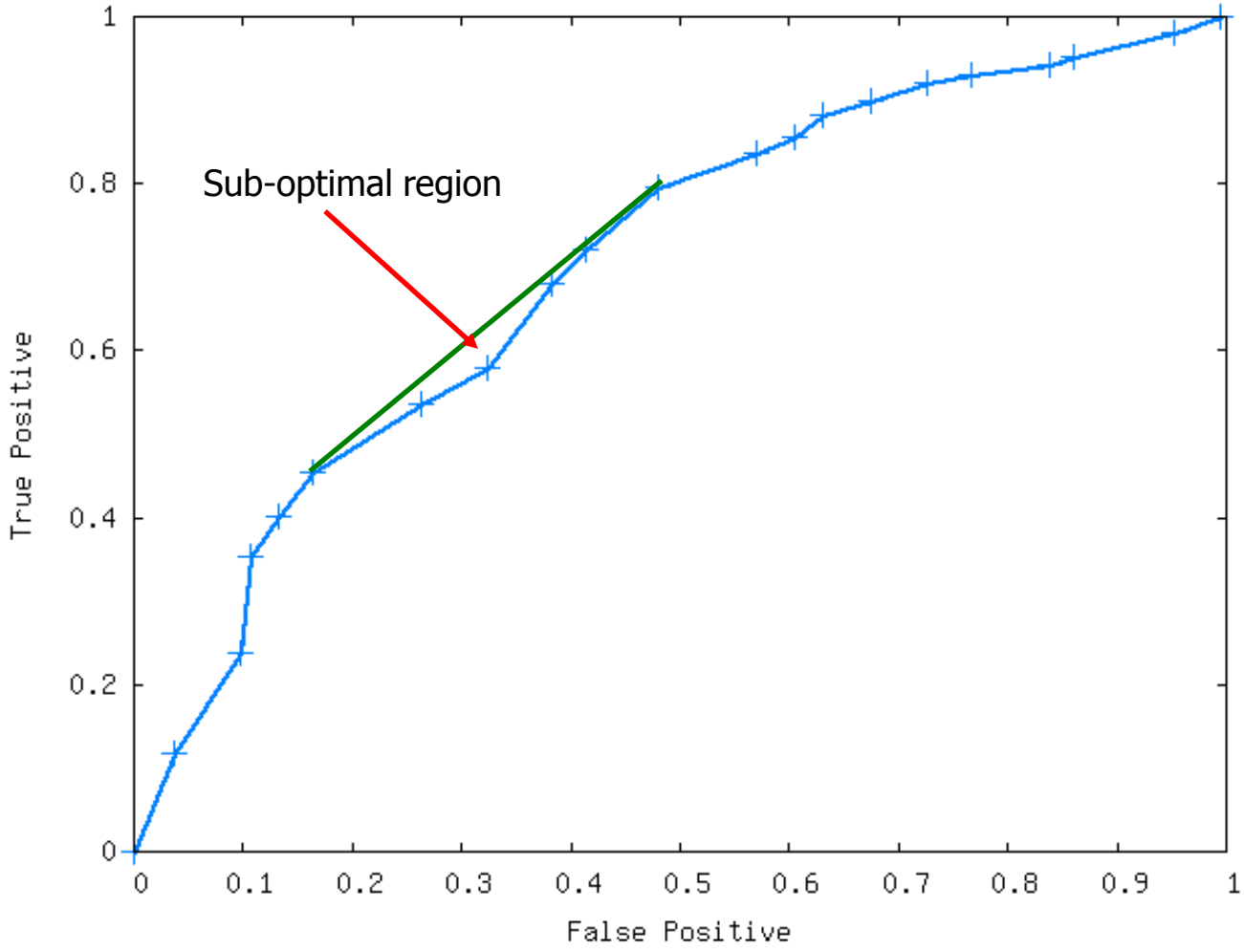
Subject to

$\mathbf{w} \cdot \mathbf{x}_i + \xi_i \geq R$ (positive examples)

$-\mathbf{w} \cdot \mathbf{x}_i + \xi_i \geq 1$ (negative examples)



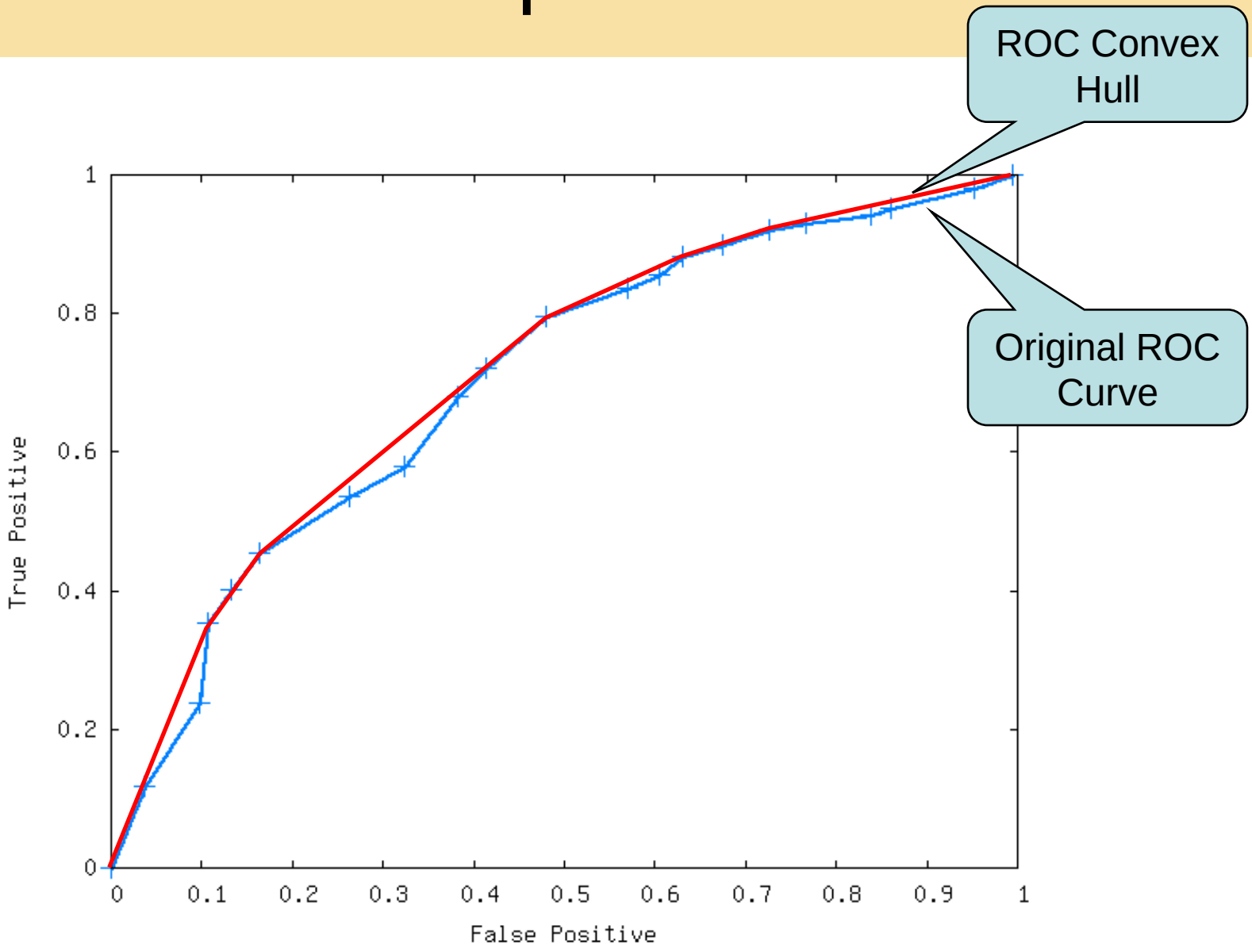
ROC: Sub-Optimal Models



ROC Convex Hull

- If we have two classifiers h_1 and h_2 with (fp1,fn1) and (fp2,fn2), then we can construct a stochastic classifier that interpolates between them. Given a new data point **x**, we use classifier h_1 with probability p and h_2 with probability (1-p). The resulting classifier has an expected false positive level of p fp1 + (1 – p) fp2 and an expected false negative level of p fn1 + (1 – p) fn2.
- This means that we can create a classifier that matches any point on the convex hull of the ROC curve

ROC: Sub-Optimal Models



Maximizing AUC

- At learning time, we may not know the cost ratio R. In such cases, we can maximize the Area Under the ROC Curve (AUC)
- Efficient computation of AUC
  - Assume $h(\mathbf{x})$ returns a real quantity (larger values => class 1)
  - Sort $\mathbf{x}_i$ according to $h(\mathbf{x}_i)$. Number the sorted points from 1 to N such that r(i) = the rank of data point $\mathbf{x}_i$
  - AUC = P(s(x=1)>s(x=0))
    - probability that a randomly chosen example from class 1 ranks above a randomly chosen example from class 0
  - This is also known as the Wilcoxon-Mann-Whitney statistic

Computing AUC

- Let S_1 = sum of r(i) for $y_i = 1$ (sum of the ranks of the positive examples)

$$\widehat{AUC} = \frac{S_1 - N_1(N_1 + 1)/2}{N_0 N_1}$$

  where N_0 is the number of negative examples and N_1 is the number of positive examples
- This can also be computed exactly using standard geometry.

Optimizing AUC

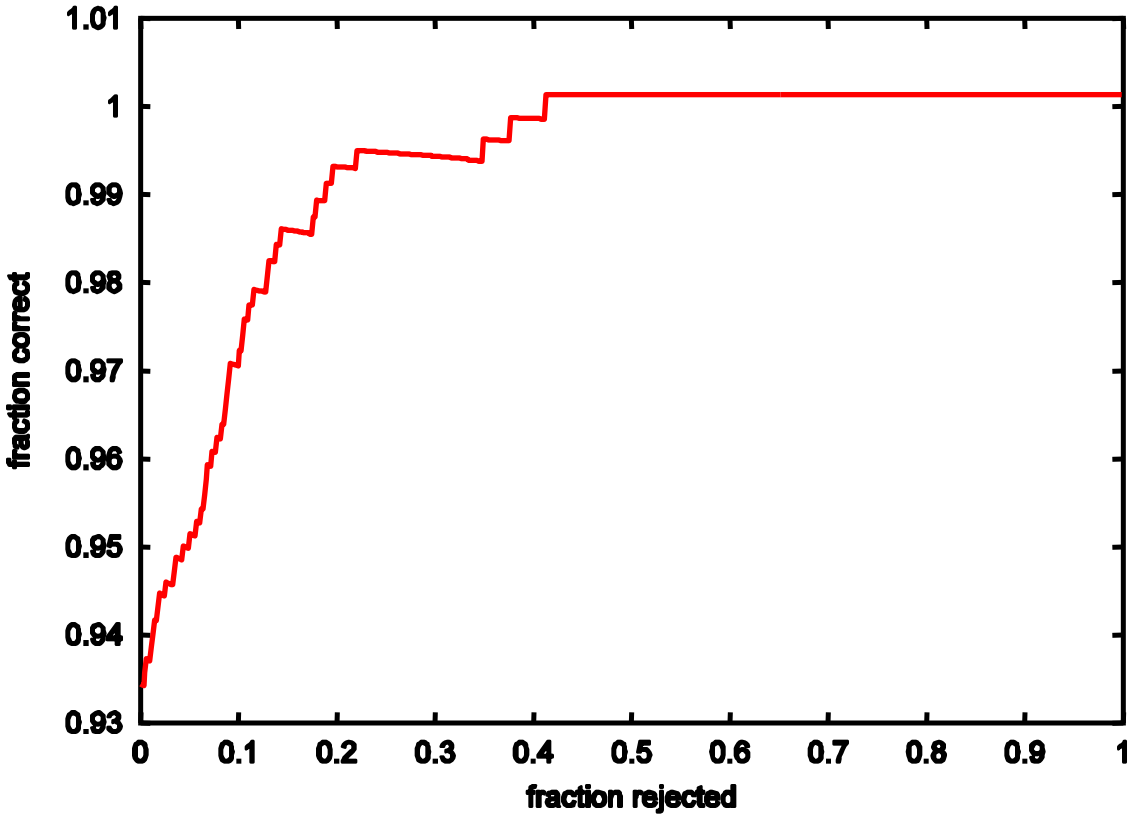
- A hot topic in machine learning right now is developing algorithms for optimizing AUC
- RankBoost: A modification of AdaBoost. The main idea is to define a "ranking loss" function and then penalize a training example **x** by the number of examples of the other class that are misranked (relative to **x**)

Rejection Curves

- In most learning algorithms, we can specify a threshold for making a rejection decision
  - Probabilistic classifiers: adjust cost of rejecting versus cost of FP and FN
  - Decision-boundary method: if a test point **x** is within θ of the decision boundary, then reject
    - Equivalent to requiring that the "activation" of the best class is larger than the second-best class by at least θ

Rejection Curves (2)

- Vary θ and plot fraction correct versus fraction rejected

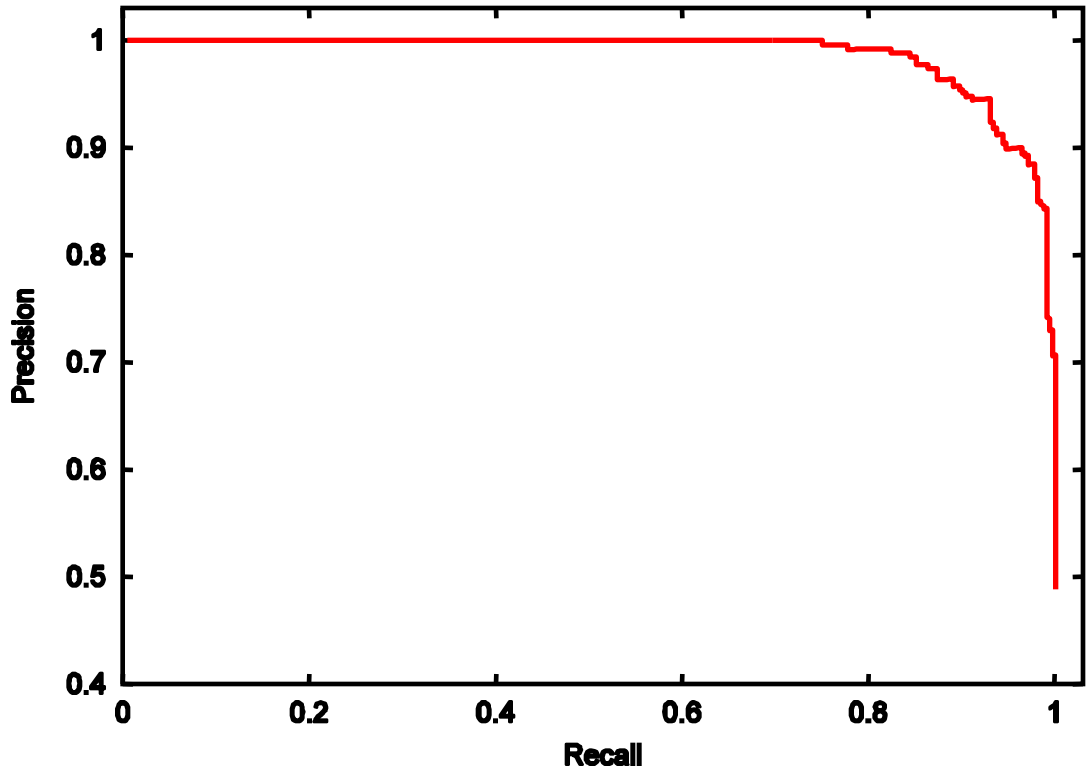


Precision versus Recall

- Information Retrieval:
  - y = 1: document is relevant to query
  - y = 0: document is irrelevant to query
  - K: number of documents retrieved
- Precision:
  - fraction of the K retrieved documents ($\hat{y}=1$) that are actually relevant ($y=1$)
  - TP / (TP + FP)
- Recall:
  - fraction of all relevant documents that are retrieved
  - TP / (TP + FN) = true positive rate

Precision Recall Graph

- Plot recall on horizontal axis; precision on vertical axis; and vary the threshold for making positive predictions (or vary K)



The F_1 Measure

- Figure of merit that combines precision and recall.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

- where P = precision; R = recall. This is twice the harmonic mean of P and R.
- We can plot F_1 as a function of the classification threshold θ

Summarizing a Single Operating Point

- WEKA and many other systems normally report various measures for a single operating point (e.g., $\theta = 0.5$). Here is example output from WEKA:

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.854	0.1	0.899	0.854	0.876	0
0.9	0.146	0.854	0.9	0.876	1

Visualizing ROC and P/R Curves in WEKA

- Right-click on the result list and choose "Visualize Threshold Curve". Select "1" from the popup window.
- ROC:
  - Plot False Positive Rate on X axis
  - Plot True Positive Rate on Y axis
  - WEKA will display the AUC also
- Precision/Recall:
  - Plot Recall on X axis
  - Plot Precision on Y axis
- WEKA does not support rejection curves

Sensitivity and Selectivity

- In medical testing, the terms "sensitivity" and "selectivity" are used
  - Sensitivity = TP/(TP + FN) = true positive rate = recall
  - Selectivity = TN/(FP + TN) = true negative rate = recall for the negative class = 1 – the false positive rate
- The sensitivity versus selectivity tradeoff is identical to the ROC curve tradeoff
  - (ROC plot is sensitivity vs. 1-selectivity)

Estimating the Error Rate of a Classifier

- Compute the error rate on hold-out data
  - suppose a classifier makes *k* errors on *n* holdout data points
  - the estimated error rate is ê = *k* / *n*.
- Compute a confidence internal on this estimate
  - the standard error of this estimate is

$$SE = \sqrt{\frac{\hat{\epsilon} \cdot (1 - \hat{\epsilon})}{n}}$$

  - A (1–α) confidence interval on the true error ε is

$$\hat{\epsilon} - z_{\alpha/2} SE <= \epsilon <= \hat{\epsilon} + z_{\alpha/2} SE$$

  - For a 95% confidence interval, $Z_{0.025}$ = 1.96, so we use

$$\hat{\epsilon} - 1.96 SE <= \epsilon <= \hat{\epsilon} + 1.96 SE.$$

Hypothesis Testing

- Instead of computing error directly, we sometimes want to know whether the error is less than some value, X
  - e.g., the error of another learner
  - i.e., our hypothesis is that $\varepsilon < X$
- If the sample data is consistent with this hypothesis, then we *accept* the hypothesis, otherwise we *reject* it.
- However, we can only assess this given our observed data, so we can only be sure to a certain *degree of confidence*.

Hypothesis Testing (2)

- For example, we may want to know whether the error ε is equal to another value μ.
- We define the *null hypothesis* as:

  H_0: $\varepsilon = \mu$

  against the *alternative* hypothesis:

  H_1: $\varepsilon \neq \mu$
- It is *reasonable* to accept H_0 if ε is not too far from μ.
- Using confidence intervals from before:
  - We *accept* H_0 with a level of significance α if μ lies in the range:

$$\hat{\epsilon} - z_{\alpha/2}SE <= \epsilon <= \hat{\epsilon} + z_{\alpha/2}SE$$

  - This is a *two-sided* test.

Types of Error

- *Type I Error* : If we reject the hypothesis wrongly (i.e., we reject that $\varepsilon = \mu$ when it actually is)
  - α controls for how much Type I Error we will tolerate. Typical values for $\alpha \in \{0.1, 0.05, 0.01\}$
- *Type II Error* : If we accept the hypothesis wrongly (i.e., we accept that $\varepsilon = \mu$ when it actually is not)

	Decision	
Truth	Accept	Reject
True	Correct	Type I Error
False	Type II Error	Correct

Problem of Multiple Comparisons

- Consider testing whether a coin is 'fair' (P(tail)=P(head)=0.5)
  - You flip the coin 10 times and it lands heads at least 9 times
  - Is it fair?
  - If we assume as a null hypothesis that the coin is fair, then the likelihood that a fair coin would come up heads at least 9 out of 10 times is $11/2^{10}=0.0107$.
  - This is relatively unlikely, and under most statistical criteria, one would declare that the null hypothesis should be rejected - i.e. the coin is unfair.

Problem of Multiple Comparisons

- A *multiple comparisons* problem arises if one wanted to use this test (which is appropriate for testing the fairness of a single coin), to test the fairness of many coins.
  - Imagine testing 100 coins by this method.
  - Given that the probability of a fair coin coming up 9 or 10 heads in 10 flips is 0.0107, one would expect that in flipping 100 fair coins ten times each, to see a particular coin come up heads 9 or 10 times would be a relatively likely event.
  - Precisely, the likelihood that all 100 fair coins are identified as fair by this criterion is $(1-0.0107)^{100}=0.34$. Therefore the application of our single-test coin-fairness criterion to multiple comparisons would more likely than not falsely identify a fair coin as unfair.

Problem of Multiple Comparisons

- Technically, the problem of multiple comparisons (also known as *multiple testing problem*) can be described as the potential increase in **Type I Error** that occurs when statistical tests are used repeatedly:
  - If *n* independent comparisons are performed, the experiment-wide significance level α is given by

$$\alpha = 1 - (1-\alpha_{single\_comparison})^n$$

  - (α=0.0107 on the previous slides)
  - Therefore α for 100 fair coin tests from previous slide = 0.66
- This increases as the number of comparisons increases.

Comparing Two Classifiers

- Goal: decide which of two classifiers h_1 and h_2 has lower error rate
- Method: Run them both on the same test data set and record the following information:
  - n_{00}: the number of examples correctly classified by both classifiers
  - n_{01}: the number of examples correctly classified by h_1 but misclassified by h_2
  - n_{10}: The number of examples misclassified by h_1 but correctly classified by h_2
  - n_{11}: The number of examples misclassified by both h_1 and h_2.

n_{00}	n_{01}
n_{10}	n_{11}

McNemar's Test

$$M = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} > \chi^2_{1,\alpha}$$

- M is distributed approximately as χ^2 with 1 degree of freedom. For a 95% confidence test, $\chi^2_{1,095} = 3.84$. So if M is larger than 3.84, then with 95% confidence, we can reject the null hypothesis that the two classifies have the same error rate

Confidence Interval on the Difference Between Two Classifiers

- Let $p_{ij} = n_{ij}/n$ be the 2x2 contingency table converted to probabilities
  - p_{00}: the ratio of examples correctly classified by both classifiers
  - p_{01}: the ratio of examples correctly classified by h_1 but misclassified by h_2
  - p_{10}: The ratio of examples misclassified by h_1 but correctly classified by h_2
  - p_{11}: The ratio of examples misclassified by both h_1 and h_2

$$SE = \sqrt{\frac{p_{01} + p_{10} + (p_{01} - p_{10})^2}{n}}$$
$$p_A = p_{10} + p_{11}$$
$$p_B = p_{01} + p_{11}$$

- A 95% confidence interval on the difference in the true error between the two classifiers is

$$p_A - p_B - 1.96\left(SE + \frac{1}{2n}\right) <= \epsilon_A - \epsilon_B <= p_A - p_B + 1.96\left(SE + \frac{1}{2n}\right)$$

Cost-Sensitive Comparison of Two Classifiers

- Suppose we have a non-0/1 loss matrix $L(\hat{y},y)$ and we have two classifiers h_1 and h_2. Goal: determine which classifier has lower expected loss.
- A method that does not work well:
  - For each algorithm *a* and each test example $(\mathbf{x}_i,y_i)$ compute $\ell_{a,i} = L(h_a(\mathbf{x}_i),y_i)$.
  - Let $\delta_i = \ell_{1,i} - \ell_{2,i}$
  - Treat the δ's as normally distributed and compute a normal confidence interval
- The problem is that there are only a finite number of different possible values for δ_i. They are not normally distributed, and the resulting confidence intervals are too wide

A Better Method: BDeltaCost

- Let $\Delta = \{\delta_i\}^N_{i=1}$ be the set of δ_i's computed as above
- For *b* from 1 to 1000 do
  - Let T_b be a bootstrap replicate of Δ
  - Let s_b = average of the δ's in T_b
- Sort the s_b's and identify the 26^{th} and 975^{th} items. These form a 95% confidence interval on the average difference between the loss from h_1 and the loss from h_2.
- The bootstrap confidence interval quantifies the uncertainty due to the size of the test set. It does not allow us to compare algorithms, only classifiers.

Estimating the Error Rate of a Learning Algorithm

- Under the PAC model, training examples **x** are drawn from an underlying distribution *D* and labeled according to an unknown function *f* to give (**x**,y) pairs where y = *f*(**x**).
- The error rate of a classifier *h* is

$$\text{error}(h) = P_D(h(\mathbf{x}) \neq f(\mathbf{x}))$$

- Define the error rate of a learning algorithm A for sample size *m* and distribution *D* as

$$\text{error}(A,m,D) = E_S\,[\text{error}(A(S))]$$

- This is the expected error rate of h = A(S) for training sets S of size *m* drawn according to *D*.
- We could estimate this if we had several training sets S_1, …, S_L all drawn from *D*. We could compute $A(S_1)$, $A(S_2)$, …, $A(S_L)$, measure their error rates, and average them.
- Unfortunately, we don't have enough data to do this!

Two Practical Methods

- k-fold Cross Validation
  - This provides an unbiased estimate of error(A, (1 – 1/k)*m*, *D*) for training sets of size (1 – 1/k)*m*
- Bootstrap error estimate (out-of-bag estimate)
  - Construct L bootstrap replicates of S_{train}
  - Train A on each of them
  - Evaluate on the examples that *did not appear* in the bootstrap replicate
  - Average the resulting error rates

Estimating the Difference Between Two Algorithms: the 5x2CV F test

for i from 1 to 5 do

perform a 2-fold cross-validation

split S evenly and randomly into S_1 and S_2

for j from 1 to 2 do

Train algorithm A on S_j, measure error rate $p_A^{(i,j)}$

Train algorithm B on S_j, measure error rate $p_B^{(i,j)}$

$p_i^{(j)} := p_A^{(i,j)} - p_B^{(i,j)}$ Difference in error rates on fold j

end /* for j */

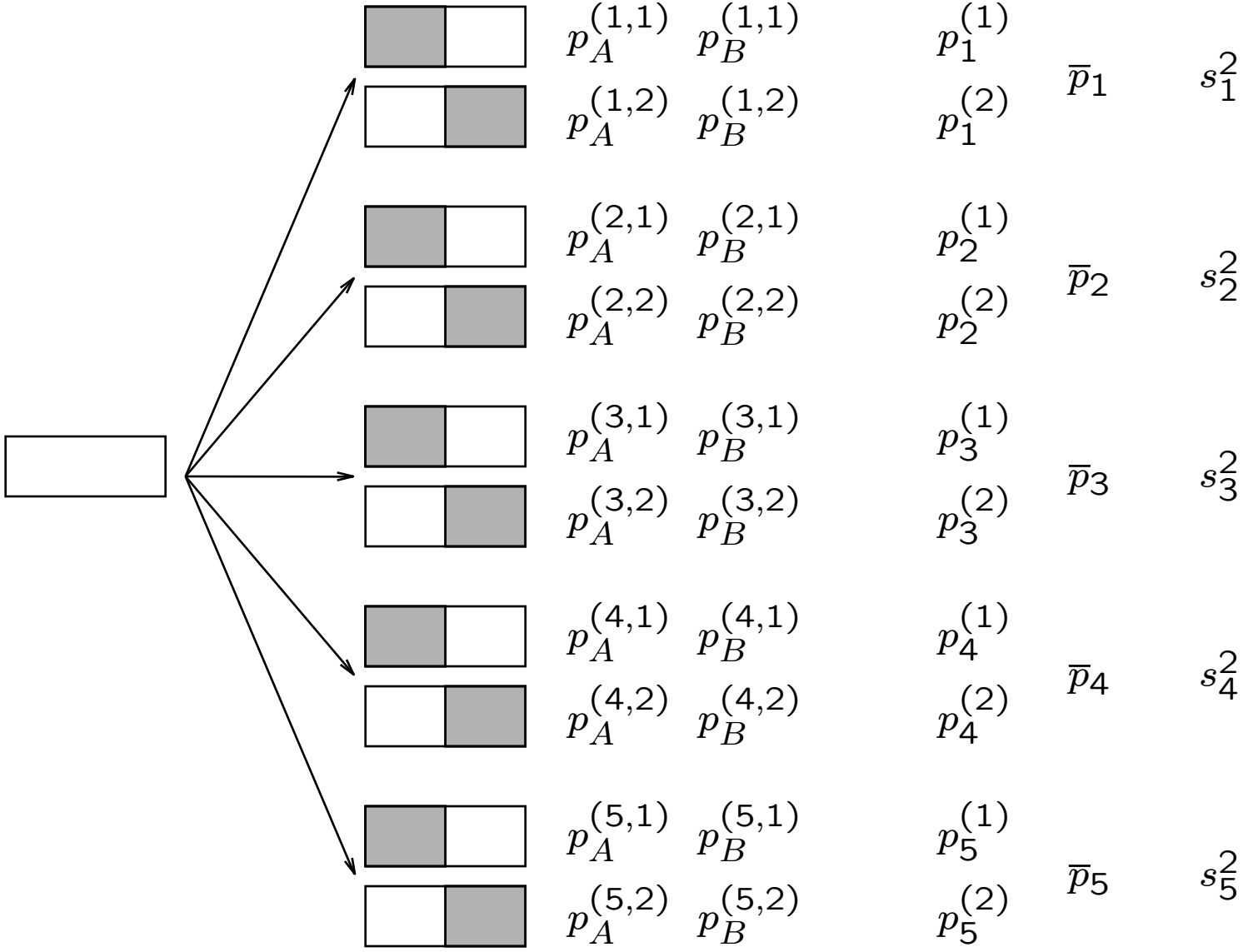
$\overline{p}_i := \frac{p_i^{(1)} + p_i^{(2)}}{2}$ Average difference in error rates in iteration i

$s_i^2 = \left(p_i^{(1)} - \overline{p}_i\right)^2 + \left(p_i^{(2)} - \overline{p}_i\right)^2$ Variance in the difference, for iteration i

end /* for i */

$F := \frac{\sum_i \overline{p}_i^2}{2\sum_i s_i^2}$

5x2CV F test


$p_A^{(1,1)}$ $p_B^{(1,1)}$ $p_1^{(1)}$
$p_A^{(1,2)}$ $p_B^{(1,2)}$ $p_1^{(2)}$
$\overline{p}_1$ s_1^2
$p_A^{(2,1)}$ $p_B^{(2,1)}$ $p_2^{(1)}$
$p_A^{(2,2)}$ $p_B^{(2,2)}$ $p_2^{(2)}$
$\overline{p}_2$ s_2^2
$p_A^{(3,1)}$ $p_B^{(3,1)}$ $p_3^{(1)}$
$p_A^{(3,2)}$ $p_B^{(3,2)}$ $p_3^{(2)}$
$\overline{p}_3$ s_3^2
$p_A^{(4,1)}$ $p_B^{(4,1)}$ $p_4^{(1)}$
$p_A^{(4,2)}$ $p_B^{(4,2)}$ $p_4^{(2)}$
$\overline{p}_4$ s_4^2
$p_A^{(5,1)}$ $p_B^{(5,1)}$ $p_5^{(1)}$
$p_A^{(5,2)}$ $p_B^{(5,2)}$ $p_5^{(2)}$
$\overline{p}_5$ s_5^2

5x2CV F test

- If F > 4.47, then with 95% confidence, we can reject the null hypothesis that algorithms A and B have the same error rate when trained on data sets of size $m/2$.

- An F-test is any statistical test in which the test statistic has an F-distribution if the null hypothesis is true, e.g.:
  - The hypothesis that the means of multiple normally distributed populations, all having the same standard deviation, are equal.
  - The hypothesis that the standard deviations of two normally distributed populations are equal, and thus that they are of comparable origin.

Summary

- ROC Curves
- Reject Curves
- Precision-Recall Curves
- Statistical Tests
  - Estimating error rate of classifier
  - Comparing two classifiers
  - Estimating error rate of a learning algorithm
  - Comparing two algorithms