Evaluation and Cost-Sensitive Learning

- Evaluation
  - Hold-out Estimates
  - Cross-validation
- Significance Testing
  - Sign test
- ROC Analysis
  - Cost-Sensitive Evaluation
  - ROC space
  - ROC convex hull
  - Rankers and Classifiers
  - ROC curves
  - AUC
- Cost-Sensitive Learning

Evaluation of Learned Models

- Validation through experts
  - a domain experts evaluates the plausibility of a learned model
    - \+ but often the only option (e.g., clustering)
    - – subjective, time-intensive, costly
- Validation on data
  - evaluate the accuracy of the model on a separate dataset drawn from the same distribution as the training data
    - – labeled data are scarce, could be better used for training
    - \+ fast and simple, off-line, no domain knowledge needed, methods for re-using training data exist (e.g., cross-validation)
- On-line Validation
  - test the learned model in a fielded application
    - \+ gives the best estimate for the overall utility
    - – bad models may be costly

Confusion Matrix (Concept Learning)

	Classified as +	Classified as –	
Is +	true positives (tp)	false negatives (fn)	tp + fn = P
Is –	false positives (fp)	true negatives (tn)	fp + tn = N
	tp + fp	fn + tn	|E| =P + N

- the confusion matrix summarizes all important information
  - how often is class i confused with class j
- most evaluation measures can be computed from the confusion matrix
  - accuracy
  - recall/precision, sensitivity/specificity
  - ...

Basic Evaluation Measures

- true positive rate: $tpr = \frac{tp}{tp + fn}$
  - percentage of *correctly* classified *positive* examples
- false positive rate: $fpr = \frac{fp}{fp + tn}$
  - percentage of negative examples *incorrectly* classified *as positive*
- false negative rate: $fnr = \frac{fn}{tp + fn} = 1 - tpr$
  - percentage of positive examples *incorrectly* classified *as negative*
- true negative rate: $tnr = \frac{tn}{fp + tn} = 1 - fpr$
  - percentage of *correctly* classified *negative* examples
- accuracy: $acc = \frac{tp + tn}{P + N}$
  - percentage of correctly classified examples
  - can be written in terms of *tpr* and *fpr*: $acc = \frac{P}{P+N} \cdot tpr + \frac{N}{P+N} \cdot (1 - fpr)$
- error: $err = \frac{fp + fn}{P + N} = 1 - acc = \frac{P}{P+N} \cdot (1 - tpr) + \frac{N}{P+N} \cdot fpr$
  - percentage of incorrectly classified examples

Confusion Matrix (Multi-Class Problems)

- for multi-class problems, the confusion matrix has many more entries:

classified as

true class	A	B	C	D	
A	$n_{A,A}$	$n_{B,A}$	$n_{C,A}$	$n_{D,A}$	n_A
B	$n_{A,B}$	$n_{B,B}$	$n_{C,B}$	$n_{D,B}$	n_B
C	$n_{A,C}$	$n_{B,C}$	$n_{C,C}$	$n_{D,C}$	n_C
D	$n_{A,D}$	$n_{B,D}$	$n_{C,D}$	$n_{D,D}$	n_D
	$\bar{n}_A$	$\bar{n}_B$	$\bar{n}_C$	$\bar{n}_D$	$\lvert E \rvert$

- accuracy is defined analogously to the two-class case:

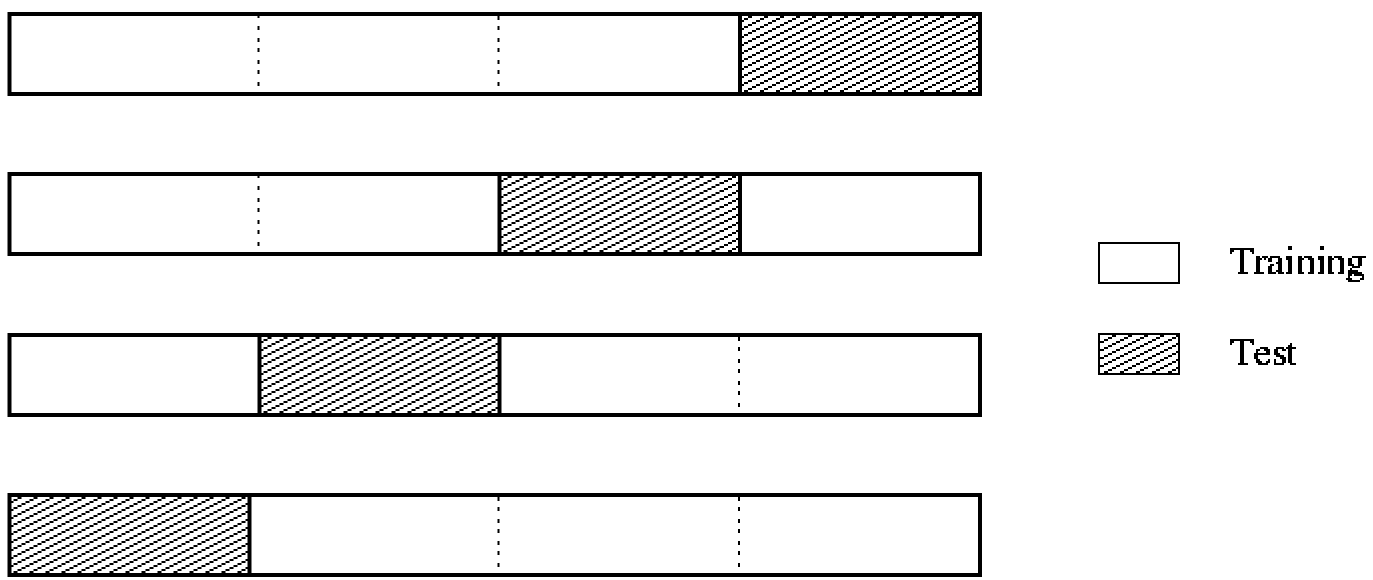
$$accuracy = \frac{n_{A,A} + n_{B,B} + n_{C,C} + n_{D,D}}{|E|}$$

Out-of-Sample Testing

- Performance cannot be measured on training data
  - overfitting!
- Reserve a portion of the available data for testing
  - typical scenario
    - 2/3 of data for training
    - 1/3 of data for testing (evaluation)
  - a classifier is trained on the training data
  - and tested on the test data
    - e.g., confusion matrix is computed for test data set
- Problems:
  - waste of data
  - labelling may be expensive
  - high variance
    - often: repeat 10 times or → cross-validation

Cross-Validation

- Algorithm:
  - split dataset into x (usually 10) partitions
  - for every partition X
    - use other x-1 partitions for learning and partition X for testing
  - average the results
- Example: 4-fold cross-validation



Leave-One-Out Cross-Validation

- n-fold cross-validation
  - where n is the number of examples:
    - use n-1 examples for training
    - 1 example for testing
    - repeat for each example
- Properties:
  + makes best use of data
    - only one example not used for testing
  + no influence of random sampling
    - training/test splits are determined deterministically
  – typically very expensive
    - but, e.g., not for k-NN (Why?)
  – bias
    - example see exercises

Experimental Evaluation of Algorithms

- Typical experimental setup (in % Accuracy):
  - evaluate *n* algorithms on *m* datasets

Dataset	Grading	Select	Stacking	Voting
audiology	83.36	77.61	76.02	84.56
autos	80.93	80.83	82.20	83.51
balance-scale	89.89	91.54	89.50	86.16
breast-cancer	73.99	71.64	72.06	74.86
breast-w	96.70	97.47	97.41	96.82
colic	84.38	84.48	84.78	85.08
credit-a	86.01	84.87	86.09	86.04
credit-g	75.64	75.48	76.17	75.23
diabetes	75.53	76.86	76.32	76.25
glass	74.35	74.44	76.45	75.70
heart-c	82.74	84.09	84.26	81.55
heart-h	83.64	85.78	85.14	83.16
heart-statlog	84.22	83.56	84.04	83.30

Dataset	Grading	Select	Stacking	Voting
hepatitis	83.42	83.03	83.29	82.77
ionosphere	91.85	91.34	92.82	92.42
iris	95.13	95.20	94.93	94.93
labor	93.68	90.35	91.58	93.86
lymph	83.45	81.69	80.20	84.05
primary-t.	49.47	49.23	42.63	46.02
segment	98.03	97.05	98.08	98.14
sonar	85.05	85.05	85.58	84.23
soybean	93.91	93.69	92.90	93.84
vehicle	74.46	73.90	79.89	72.91
vote	95.93	95.95	96.32	95.33
vowel	98.74	99.06	99.00	98.80
zoo	96.44	95.05	93.96	97.23

- Can we conclude that algorithm X is better than Y? How?

Summarizing Experimental Results

- Averaging the performance
  - May be deceptive:
    - algorithm A is 0.1% better on 19 datasets with thousands of examples
    - algorithm B is 2% better on 1 dataset with 50 examples
    - A is better, but B has the higher average accuracy
  - In our example: “Grading” is best on average

Dataset	Grading	Select	Stacking	Voting
Avg	85.04	84.59	84.68	84.88

- Counting wins/ties/losses
  - now “Stacking” is best
  - Results are “inconsistent”:
    - Grading > Select > Voting > Grading
  - How many “wins” are needed to conclude that one method is better than the other?

	Grading	Select	Stacking	Voting
Grading	—	15/1/10	11/0/15	12/0/14
Select	10/1/15	—	10/0/16	14/0/12
Stacking	15/0/11	16/0/10	—	15/1/10
Voting	14/0/12	12/0/14	10/1/15	—

Sign Test

- Given:
  - A coin with two sides (heads and tails)
- Question:
  - How often do we need heads in order to be sure that the coin is not fair?
- Null Hypothesis:
  - The coin is fair ($P(\text{heads}) = P(\text{tails}) = 0.5$)
  - We want to refute that!
- Experiment:
  - Throw up the coin N times
- Result:
  - i heads, $N-i$ tails
  - What is the probability of observing i under the null hypothesis?

Sign Test

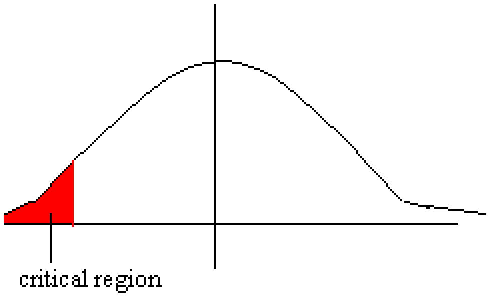
- Given:
  - ~~A coin with two sid~~ Two Learning Algorithms (A and B)
- Question:
  - ~~How ofte~~ ~~the coin i~~ On how many datasets must A be better than B to ensure that A is a better algorithm than B?
- Null Hypothesis:
  - ~~The coin is fair ($P(heads) = P(ta$~~ Both Algorithms are equal.
  - We want to refute that!
- Experiment:
  - ~~Throw up the coin N ti~~ Run both algorithms on N datasets
- Result:
  - ~~i heads, $N - i$ tails~~ i wins for A on N-i wins for B
  - What is the probability of observing i under the null hypothesis?

Sign Test: Summary

We have a binomial distribution with $p = ½$

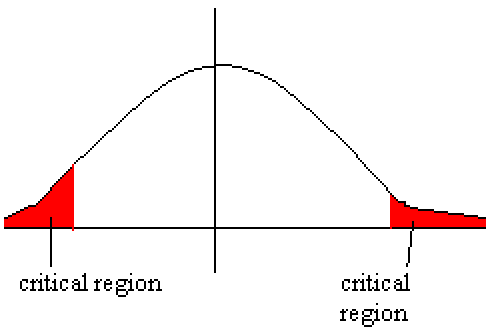
- the probability of having i successes is $P(i) = \binom{N}{i} p^i (1-p)^{N-i}$
- the probability of having at most k successes is (one-tailed test)

$$P(i \leq k) = \sum_{i=1}^{k} \binom{N}{i} \frac{1}{2^i} \cdot \frac{1}{2^{N-i}} = \frac{1}{2^N} \sum_{i=1}^{k} \binom{N}{i}$$



- the probability of having at most k successes or at least $N-k$ successes is (two-tailed test)

$$P(i \leq k \vee i \geq N-k) = \frac{1}{2^N} \sum_{i=1}^{k} \binom{N}{i} + \frac{1}{2^N} \sum_{i=1}^{k} \binom{N}{N-i} = \frac{1}{2^{N-1}} \sum_{i=1}^{k} \binom{N}{i}$$



- for large N, this can be approximated with a normal distribution

Table Sign Test

Source: Erich Mittenecker, *Planung und Statistische Auswertung von Experimenten*, 10th edition, Deuticke Verlag, Wien, 1983.

- Example:
  - 20 datasets
  - Alg. A vs. B
    - A 4 wins
    - B 14 wins
    - 2 ties (not counted)
  - we can say with a certainty of 95% that B is better than A
  - but not with 99% certainty!
- Online:
  http://www.fon.hum.uva.nl/Service/Statistics/Sign_Test.html

Vorzeichentest: Kritische Häufigkeiten i bzw. $N - i$ (s. S. 167)

N	Irrtumswahrscheinlichkeit 1%	Irrtumswahrscheinlichkeit 5%	N	Irrtumswahrscheinlichkeit 1%	Irrtumswahrscheinlichkeit 5%
6	—	0	41	11	13
7	—	0	42	12	14
8	0	0	43	12	14
9	0	1	44	13	15
10	0	1	45	13	15
11	0	1	46	13	15
12	1	2	47	14	16
13	1	2	48	14	16
14	1	2	49	15	17
15	2	3	50	15	17
16	2	3	51	15	18
17	2	4	52	16	18
18	3	4	53	16	18
19	3	4	54	17	19
20	3	5	55	17	19
21	4	5	56	17	20
22	4	5	57	18	20
23	4	6	58	18	21
24	5	6	59	19	21
25	5	7	60	19	21
26	6	7	61	20	22
27	6	7	62	20	22
28	6	8	63	20	23
29	7	8	64	21	23
30	7	9	65	21	24
31	7	9	66	22	24
32	8	9	67	22	25
33	8	10	68	22	25
34	9	10	69	23	25
35	9	11	70	23	26
36	9	11	71	24	26
37	10	12	72	24	27
38	10	12	73	25	27
39	11	12	74	25	28
			[illegible]	5	28

Properties

- Sign test is a very simple test
  - does not make any assumption about the distribution
- Sign test is very conservative
  - If it detects a significant difference, you can be sure it is
  - If it does not detect a significant difference, a different test that models the distribution of the data may still yield significance
- Alternative tests:
  - two-tailed t-test:
    - incorporates magnitude of the differences in each experiment
    - assumes that differences form a normal distribution
- Rule of thumb:
  - Sign test answers the question “How often?”
  - t-test answers the question “How much?”

Problem of Multiple Comparisons

- Problem:
  - for each pair of algorithms we have a probability of 5% that one algorithm appears to be better than the other
  - even if the null hypothesis holds
  - then if we make many pairwise comparisons
  - the chance that an apparently “significant” difference is observed increases rapidly
- Solutions:
  - Bonferroni adjustments:
    - **Basic idea:** tighten the significance thresholds depending on the number of comparisons
    - Too conservative
  - No recommended procedure yet

Cost-Sensitive Evaluation

- Predicting class j instead of the correct i is associated with a cost factor $C(i \mid j)$
  - 0/1-loss (accuracy): $C(i|j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$
  - general case for concept learning:

	Classified as +	**Classified as –**
Is +	C(+|+)	C(–|+)
Is –	C(+|–)	C(–|–)

Examples

- Loan Applications
  - rejecting an applicant who will not pay back → minimal costs
  - accepting an applicant who will pay back → gain
  - accepting an applicant who will not pay back → big loss
  - rejecting an applicant who would pay back → loss
- Spam-Mail Filtering
  - rejecting good E-mails (ham) is much worse than accepting a few spam mails
- Medical Diagnosis
  - failing to recognize a disease is often much worse than to treat a healthy patient for this disease

Cost-Sensitive Evaluation

- Total Cost (Loss):

$$L = tpr \cdot C(+|+) + fpr \cdot C(+|-) + fnr \cdot C(-|+) + tnr \cdot C(-|-)$$

- If there are no costs for correct classification:

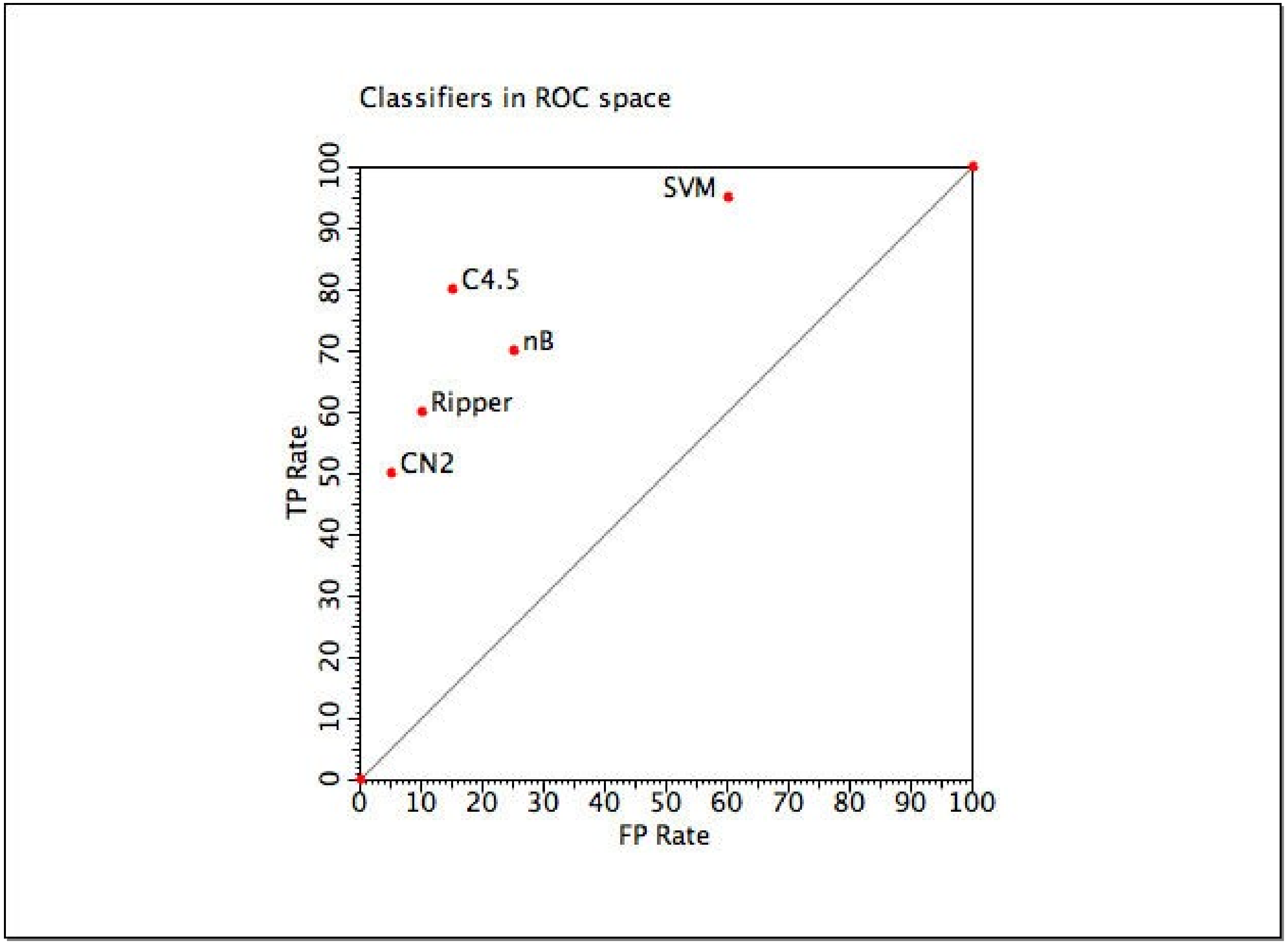
$$L = fpr \cdot C(+|-) + fnr \cdot C(-|+) = \boxed{fpr \cdot C(+|-) + (1-tpr) \cdot C(-|+)}$$

  - note the general form:
    - this is (except for a constant term) the linear cost metric we know from rule learning
- Distribution of positive and negative examples may be viewed as a cost parameter
  - error is a special case $\left(C(+|-)=\frac{N}{P+N},\ C(-|+)=\frac{P}{P+N}\right)$
  - we abbreviate the costs with $c_- = C(+|-)$, $c_+ = C(-|+)$

ROC Analysis

- Receiver Operating Characteristic
  - origins in signal theory to show tradeoff between hit rate and false alarm rate over noisy channel
- Basic Objective:
  - Determine the best classifier for varying cost models
    - accuracy is only one possibility, where true positives and false positives receive equal weight
- Method:
  - Visualization in ROC space
  - ROC space is like coverage space (→ rule learning) except that axes are normalized
    - x-axis: false positive rate fpr
    - y-axis: true positive rate tpr

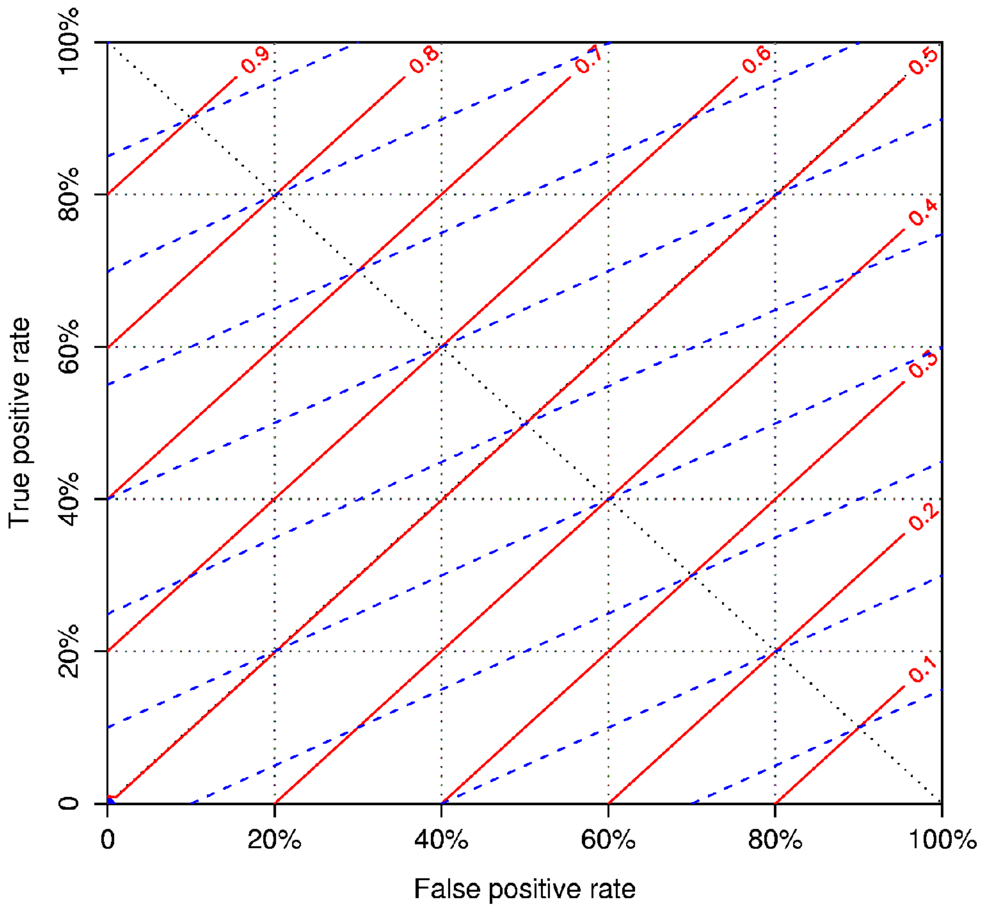
Example ROC plot



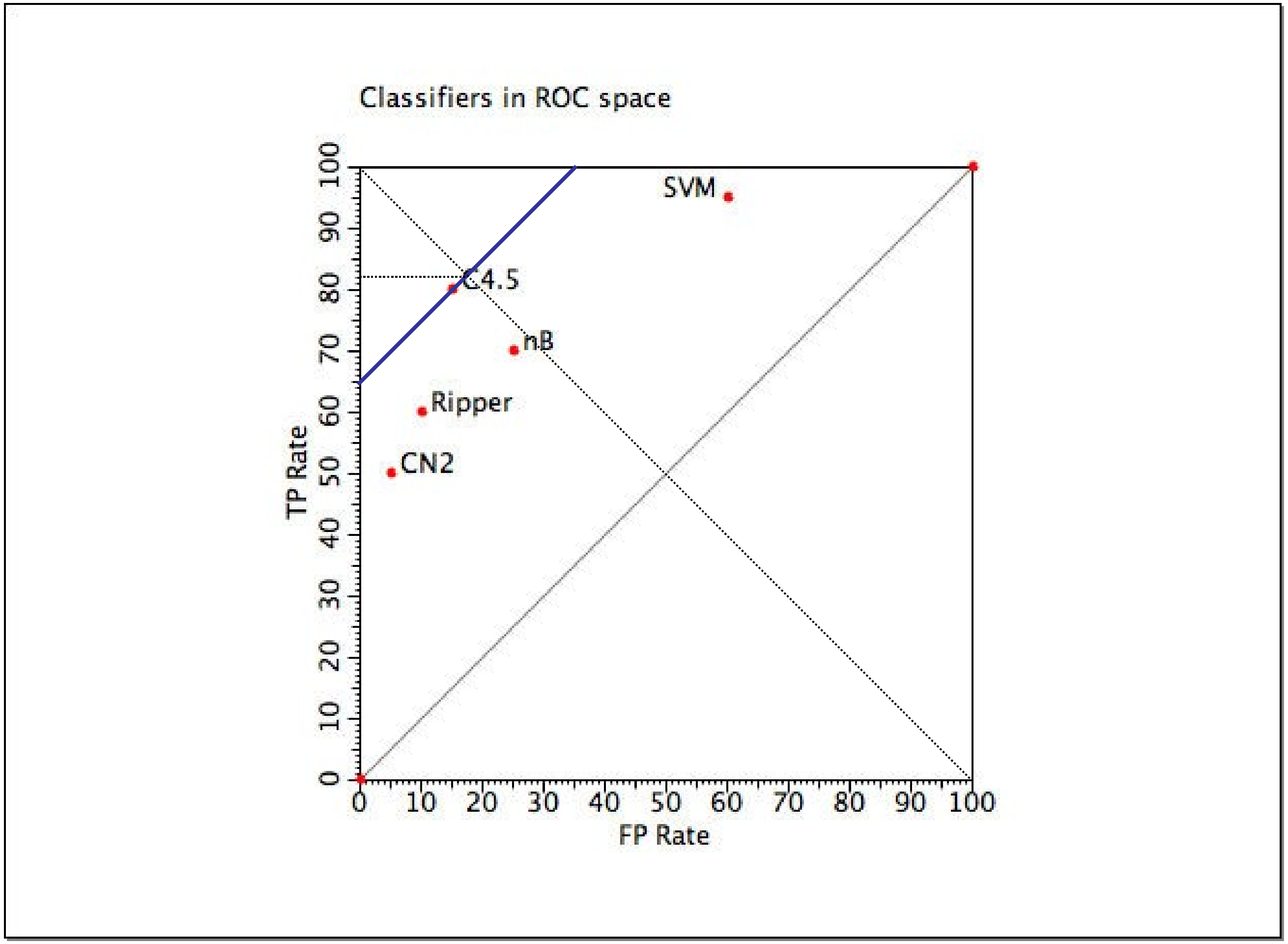
ROC plot produced by ROCon (http://www.cs.bris.ac.uk/Research/MachineLearning/rocon/)

ROC isometrics

- Iso-cost lines connects ROC points with the same costs c
  - $c = c_{+} \cdot (1 - tpr) + c_{-} \cdot fpr$
  - $tpr = \frac{c_{-}}{c_{+}} \cdot fpr + \left(\frac{c}{c_{+}} - 1 \right)$
- Cost isometrics are parallel ascending lines with slope c_{-}/c_{+}
  - e.g., error/accuracy slope = P/N

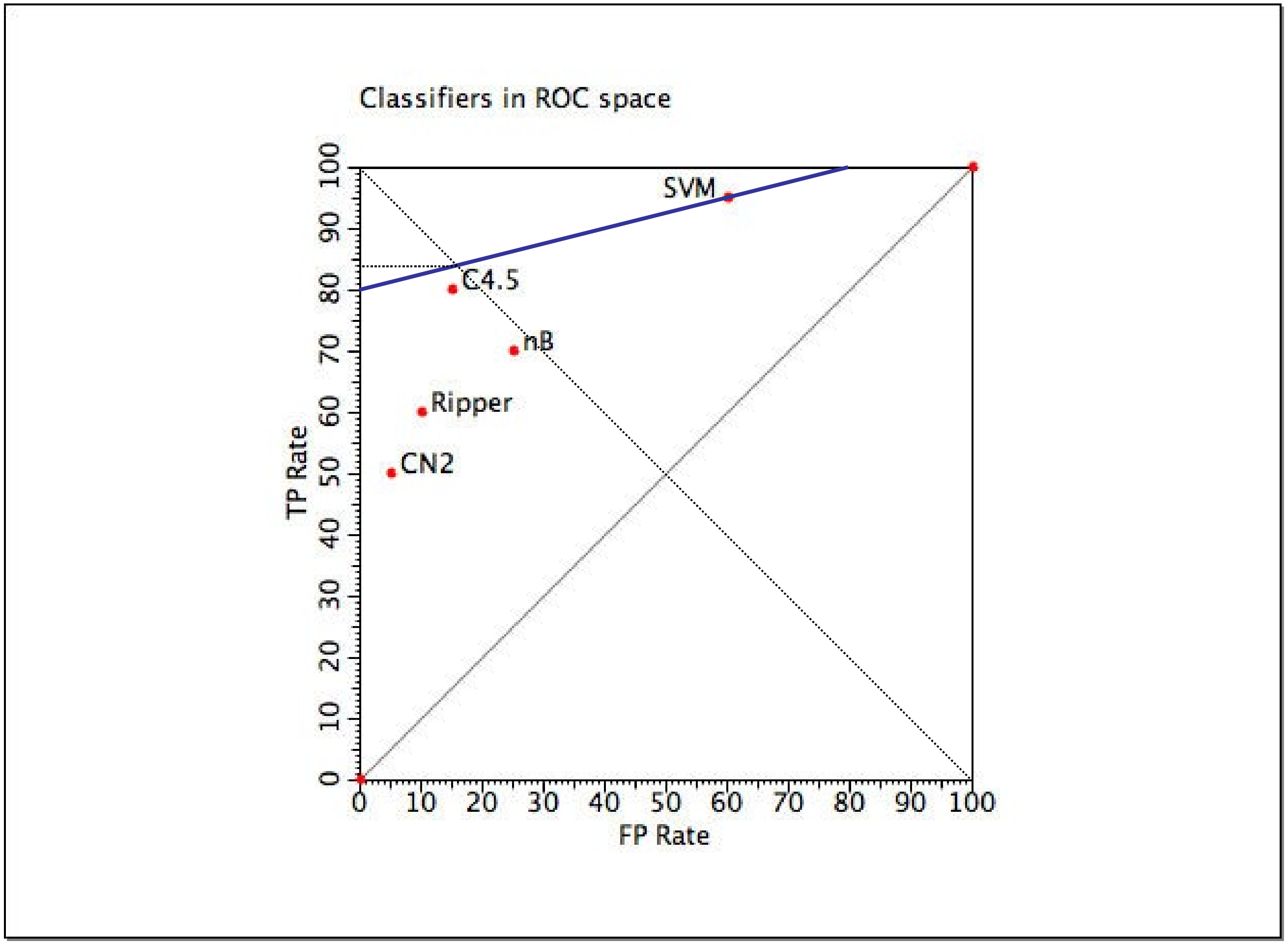


Selecting the optimal classifier



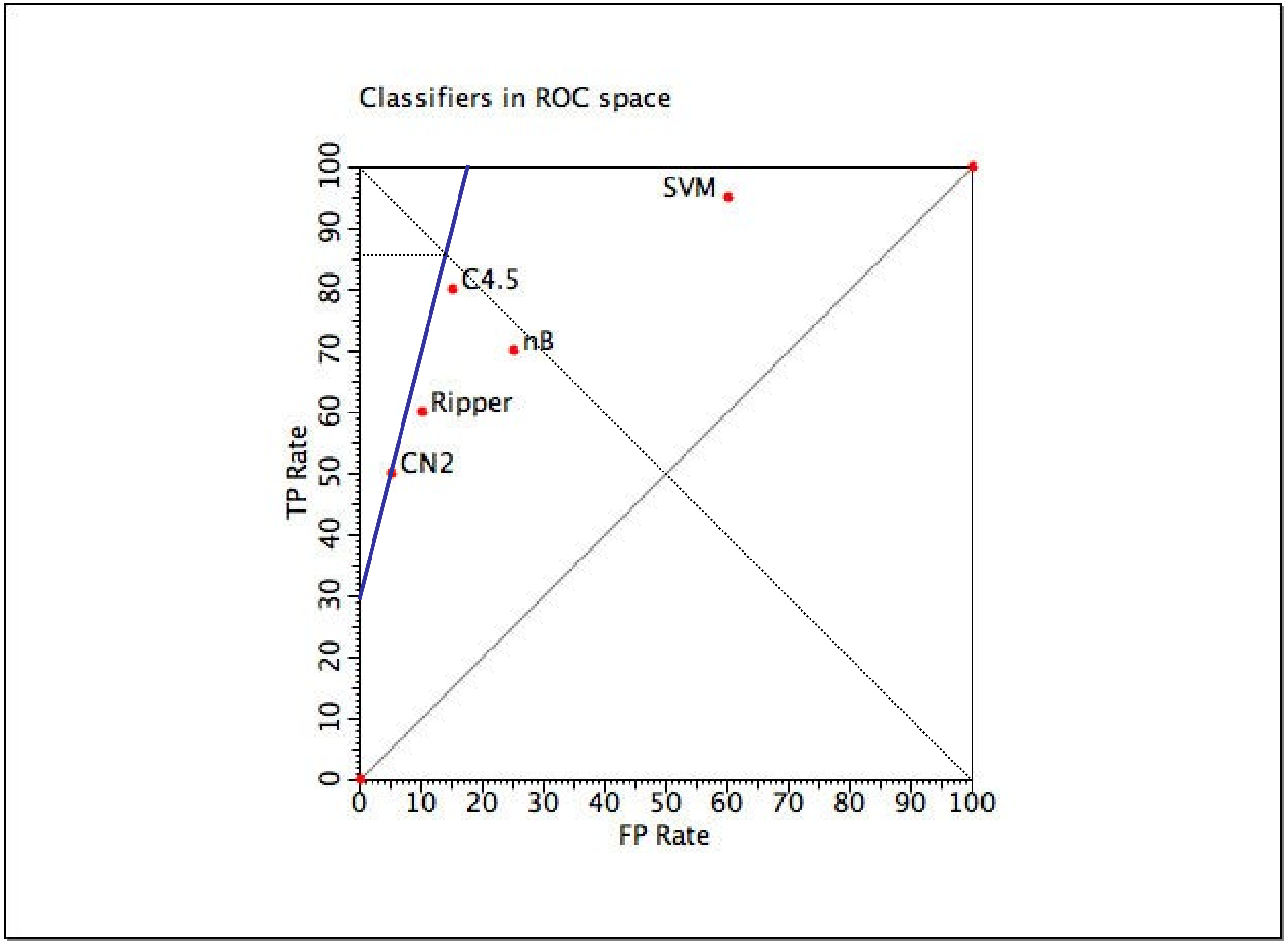
For uniform class distribution, C4.5 is optimal

Selecting the optimal classifier



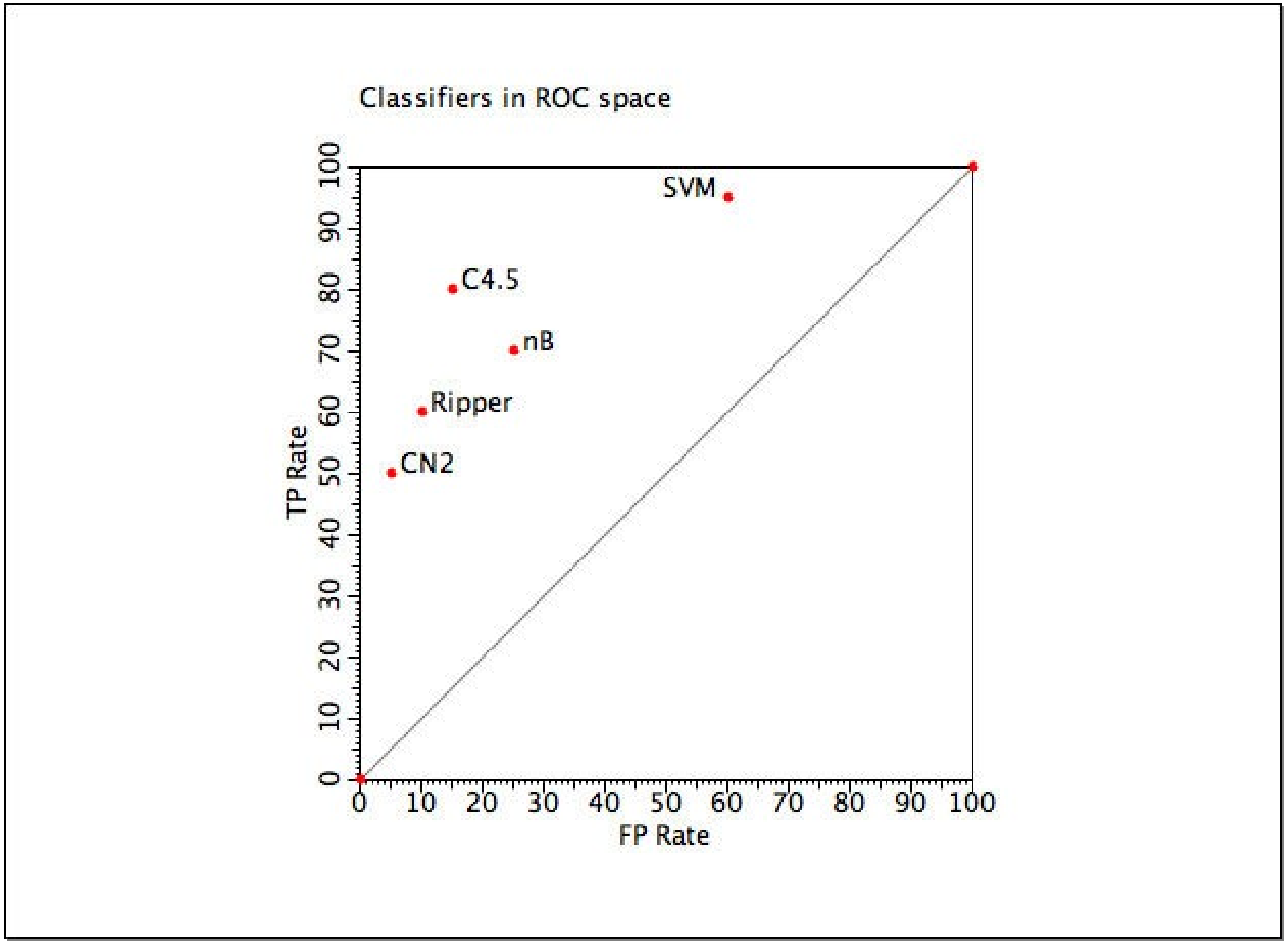
With four times as many positives as negatives, SVM is optimal

Selecting the optimal classifier



With four times as many negatives as positives, CN2 is optimal

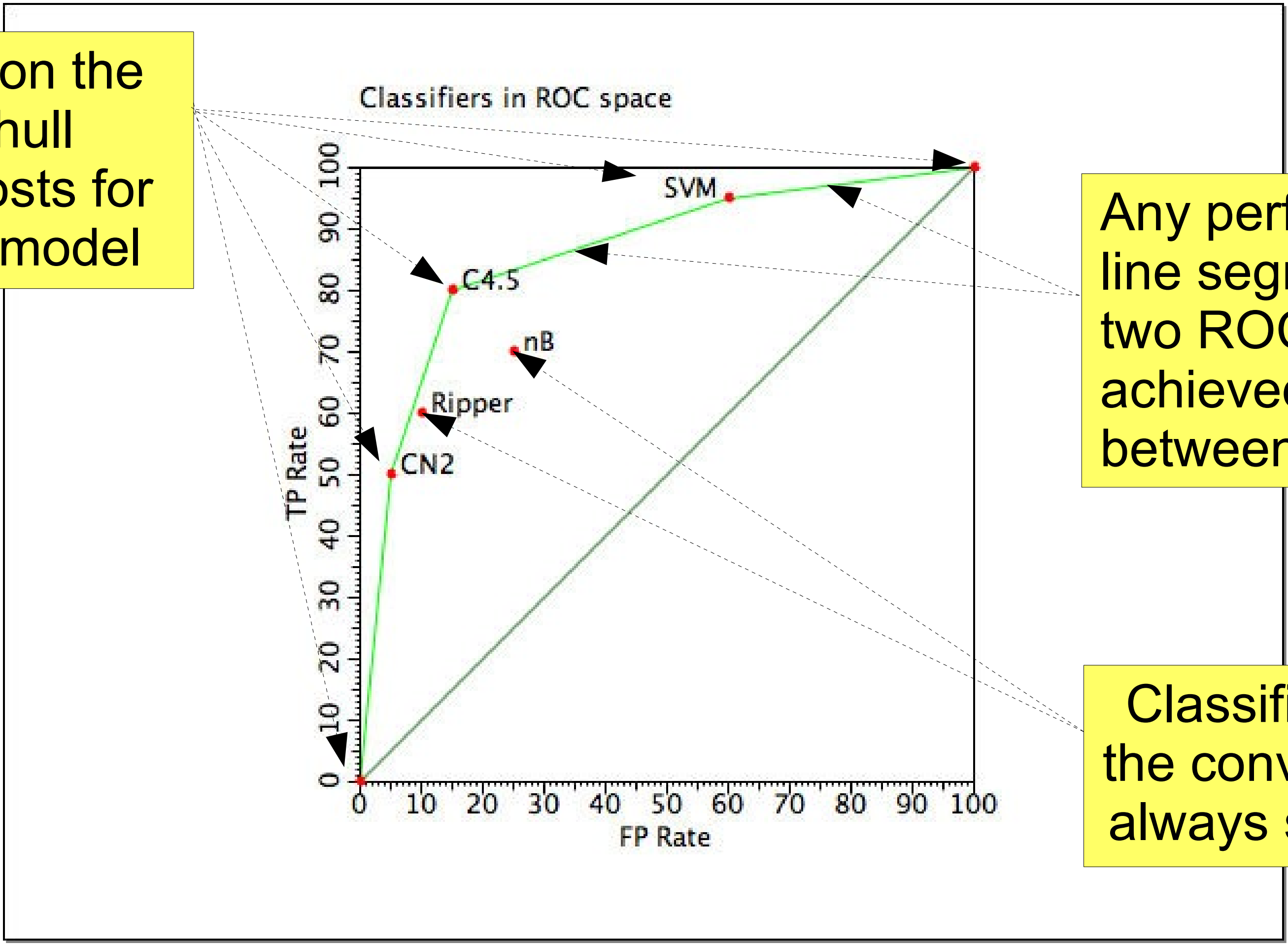
Selecting the optimal classifier



- With less than 9% positives, AlwaysNeg is optimal
- With less than 11% negatives, AlwaysPos is optimal

The ROC convex hull

Classifiers on the convex hull minimize costs for some cost model



Any performance on a line segment connecting two ROC points can be achieved by interpolating between the classifiers

Classifiers below the convex hull are always suboptimal

Interpolating Classifiers

- Given two learning schemes we can achieve any point on the convex hull!
  - TP and FP rates for scheme 1: tpr_1 and fpr_1
  - TP and FP rates for scheme 2: tpr_2 and fpr_2

- If scheme 1 is used to predict $100 \times q\%$ of the cases and scheme 2 for the rest, then
  - TP rate for combined scheme: $tpr_q = q \cdot tpr_1 + (1-q) \cdot tpr_2$
  - FP rate for combined scheme: $fpr_q = q \cdot fpr_1 + (1-q) \cdot fpr_2$

Rankers and Classifiers

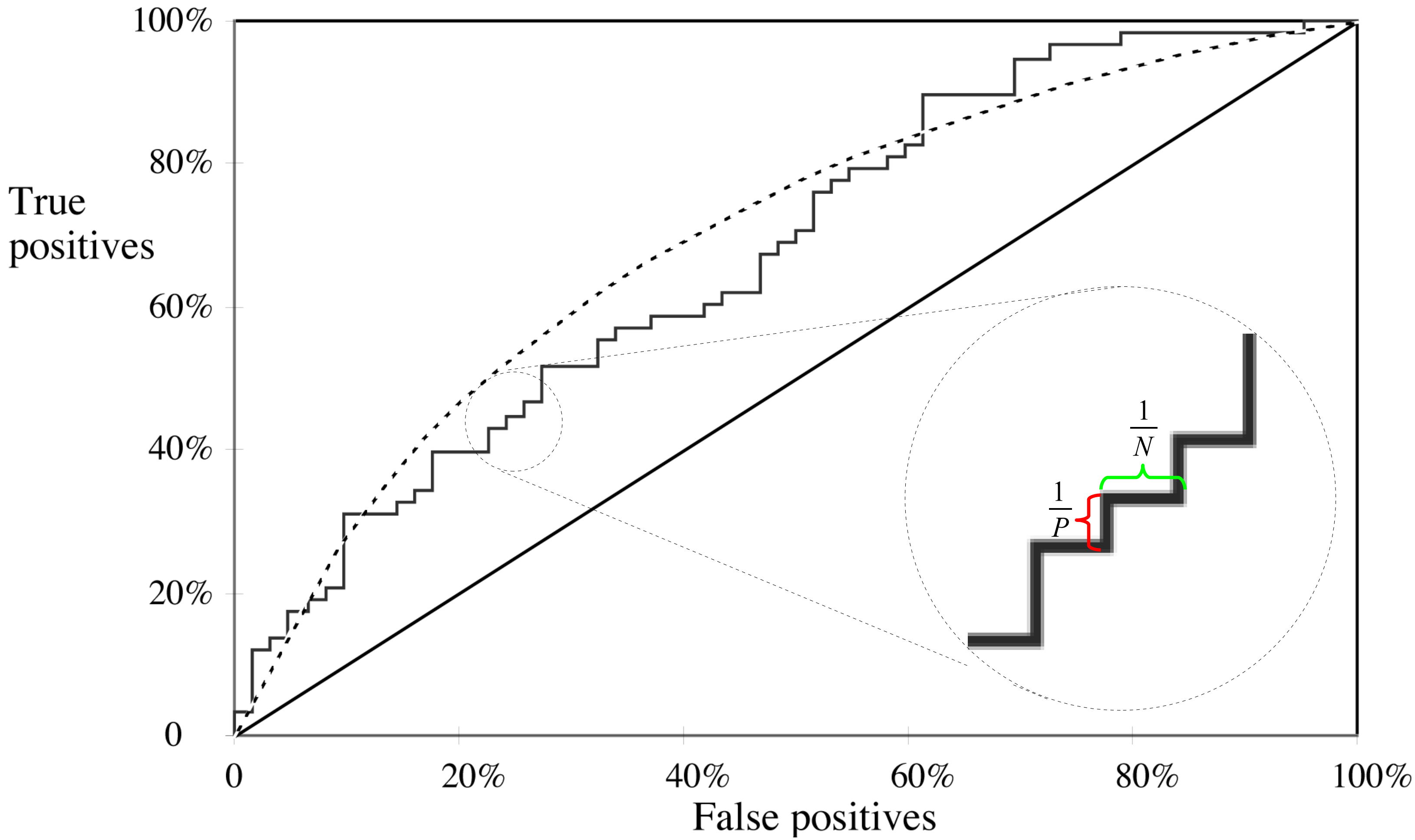
- A scoring classifier outputs scores $f(x,+)$ and $f(x,-)$ for each class
  - e.g. estimate probabilities $P(+|x)$ and $P(-|x)$
  - scores don't need to be normalised
- $f(x) = f(x,+) / f(x,-)$ can be used to rank instances from most to least likely positive
  - e.g. odds ratio $P(+|x) / P(-|x)$
- Rankers can be turned into classifiers by setting a threshold on $f(x)$
- Example:
  - Naïve Bayes Classifier for two classes is actually a ranker
  - that has been turned into classifier by setting a probability threshold of 0.5 (corresponds to a odds ratio treshold of 1.0)
    - $P(+|x) > 0.5 > 1 - P(+|x) = P(-|x)$ means that class $+$ is more likely

Drawing ROC Curves for Rankers

Performance of a ranker can be visualized via a ROC curve

- Naïve method:
  - consider all possible thresholds
    - in fact, only $k+1$ for k instances
  - each threshold corresponds to one a new classifier
  - for each classifier
    - construct confusion matrix
    - plot classifier at point (fpr,tpr) in ROC space
- Practical method:
  - rank test instances on decreasing score $f(x)$
  - start in $(0,0)$
    - if the next instance in the ranking is + move $1/P$ up
    - if the next instance in the ranking is – move $1/N$ to the right
    - make diagonal move in case of ties

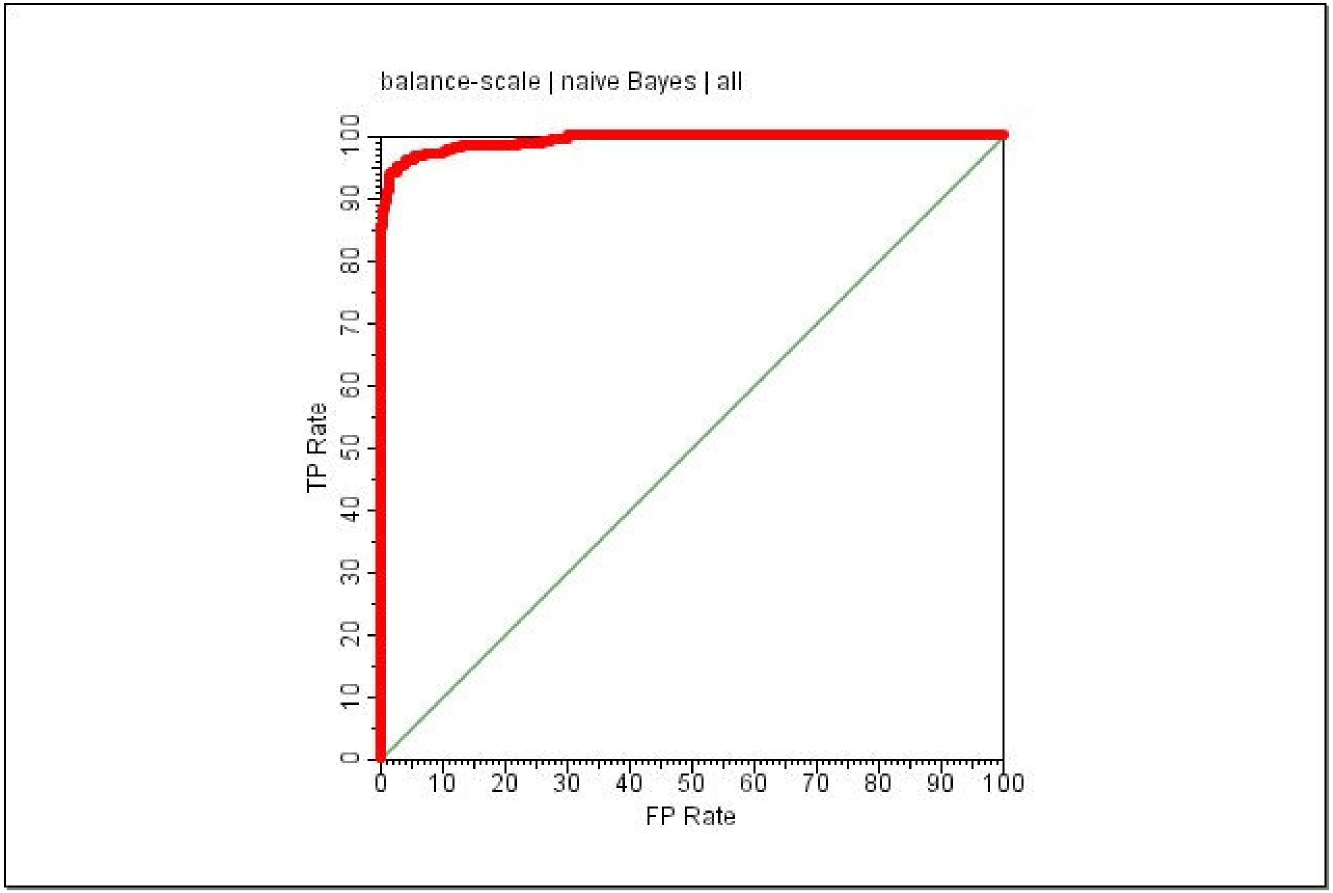
A sample ROC curve



Properties of ROC Curves for Rankers

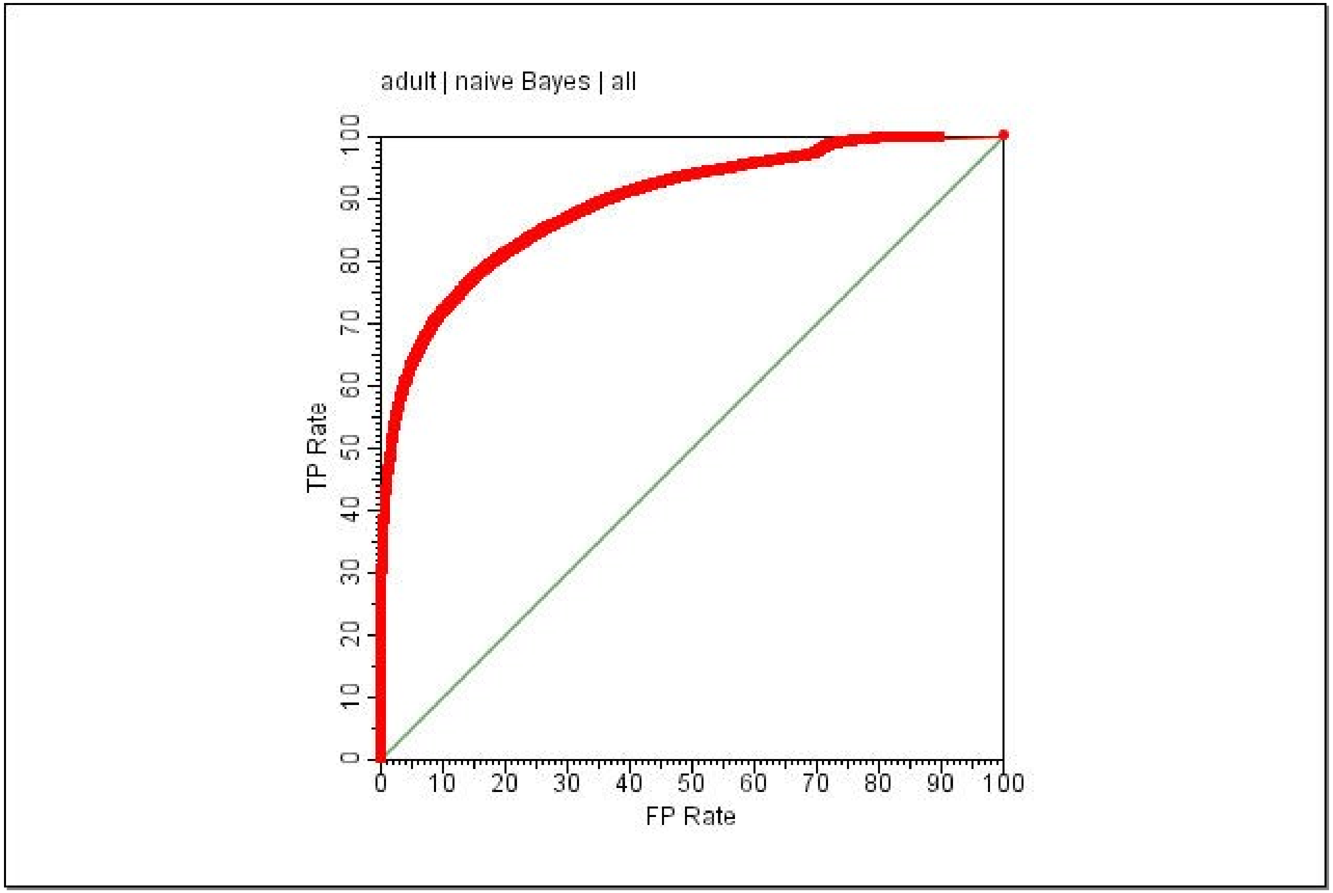
- The curve visualizes the quality of the ranker or probabilistic model on a test set, without committing to a classification threshold
  - aggregates over all possible thresholds
- The slope of the curve indicates class distribution in that segment of the ranking
  - diagonal segment → locally random behaviour
- Concavities indicate locally worse than random behaviour
  - convex hull corresponds to discretizing scores
  - can potentially do better: repairing concavities

Some example ROC curves



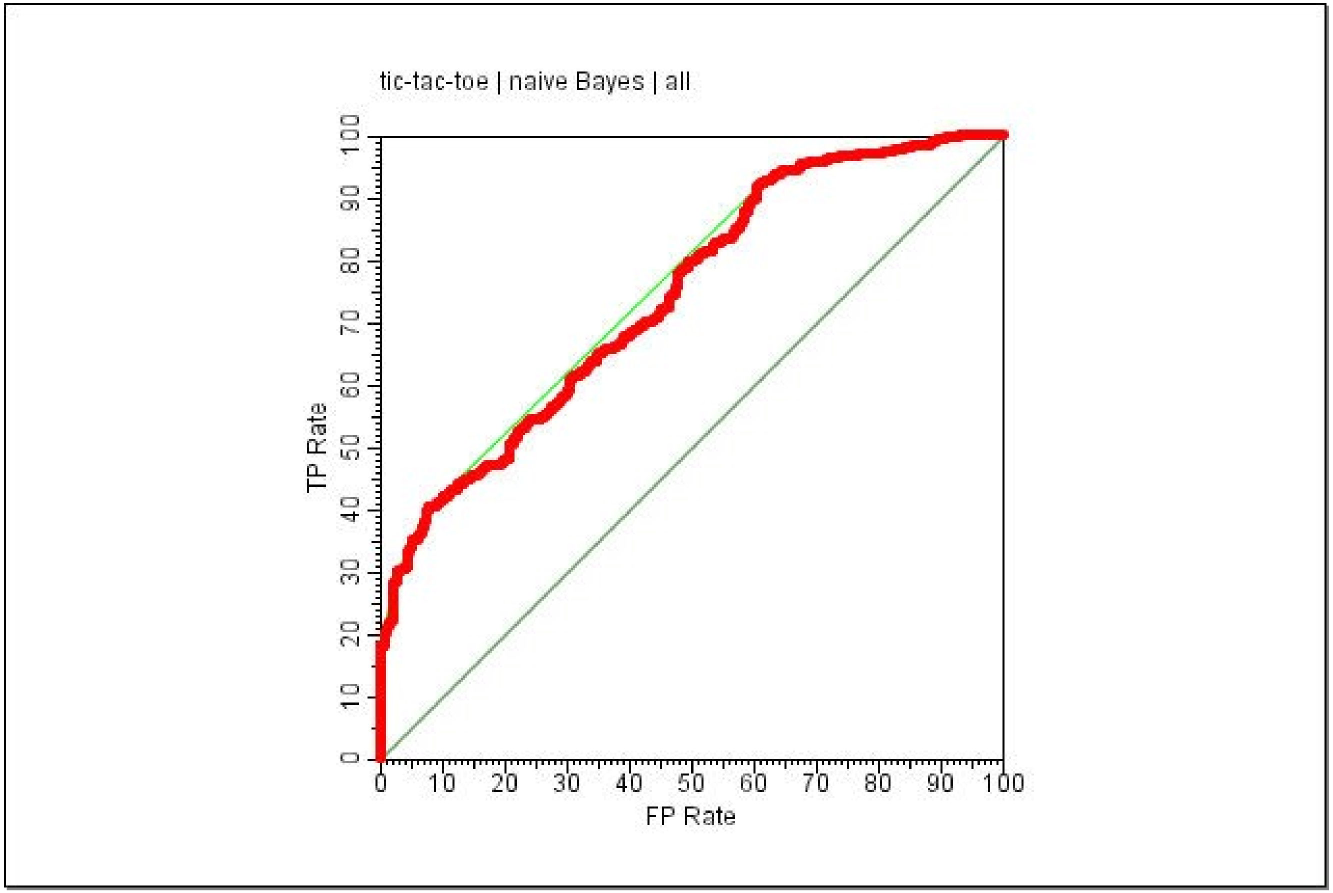
- Good separation between classes, convex curve

Some example ROC curves



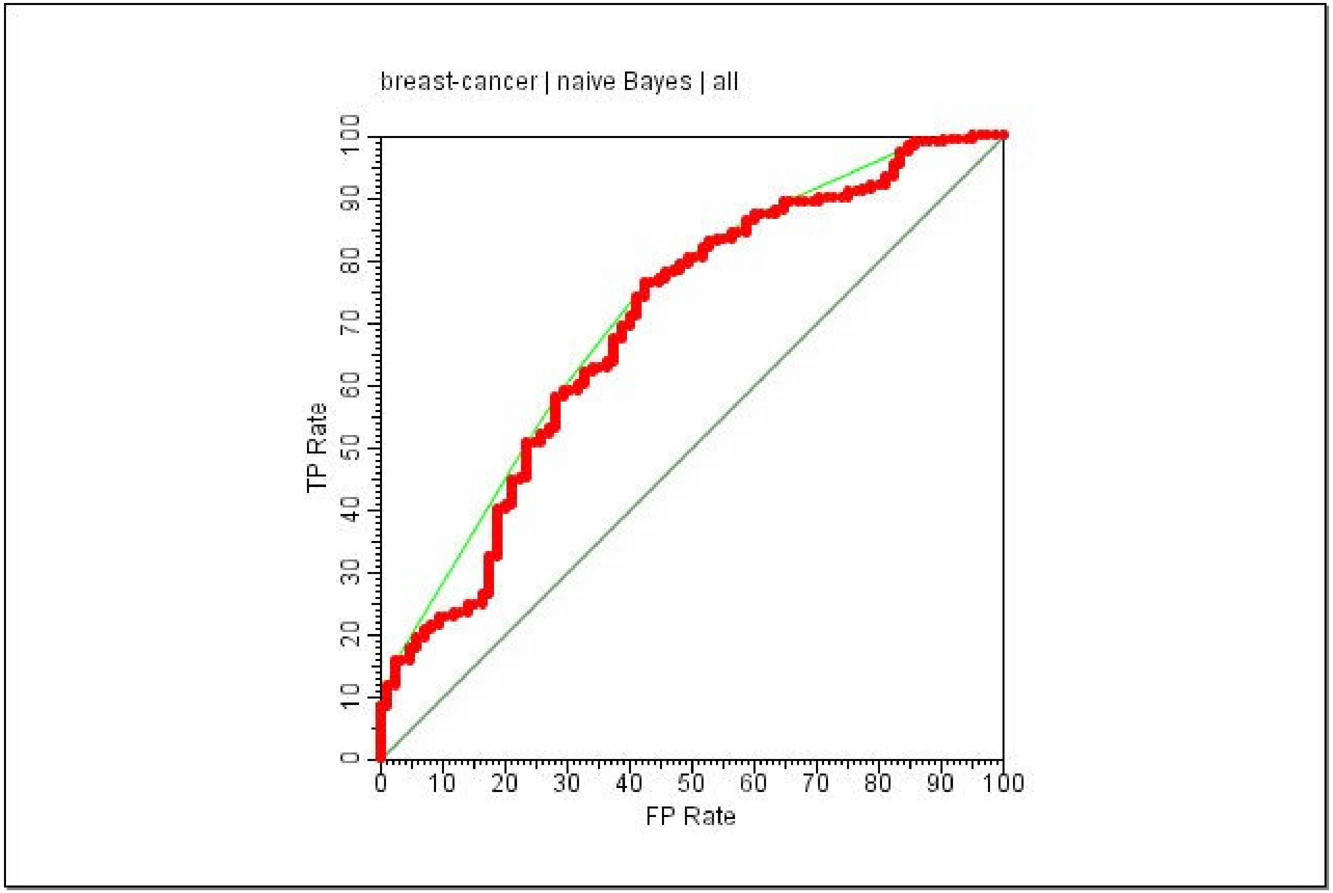
- Reasonable separation, mostly convex

Some example ROC curves



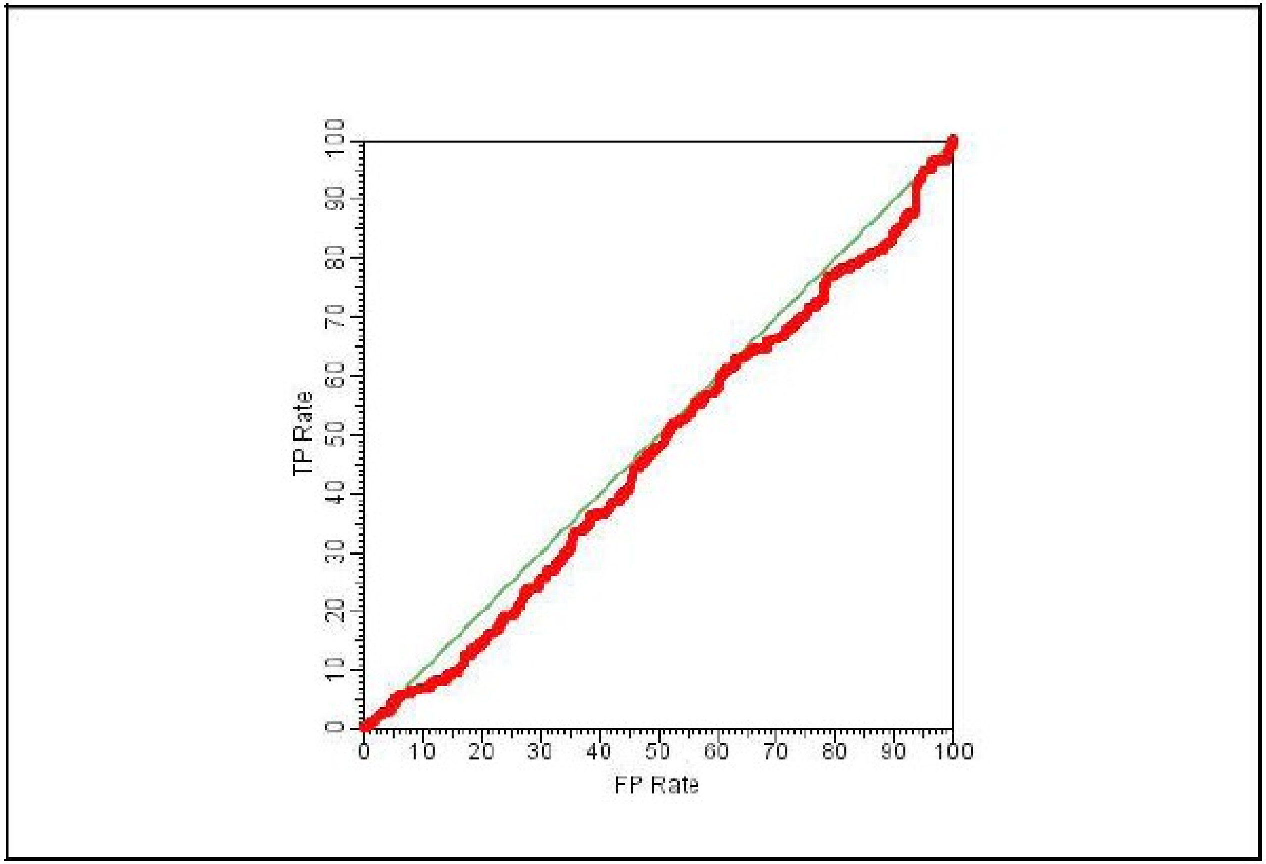
- Fairly poor separation, mostly convex

Some example ROC curves



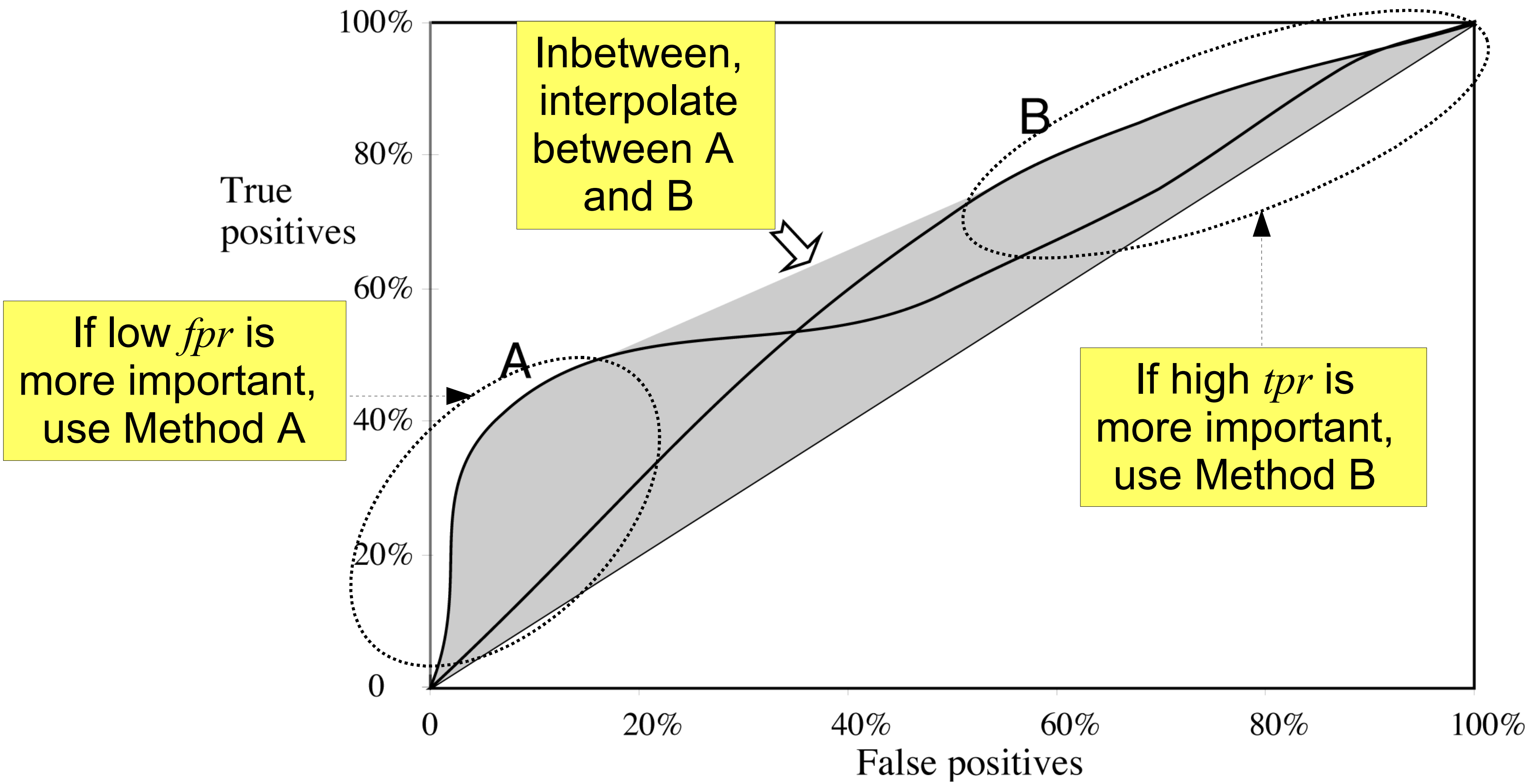
- Poor separation, large and small concavities

Some example ROC curves

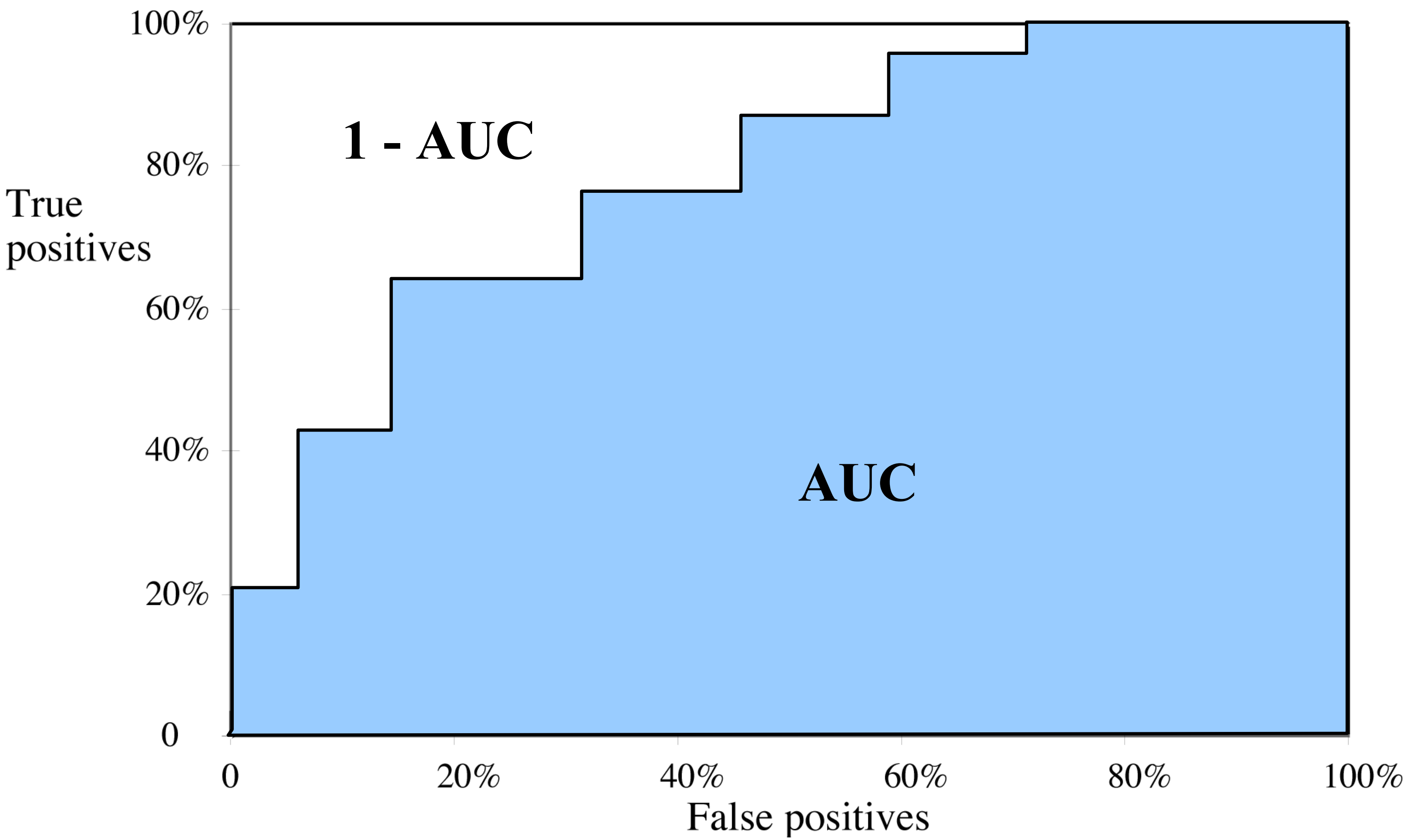


- Random performance

Comparing Rankers with ROC Curves



AUC: The Area Under the ROC Curve



The AUC metric

- The Area Under ROC Curve (AUC) assesses the ranking in terms of separation of the classes
  - all the positives before the negatives: $\mathrm{AUC} = 1$
  - random ordering: $\mathrm{AUC} = 0.5$
  - all the negatives before the positives: $\mathrm{AUC} = 0$
- can be computed from the step-wise curve as:

$$\mathrm{AUC} = \frac{1}{P \cdot N} \sum_{i=1}^{N} (r_i - i) = \frac{1}{P \cdot N} \left(\sum_{i=1}^{N} r_i - \sum_{i=1}^{N} i \right) = \frac{S_- - N(N+1)/2}{P \cdot N}$$

  where r_i is the rank of a negative example and $S_- = \sum_{i=1}^{N} r_i$
- Equivalent to the Mann-Whitney-Wilcoxon sum of ranks test
  - estimates probability that randomly chosen positive example is ranked before randomly chosen negative example

Multi-Class AUC

- ROC-curves and AUC are only defined for two-class problems (concept learning)
  - Extensions to multiple classes are still under investigation
- Some Proposals for extensions:
  - In the most general case, we want to calculate Volume Under ROC Surface (VUS)
    - number of dimensions proportional to number of entries in confusion matrix
  - Projecting down to sets of two-dimensional curves and averaging
    - MAUC (Hand & Till, 2001): $$\mathrm{MAUC} = \frac{2}{c \cdot (c-1)} \sum_{i<j} \mathrm{AUC}(i, j)$$
      - unweighted average of AUC of pairwise classification (1-vs-1)
    - (Provost & Domingos, 2001):
      - weighted average of 1-vs-all, AUC for class c weighted by P(c)

Cost-sensitive learning

- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - For any classifier
    - resampling of instances according to costs
    - proportion of instances with higher weights will be increased
  - If classifier is able to handle weighted instances
    - weighting of instances according to costs
    - covered examples are not counted with 1, but with their weight
  - If classifier returns a score f or probability P
    - varying the classification threshold

Costs and Distributions

- assume no costs for correct classification and a cost ratio $r = c_- / c_+$ for incorrect classifications
  - this means that false positives are r times as expensive as false negatives
- this situation can be simulated by increasing the proportion of negative examples by a factor of r
  - e.g. by replacing each negative example with r identical copies of the same example
  - the number of mistakes on negative examples are then counted with r, the number of mistakes on positive examples are still counted with 1
  - computing the error in the new set corresponds to computing a cost-sensitive evaluation in the original dataset
- the same trick can be used for cost-sensitive *learning*!

Costs and Example Weights

- The effort of duplicating examples can be saved if the learner can use example weights
  - positive examples get a weight of c_+
  - negative examples get a weight of c_-
- All computations that involve counts are henceforth computed with weights
  - instead of counting, we add up the weights
- Example:
  - Precision with weighted examples is $prec = \frac{\sum_{x \in Cov \cap Pos} w_x}{\sum_{x \in Cov} w_x}$

    w_x is the weight of example x
    Cov is the set of covered examples
    Pos is the set of positive examples
  - if $w_x = 1$ for all x, this reduces to the familiar $prec = \frac{p}{p+n}$

Minimizing Expected Cost

- Given a specification of costs for correct and incorrect predictions
  - an example should be predicted to have the class that leads to the lowest expected cost
  - not necessarily to the lowest error
- The expected cost (*loss*) for predicting class i for an example x
  - sum over all possible outcomes, weighted by estimated probabilities

$$L(i,x) = \sum_j C(i|j)\, P(j|x)$$

- A classifier should predict the class that minimizes $L(i,x)$
  - If the classifier can estimate the probability distribution $\mathrm{P}(i \mid x)$ of an example x

Minimizing Cost in Concept Learning

- For two classes:
  - predict positive if it has the smaller expected cost:

$$\underbrace{C(+|+)\cdot P(+|x) \;+\; C(+|-)\cdot P(-|x)}_{\text{cost if we predict positive}} \;\le\; \underbrace{C(-|+)\cdot P(+|x) \;+\; C(-|-)\cdot P(-|x)}_{\text{cost if we predict negative}}$$

  - as $P(+|x) = 1 - P(-|x)$:

predict positive if $P(+|x) \ge \dfrac{C(+|-) - C(-|-)}{C(+|-) + C(-|+) - C(+|+) - C(-|-)}$

- Example:
  - Classifying a spam mail as ham costs 1, classifying ham as spam costs 99, correct classification cost nothing:

$\Rightarrow$ classify as spam if spam-probability is at least 99%

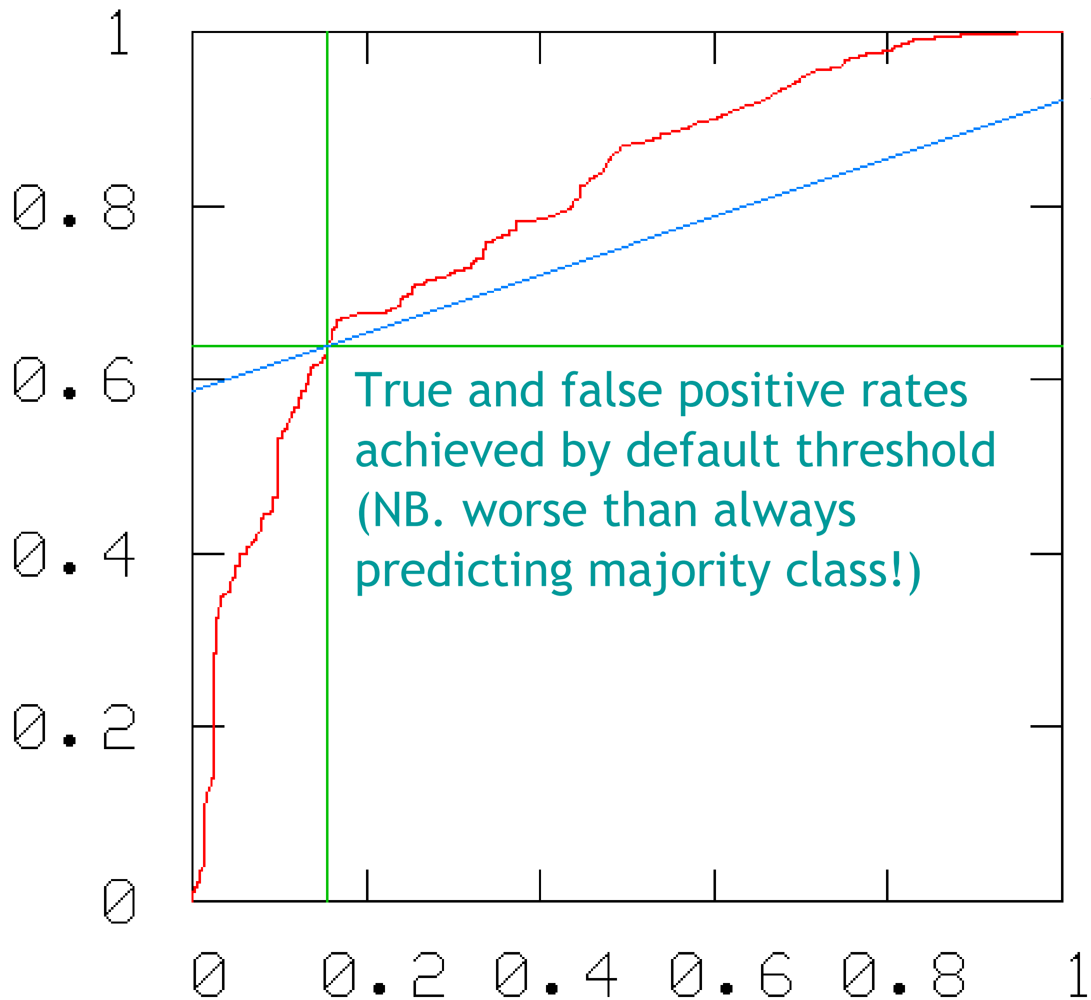
Calibrating a Ranking Classifier

- What is the right threshold of the ranking score if the ranker does not estimate probabilities?
  - classifier can be *calibrated* by choosing appropriate threshold that minimizes costs
  - may also lead to improved performance in accuracy if probability estimates are bad (e.g., Naïve Bayes)

- Easy in the two-class case:
  - calculate cost for each point/threshold while tracing the curve
  - return the threshold with minimum cost
- Non-trivial in the multi-class case

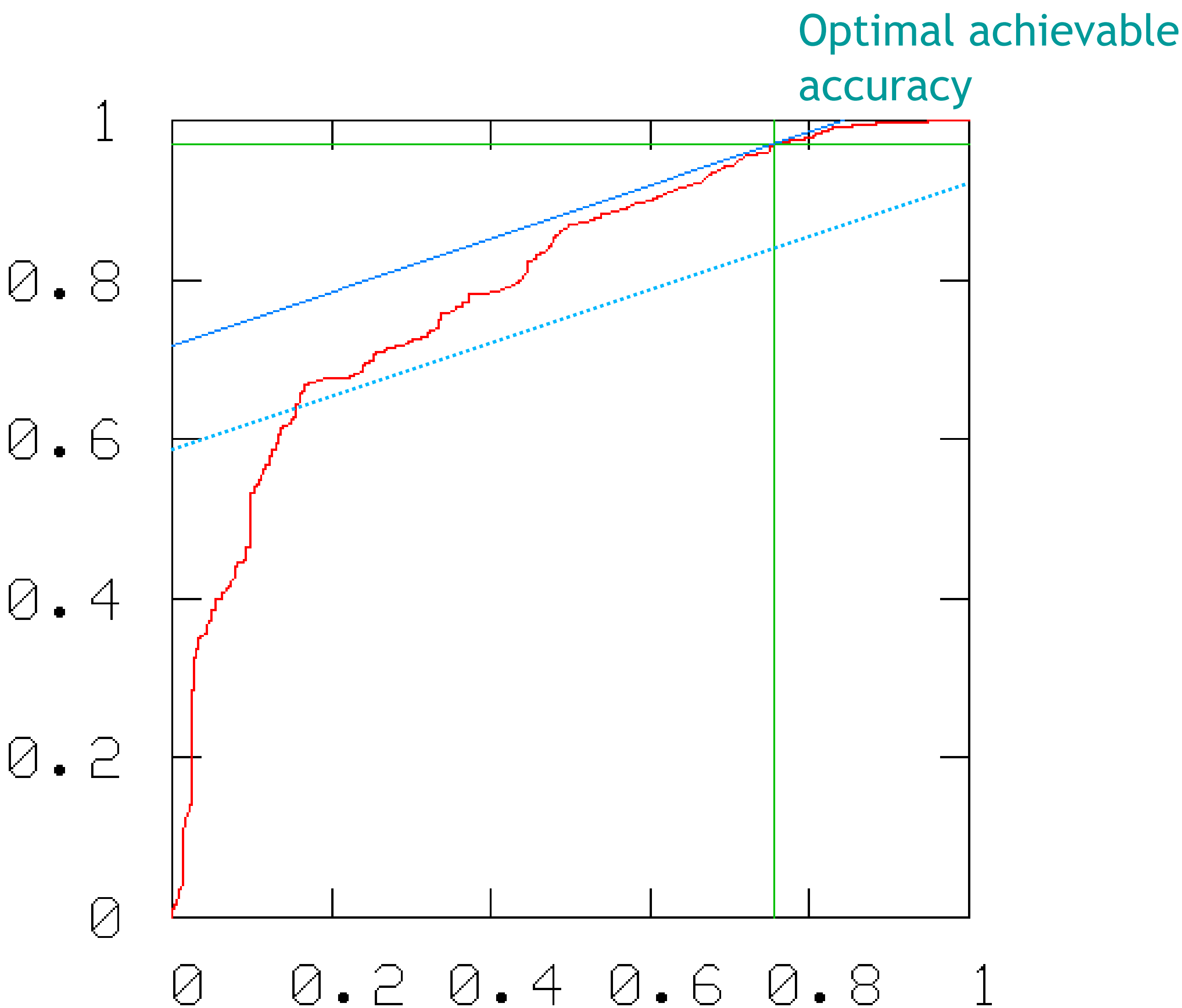
Note: threshold selection is part of the classifier training and must therefore be performed on the training data!

Example: Uncalibrated threshold



Accuracy isometric for this domain

Example: Calibrated threshold



References

- Charles Elkan: *The Foundations of Cost-Sensitive Learning.* In Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), pp. 973-978.
  http://www-cse.ucsd.edu/users/elkan/rescale.pdf
- Peter Flach: *The many faces of ROC analysis in machine learning*, Tutorial held at ICML-04.
  http://www.cs.bris.ac.uk/~flach/ICML04tutorial/
- David J. Hand, Robert J. Till: A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning 45(2): 171-186 (2001)
- Foster Provost and Tom Fawcett (2001). Robust classification for imprecise environments. Machine Learning, 42, 203–231.
  http://pages.stern.nyu.edu/~fprovost/Papers/rocch-mlj.pdf
- Ian Witten and Eibe Frank: *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edition, Morgan Kaufmann 2005. Chapter 5.
  http://www.cs.waikato.ac.nz/~ml/weka/book.html