Putting things in order

On the fundamental role of ranking in classification and probability estimation

Peter.Flach@bristol.ac.uk

Peter A. Flach

www.cs.bris.ac.uk/~flach/

Machine Learning and
Biological Computation Group

Department of
Computer Science

University of
Bristol

Motivation

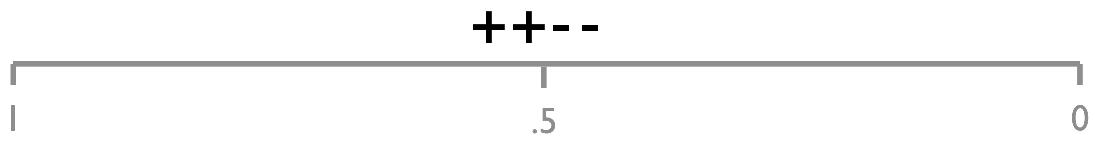
- Three typical machine learning tasks
  - (binary) classification: distinguish between positives and negatives
  - (bipartite) ranking: order the positives before the negatives
  - probability estimation: model the posterior probability distribution p(+|x)
- Q: Is there a natural hierarchy among these tasks? E.g., is a good probability estimator also a good ranker?
- Q: Is it simply a matter of adjusting the loss function when training a model to achieve a different task?

Motivation (2)

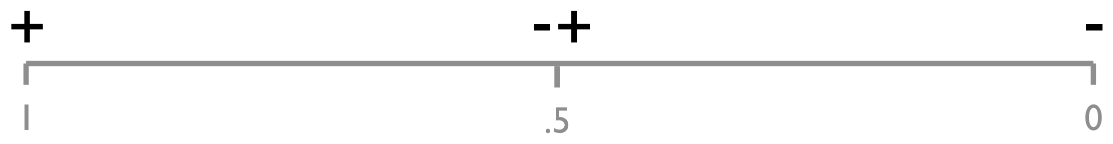
- Q: Why do some models, trained to be good classifiers, turn out to be good rankers?
    - Steck (ECML’07): SVMs approximately maximise AUC
    - Rudin et al. (COLT’05): AdaBoost achieves same AUC as RankBoost
- Q: Which models produce good probability estimates, and why? How do we measure that? Which calibration procedure is appropriate for which model?
    - Niculescu-Mizil & Caruana (ICML’05): neural networks and bagged trees are well-calibrated; SVMs and naive Bayes produce distinct distortions

Motivation (3)

- Better probabilities ≠ better ranking

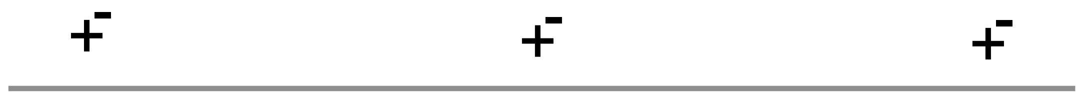


  - no ranking errors, mean squared error ≈ 0.25

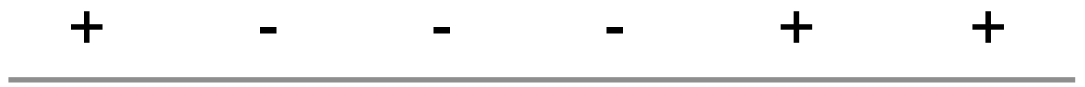


  - 1 ranking error (worse), mean squared error ≈ 0.13 (better)

- Better classification ≠ better ranking



  - 4.5 ranking errors, 3 classification errors



  - 6 ranking errors (worse), 2 classification errors (better)

Outline

I. Building models

- ROC plots give a wealth of insight in the behaviour of models

II. Classification and ranking

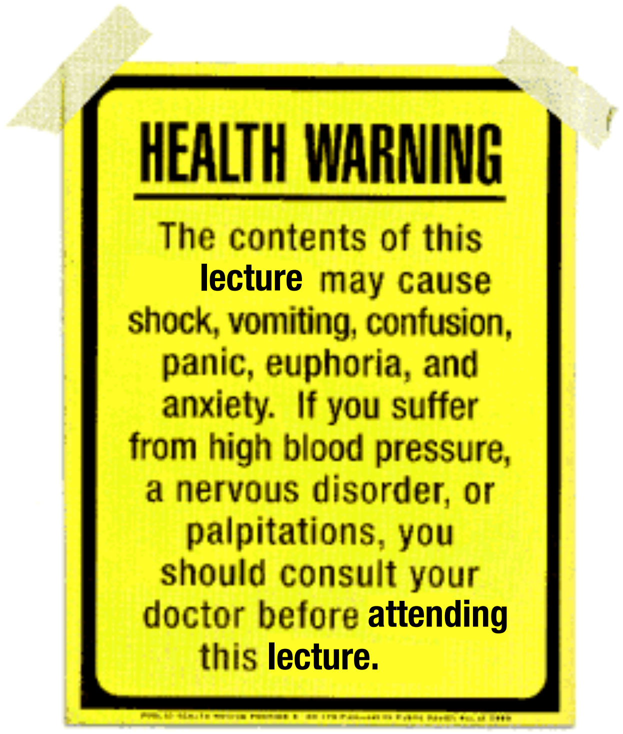
- Several relationships between AUC and accuracy exist, but they are still distinct optimisation criteria

III. Ranking and probability estimation

- How to measure and improve the quality of probability estimates

Health warning

- ROC curves are mostly constructed from the training set in this talk!
  - This is not to say that generalisation and overfitting are not issues.
  - But my emphasis here is on *training*, rather than evaluating, models.
  - Training set ROC curves can help us to understand, and improve, the behaviour of models.

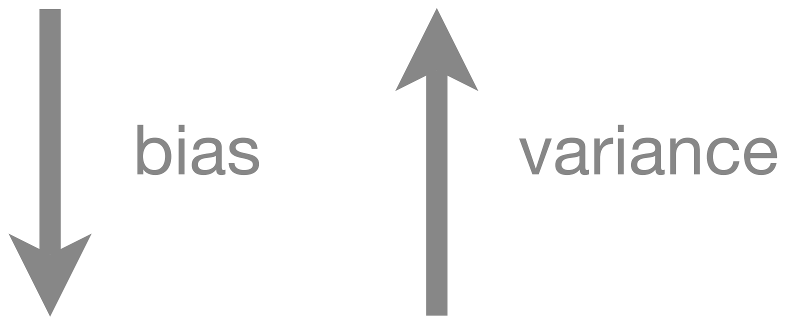


HEALTH WARNING

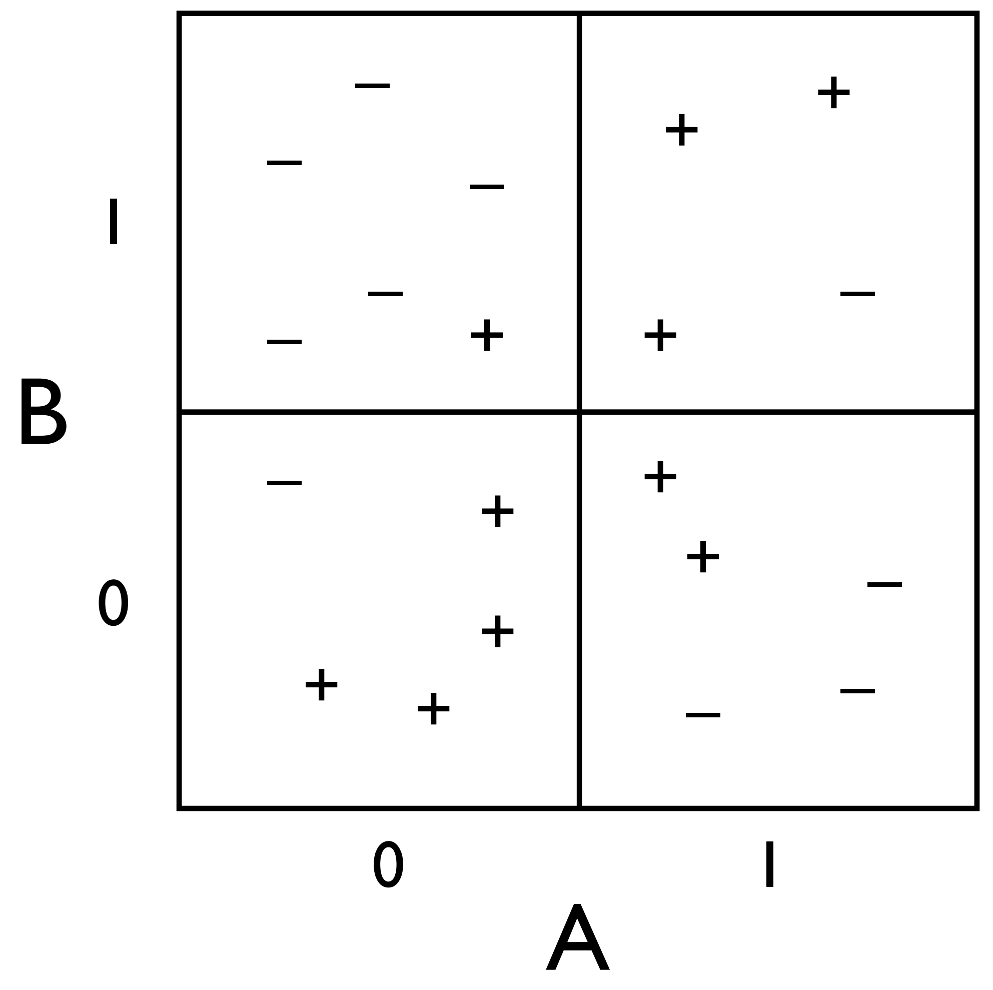
The contents of this lecture may cause shock, vomiting, confusion, panic, euphoria, and anxiety. If you suffer from high blood pressure, a nervous disorder, or palpitations, you should consult your doctor before attending this lecture.

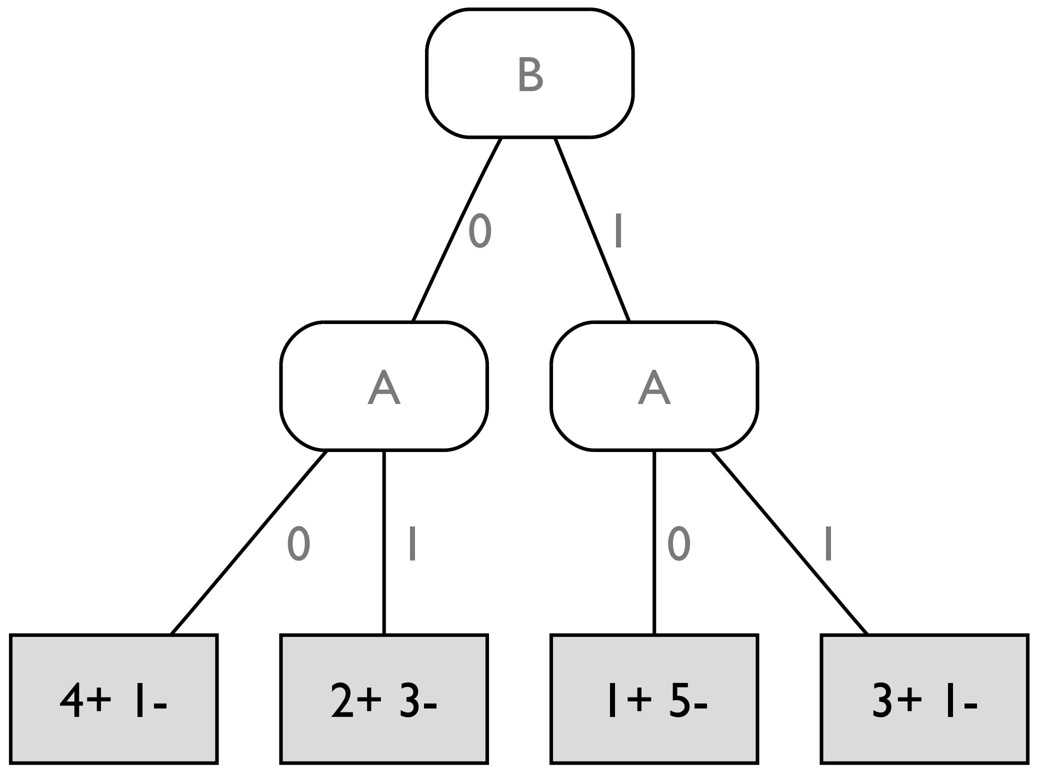
I Building models

- In this talk I will consider the following models:
  - decision tree
  - naive Bayes
  - LexRank

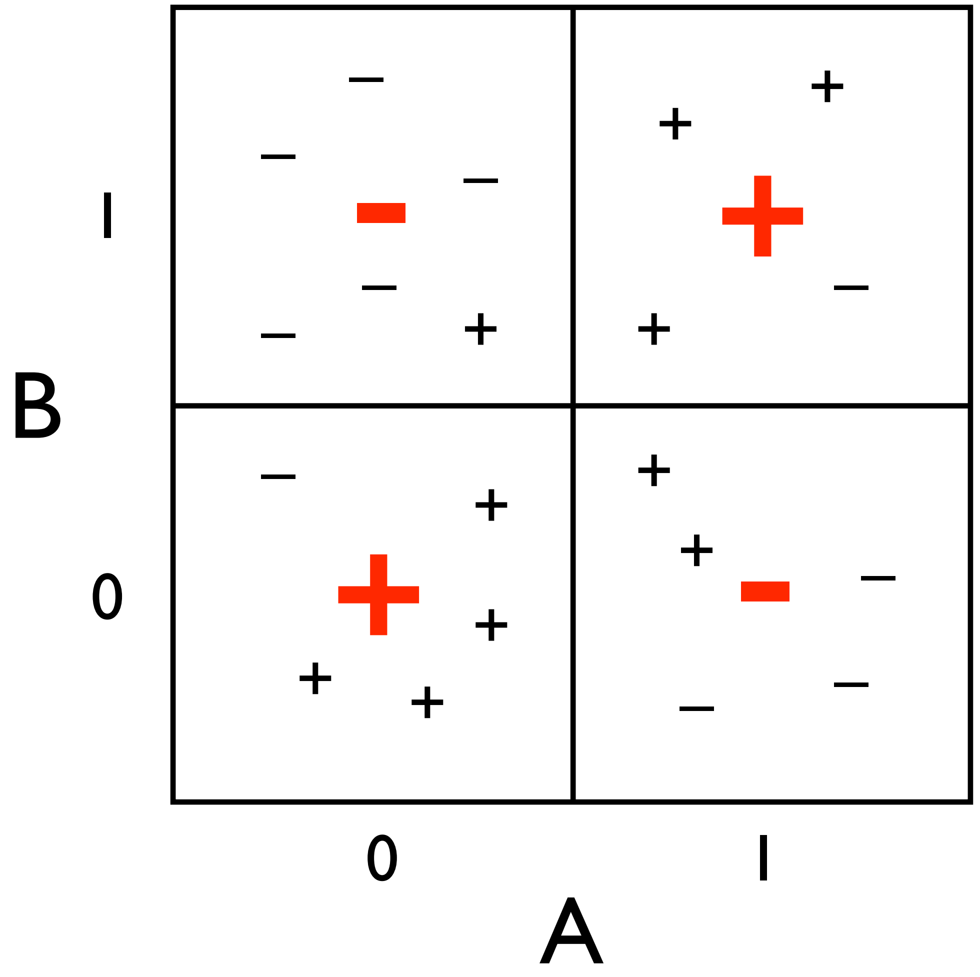


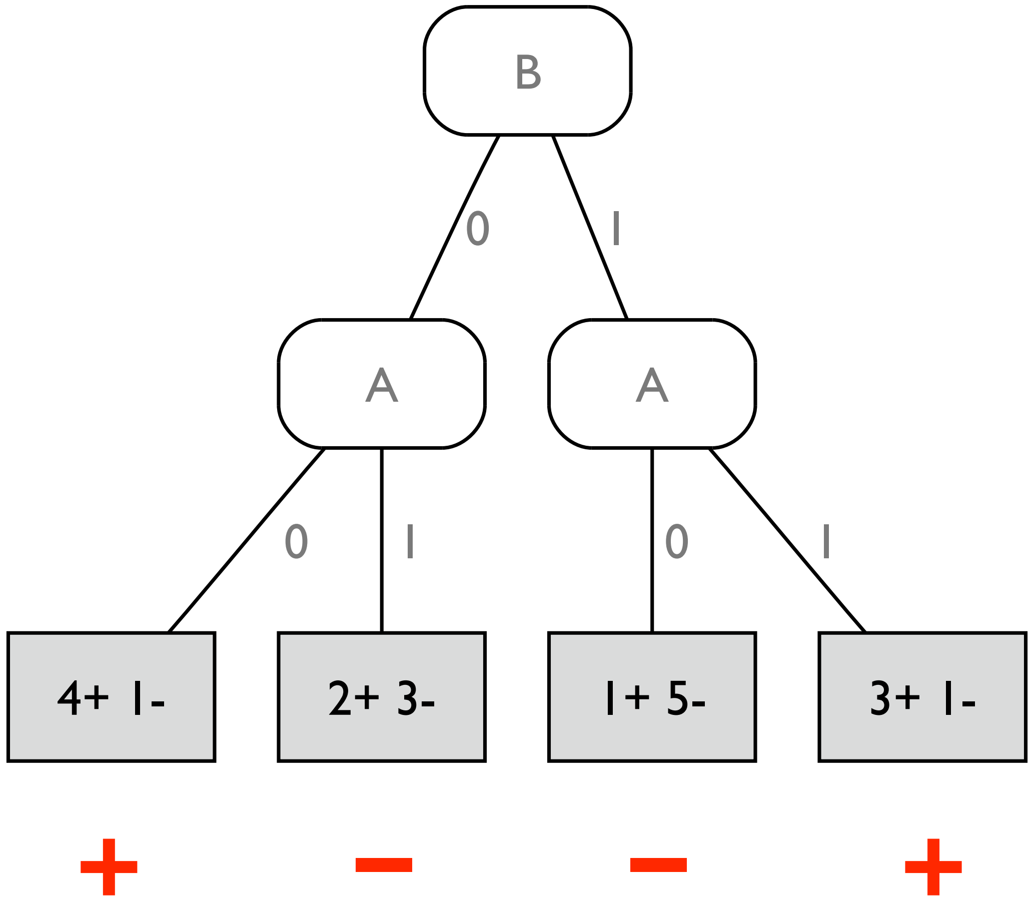
Decision tree classifier





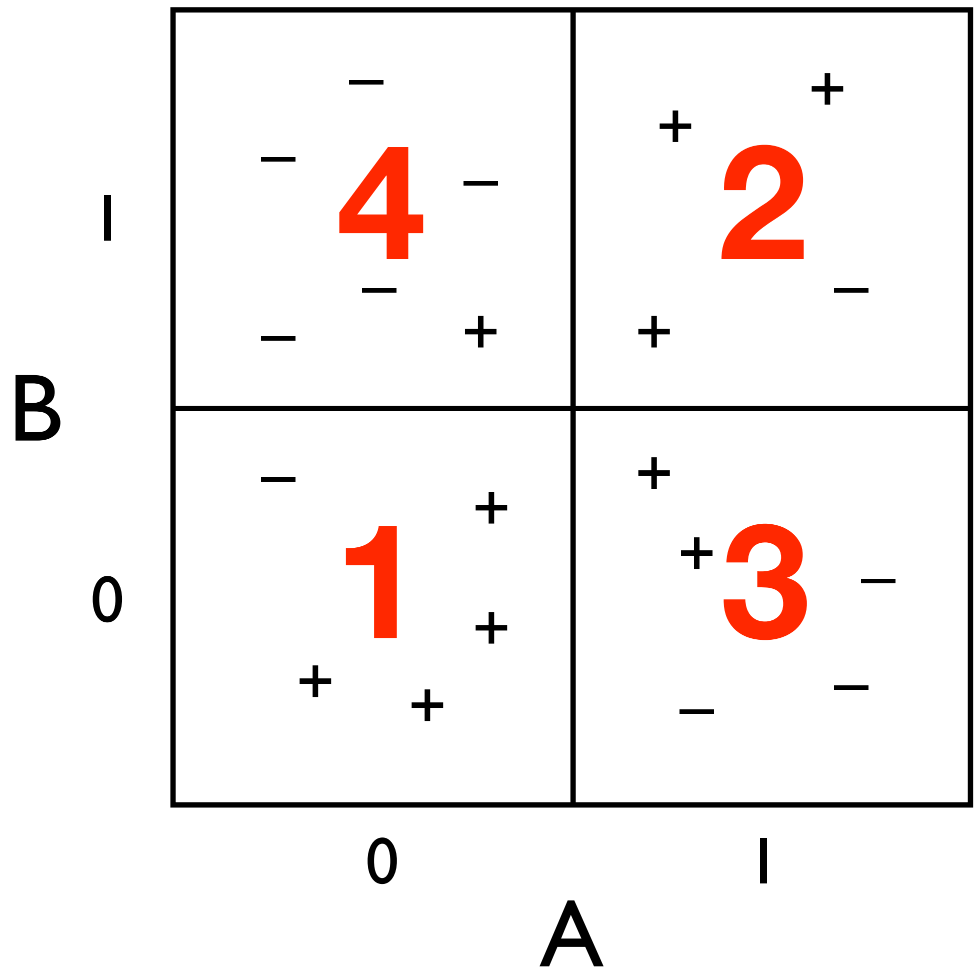
Decision tree classifier

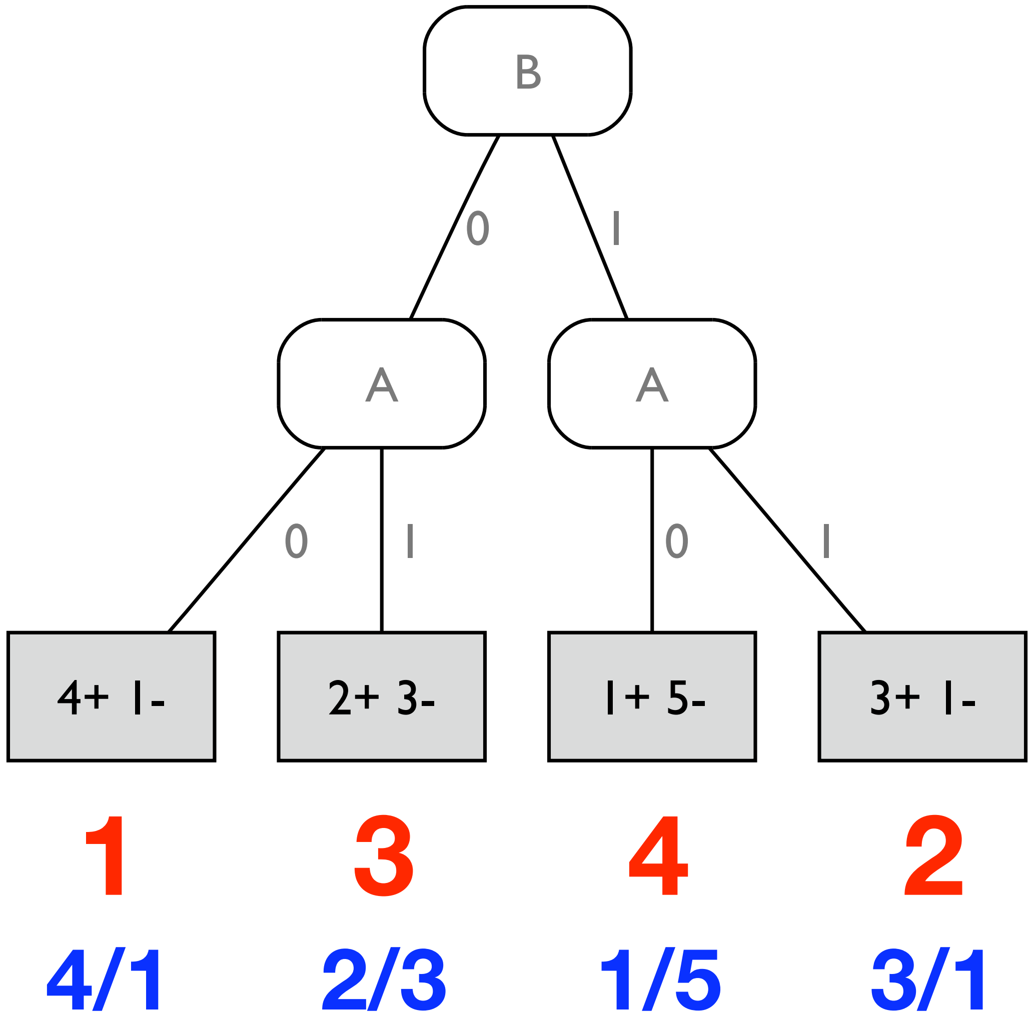


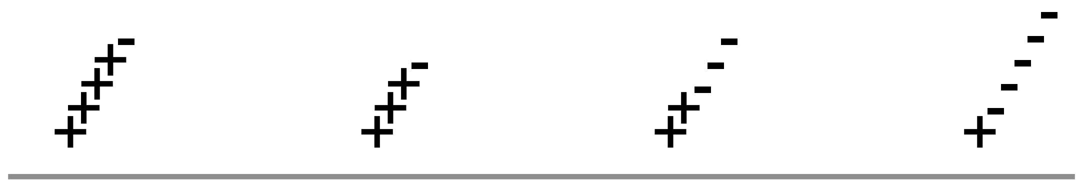


Labels obtained by majority class decision rule.

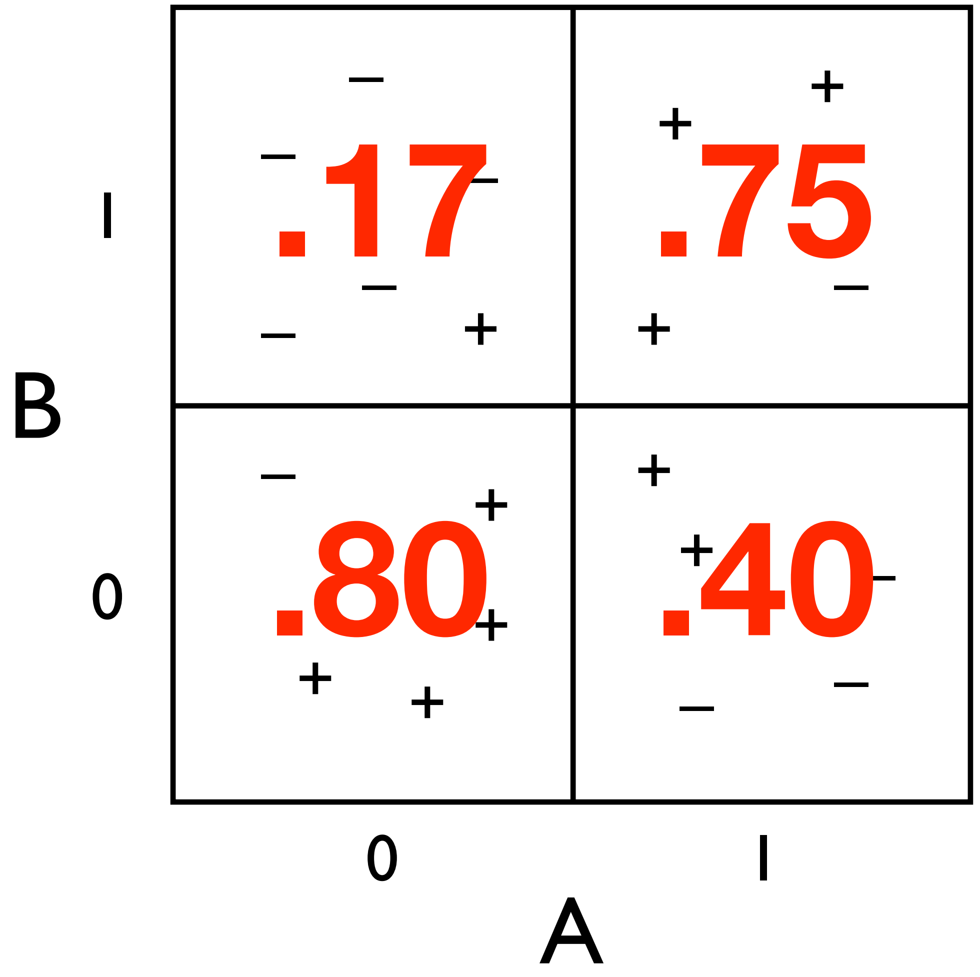
Decision tree ranker

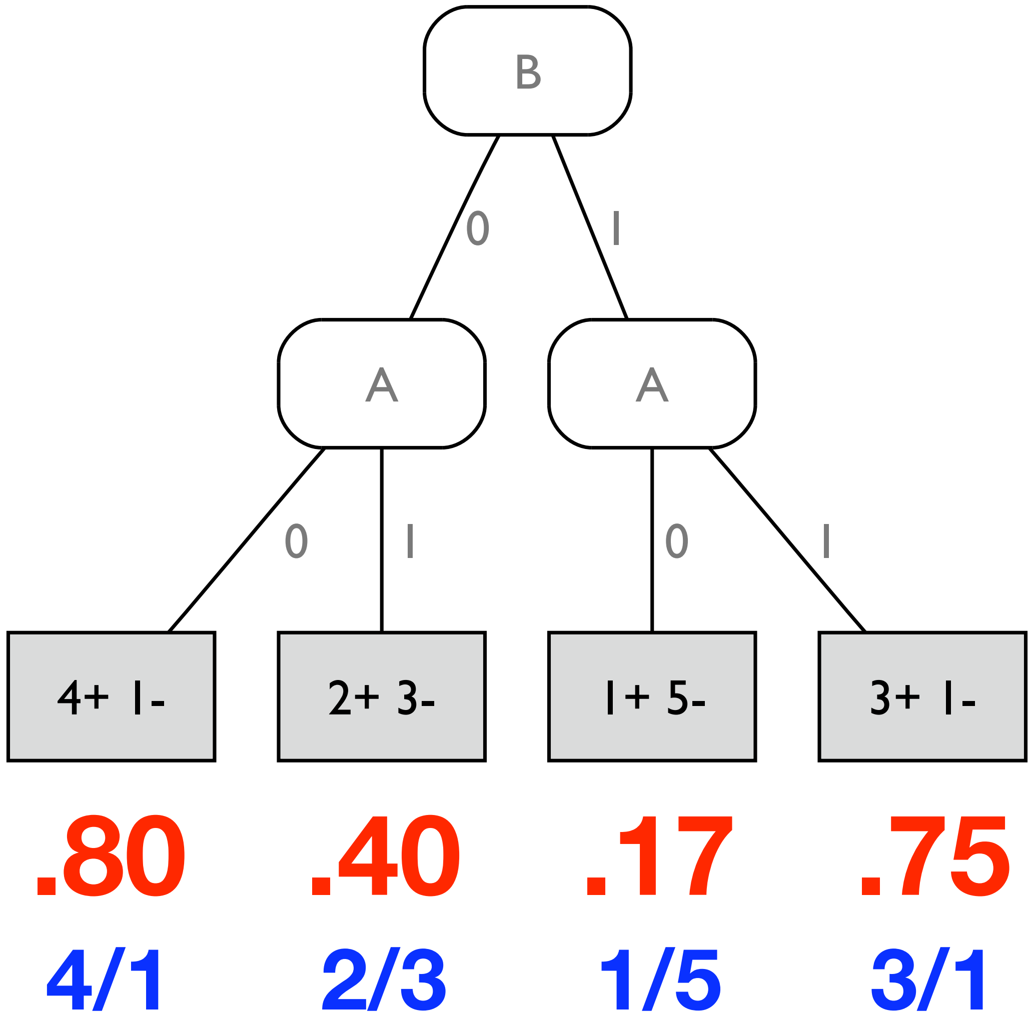


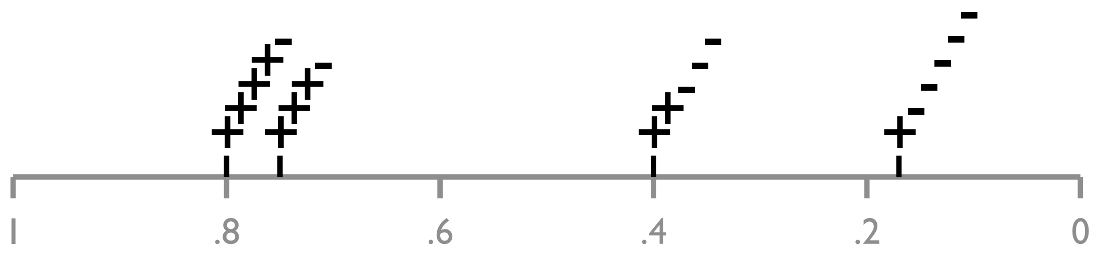




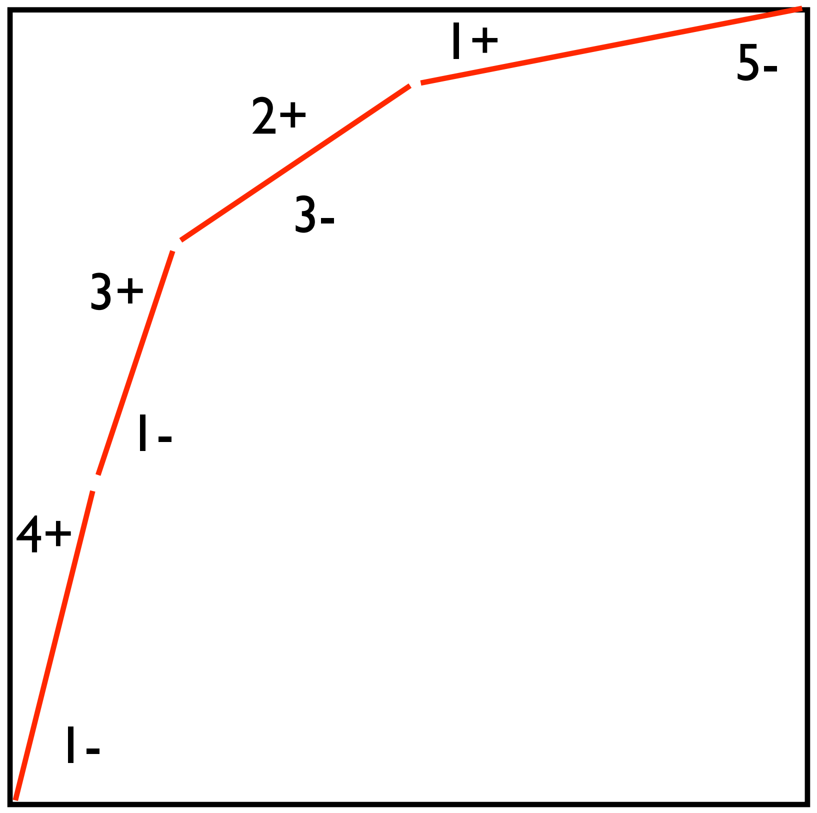
Decision tree probability estimator

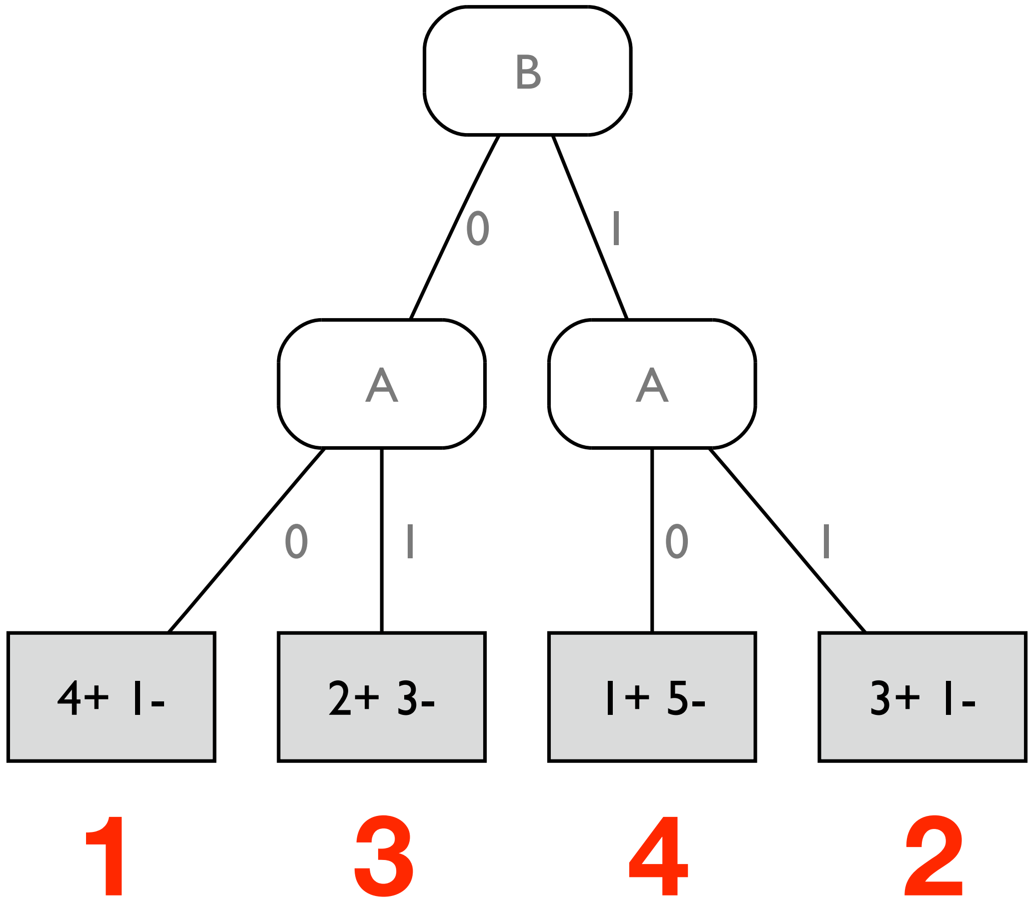






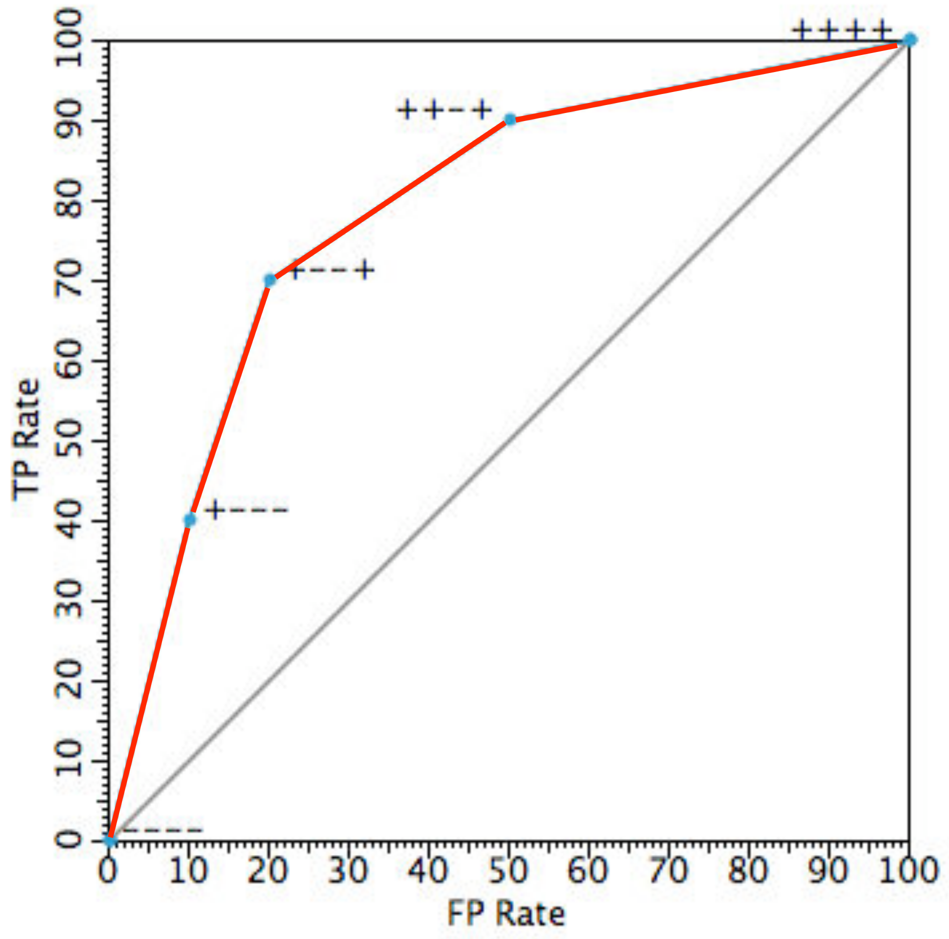
Visualising ranking performance

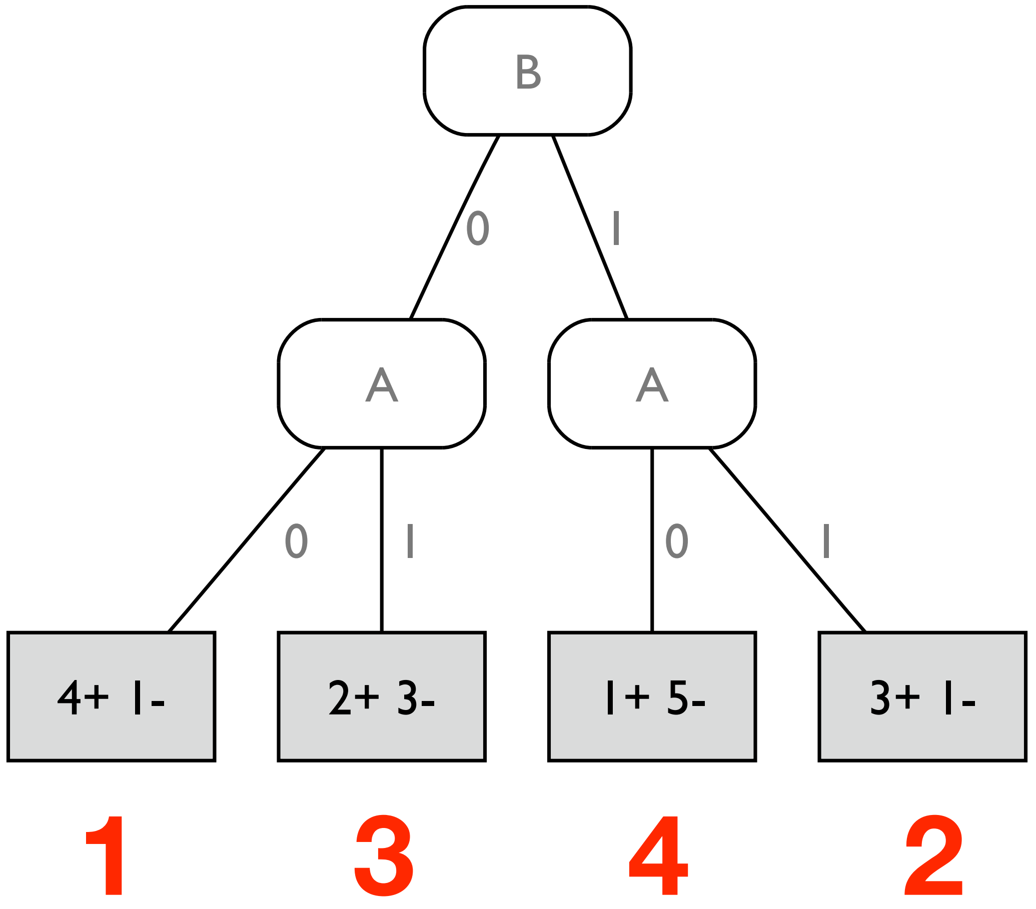




Each leaf is visualised by a line segment; by stacking these line segments in the ranking order we can keep track of cumulative performance (aka Lorenz curve or ROC curve).

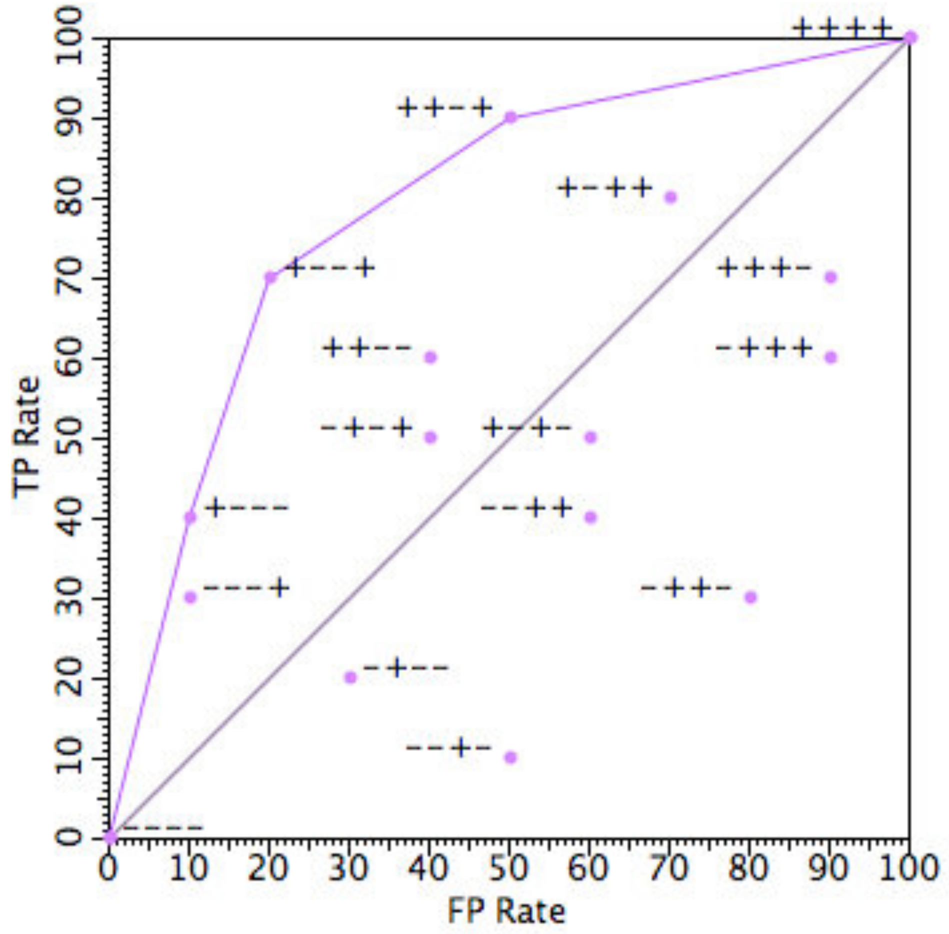
Visualising ranking performance (2)

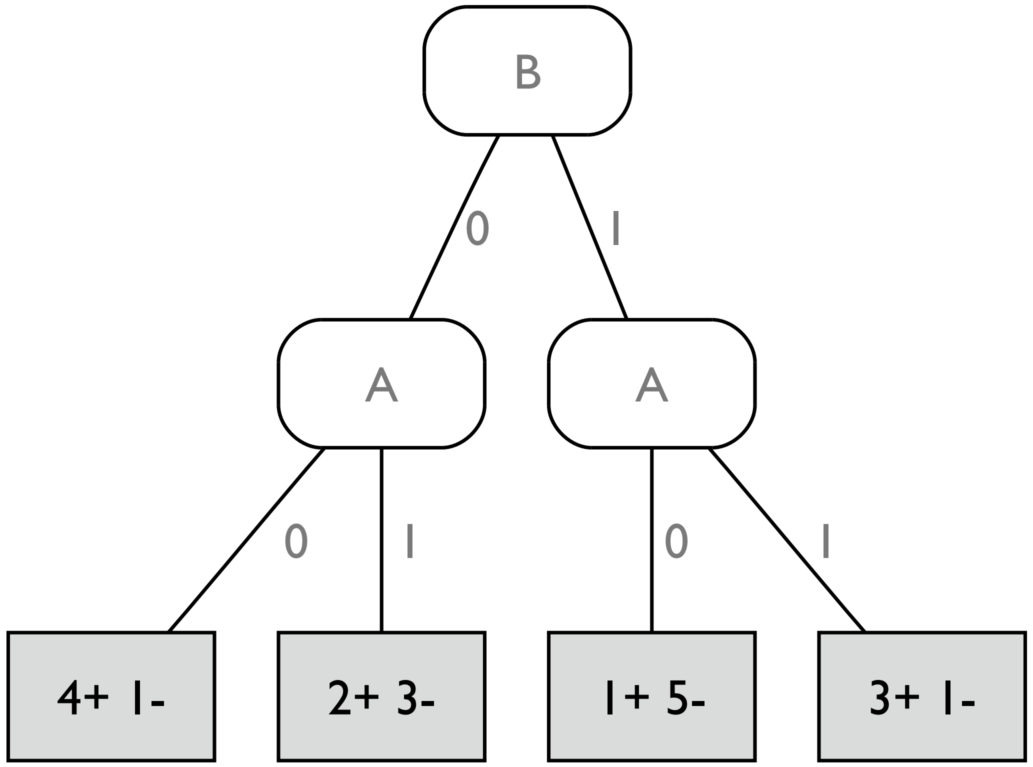




Absolute numbers on the axes mean that slopes represent *posterior odds*; normalising these by the number of positives/negatives means that slopes represent *likelihood ratios* instead.

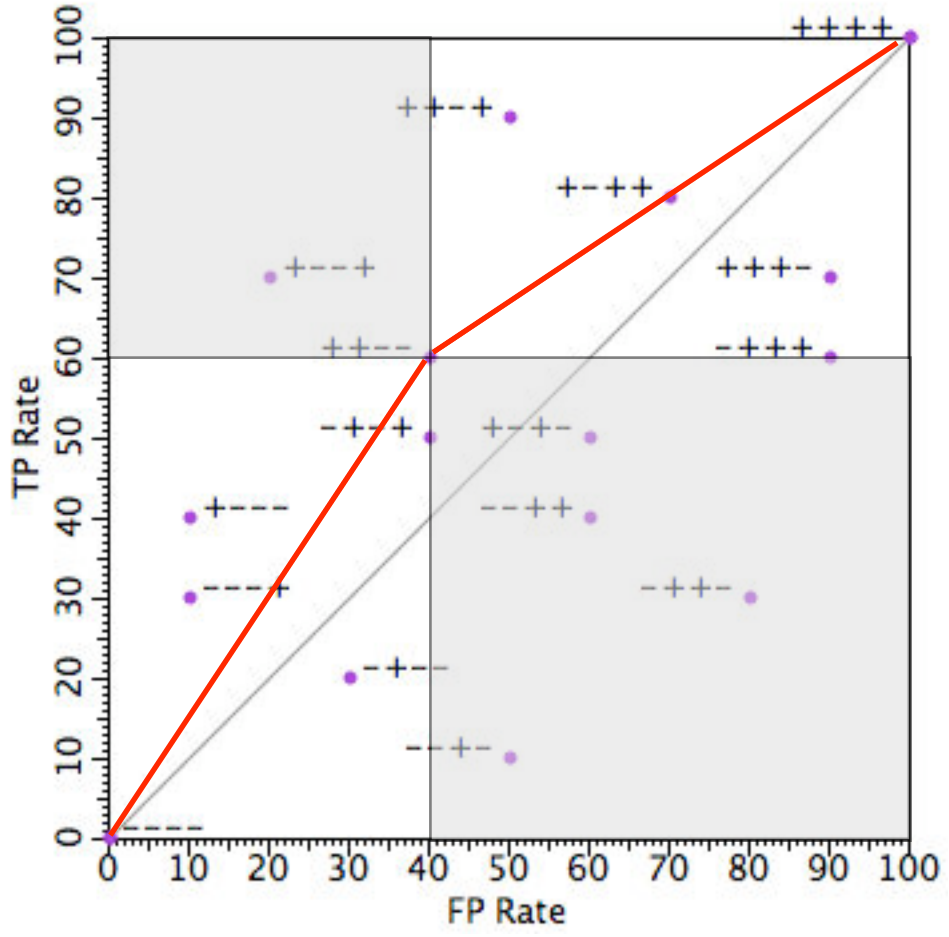
All possible tree labellings

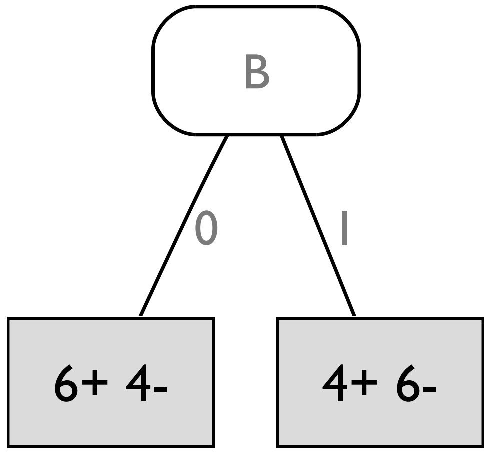




A tree with *n* leaves has 2^n possible labellings, which sum up all possible model behaviours. Notice that a labelling and its opposite (e.g., +—+ and –++–) are each other's mirror image in ROC space (through (1/2,1/2)).

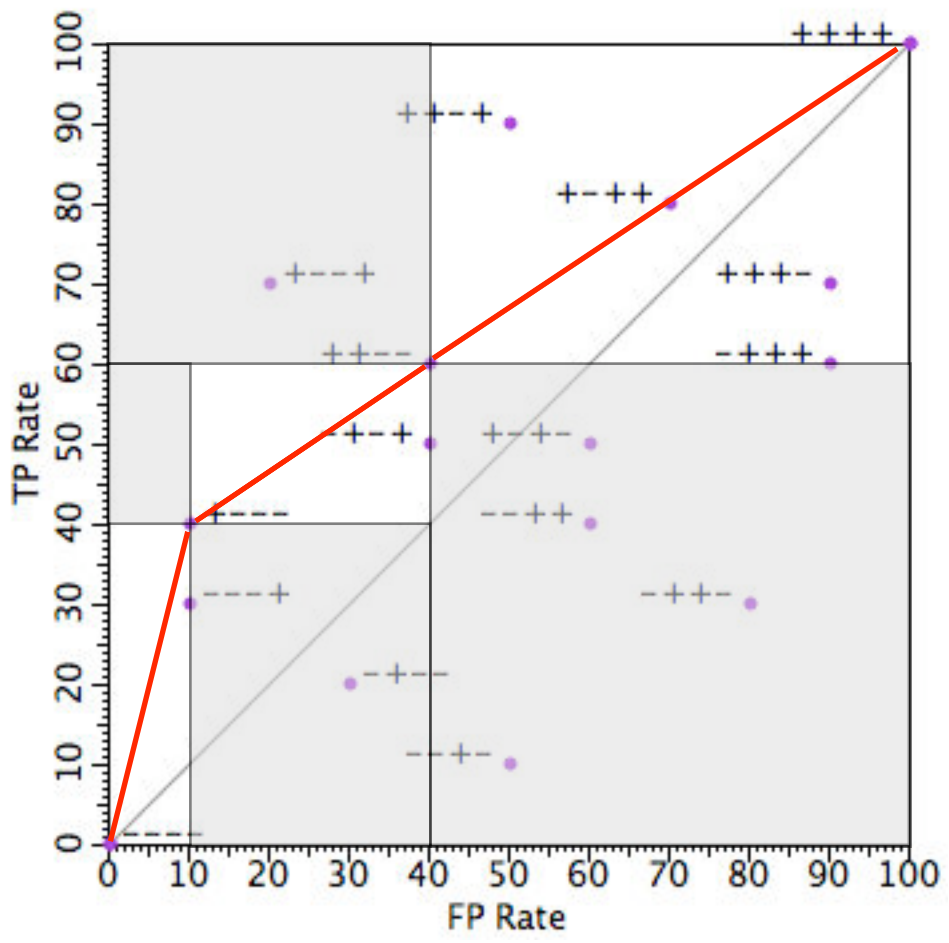
Building the tree recursively (1)

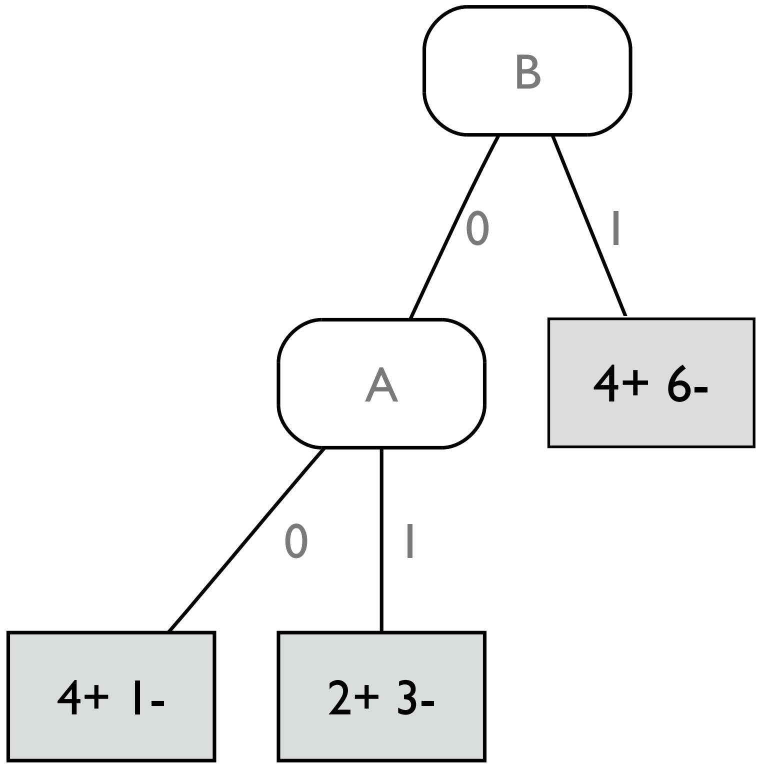




The first split partitions the data in two subsets, each of which is then recursively split again.

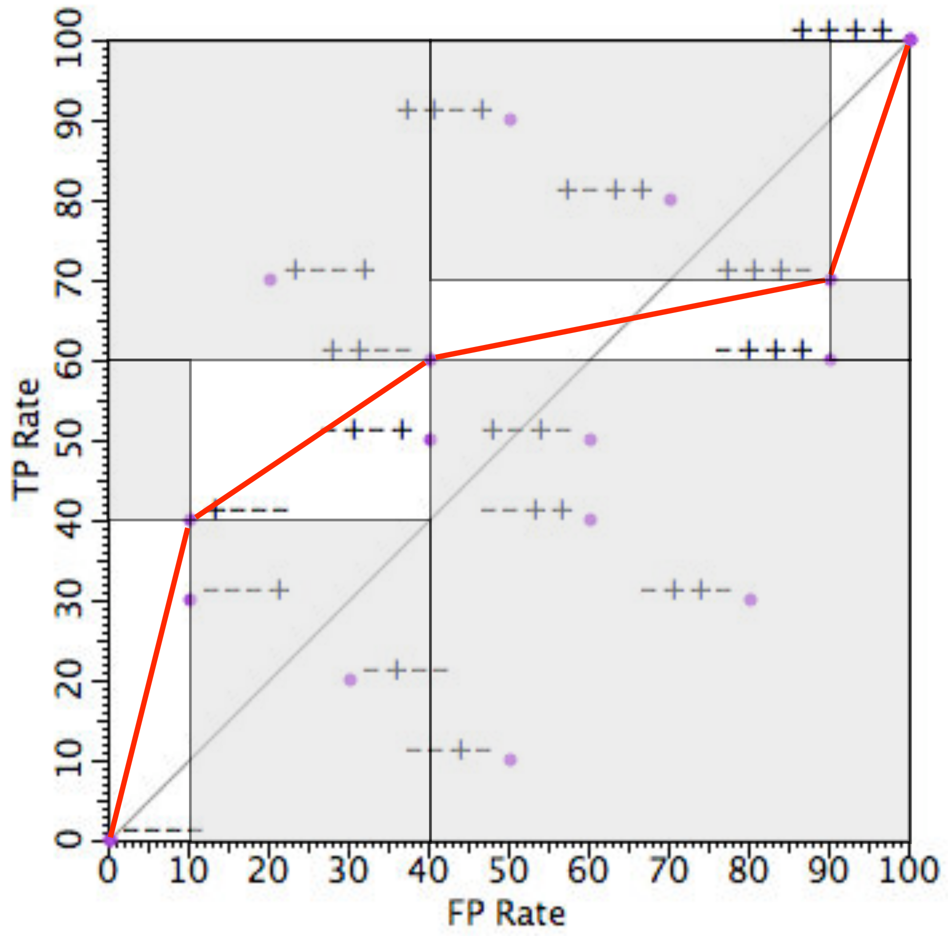
Building the tree recursively (2)

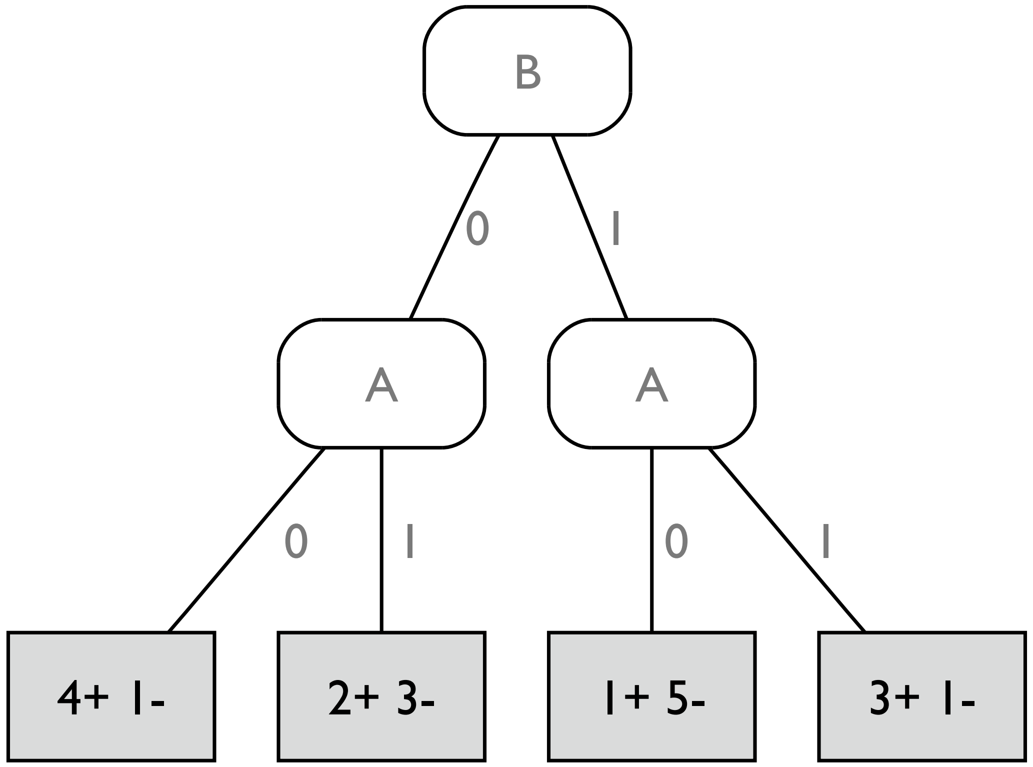




Second split.

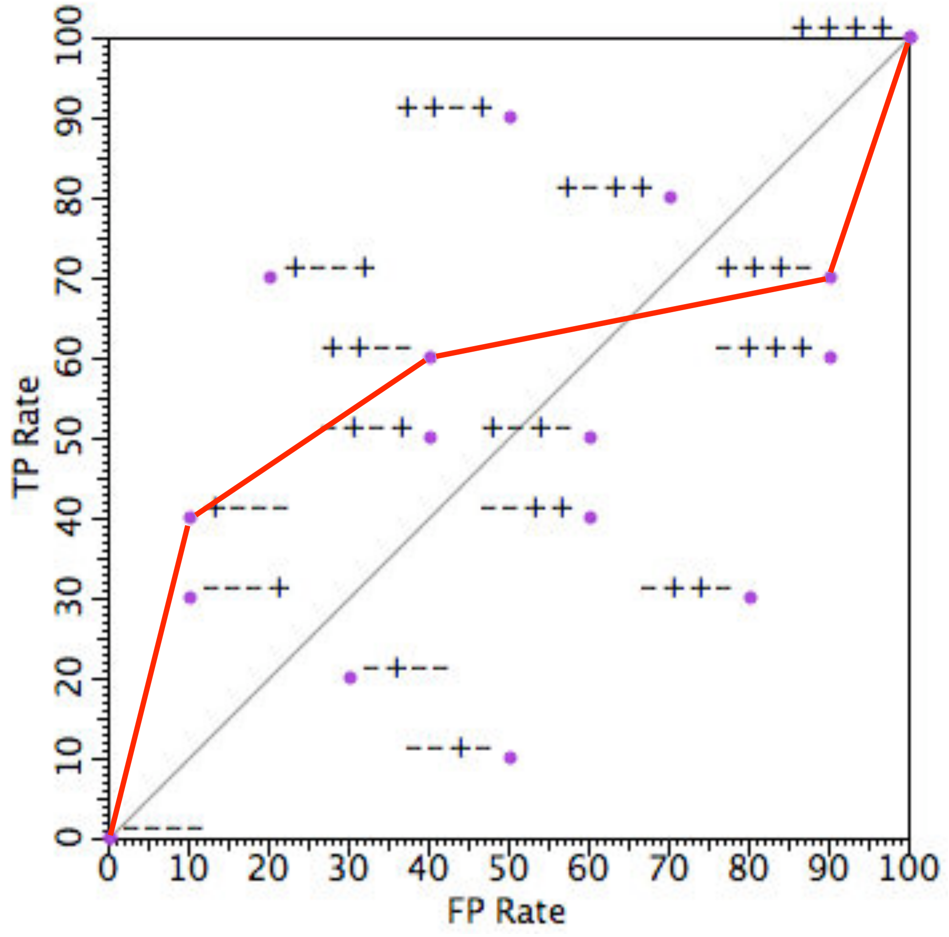
Building the tree recursively (3)

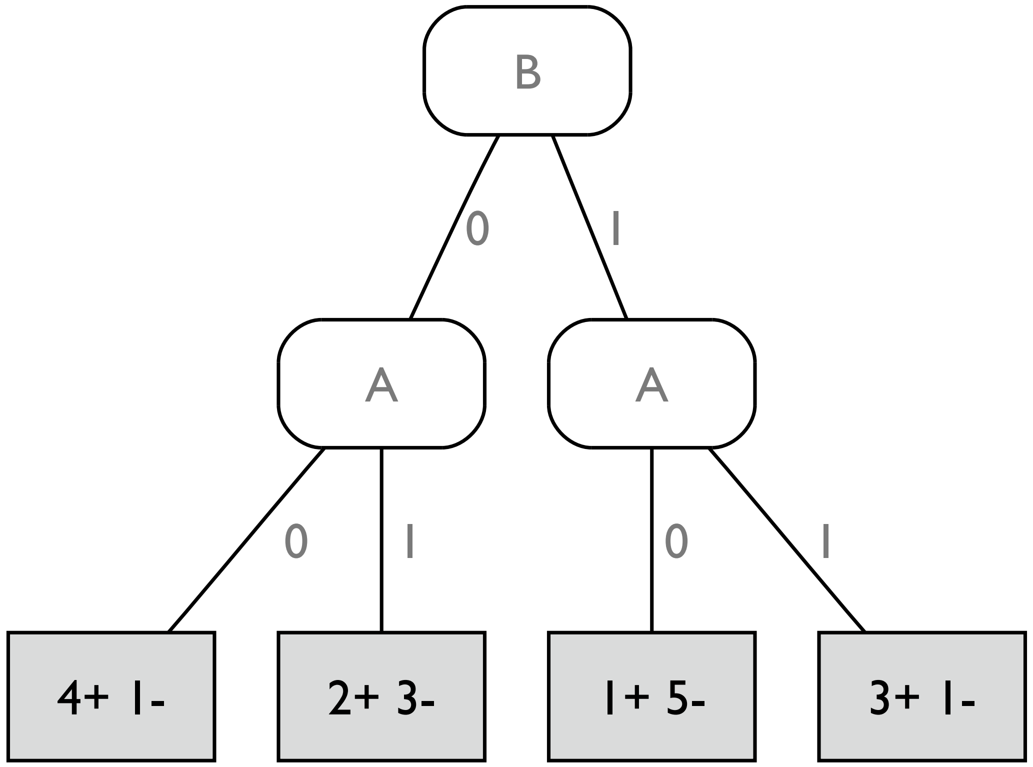




Notice that the final split (top-right, +++–) could equally have been represented by ++–+.

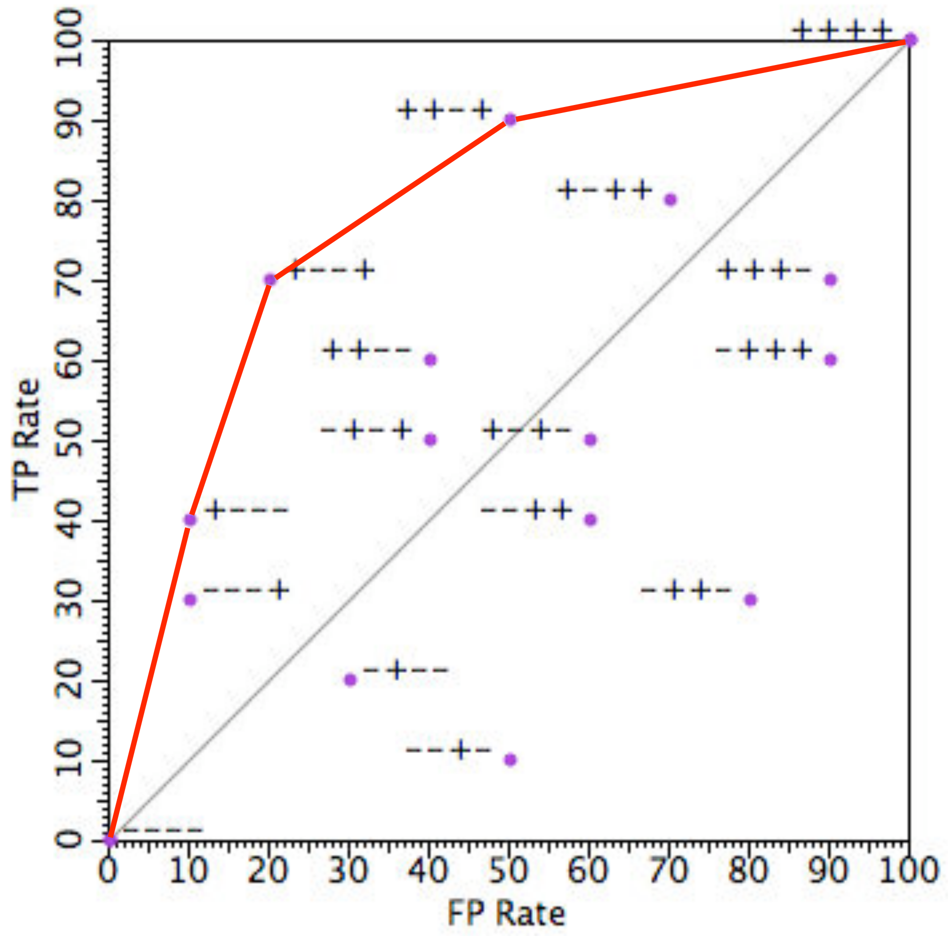
Reordering ROC segments

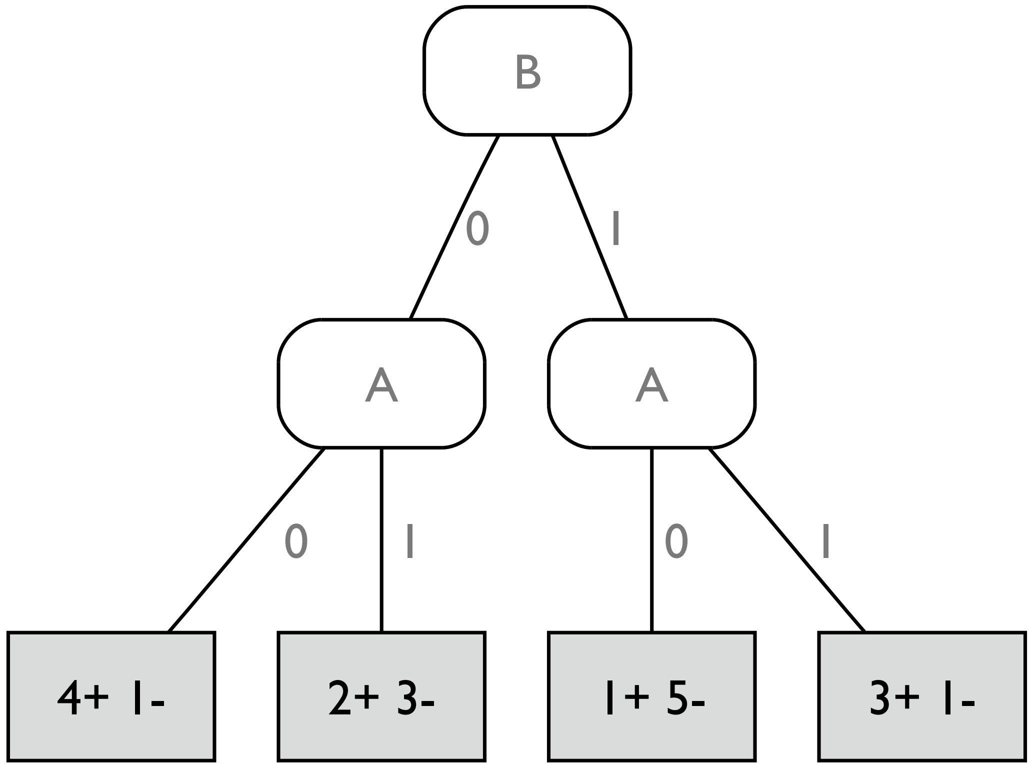




The *joint* probabilities in the leaves are used to re-order the ROC segments, resulting in a convex ROC curve.

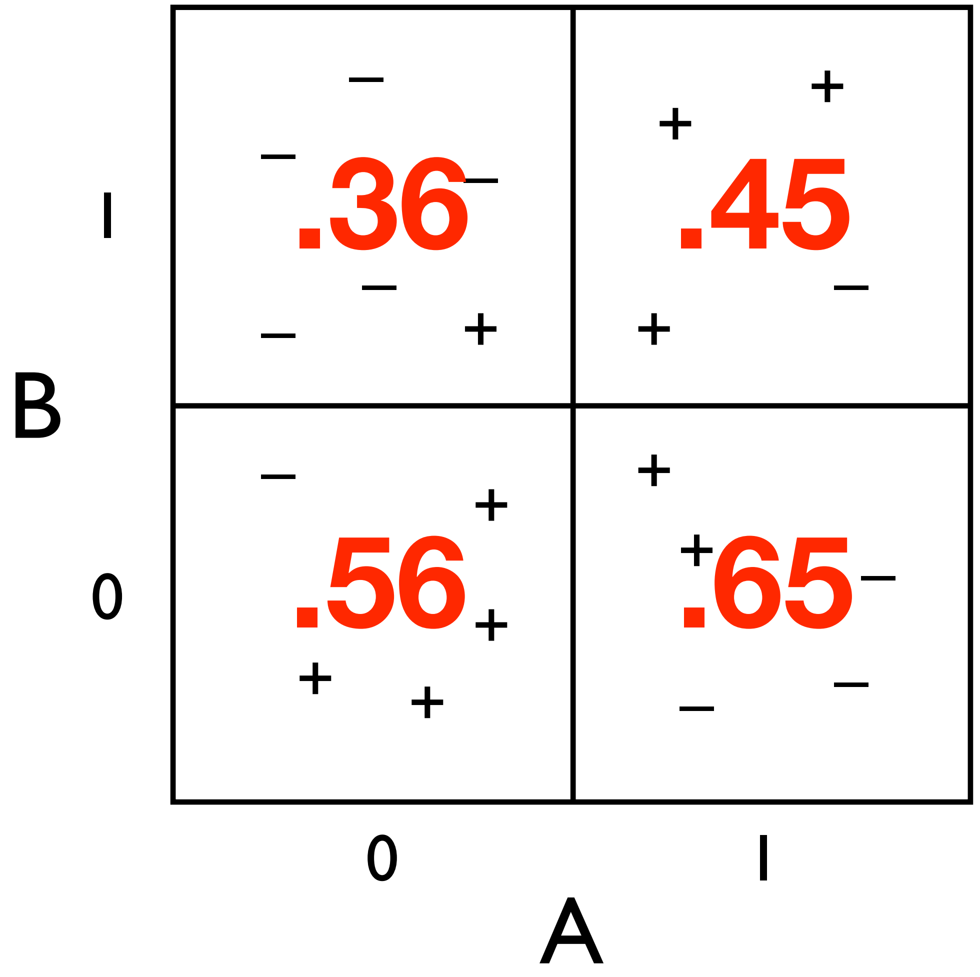
Reordering ROC segments

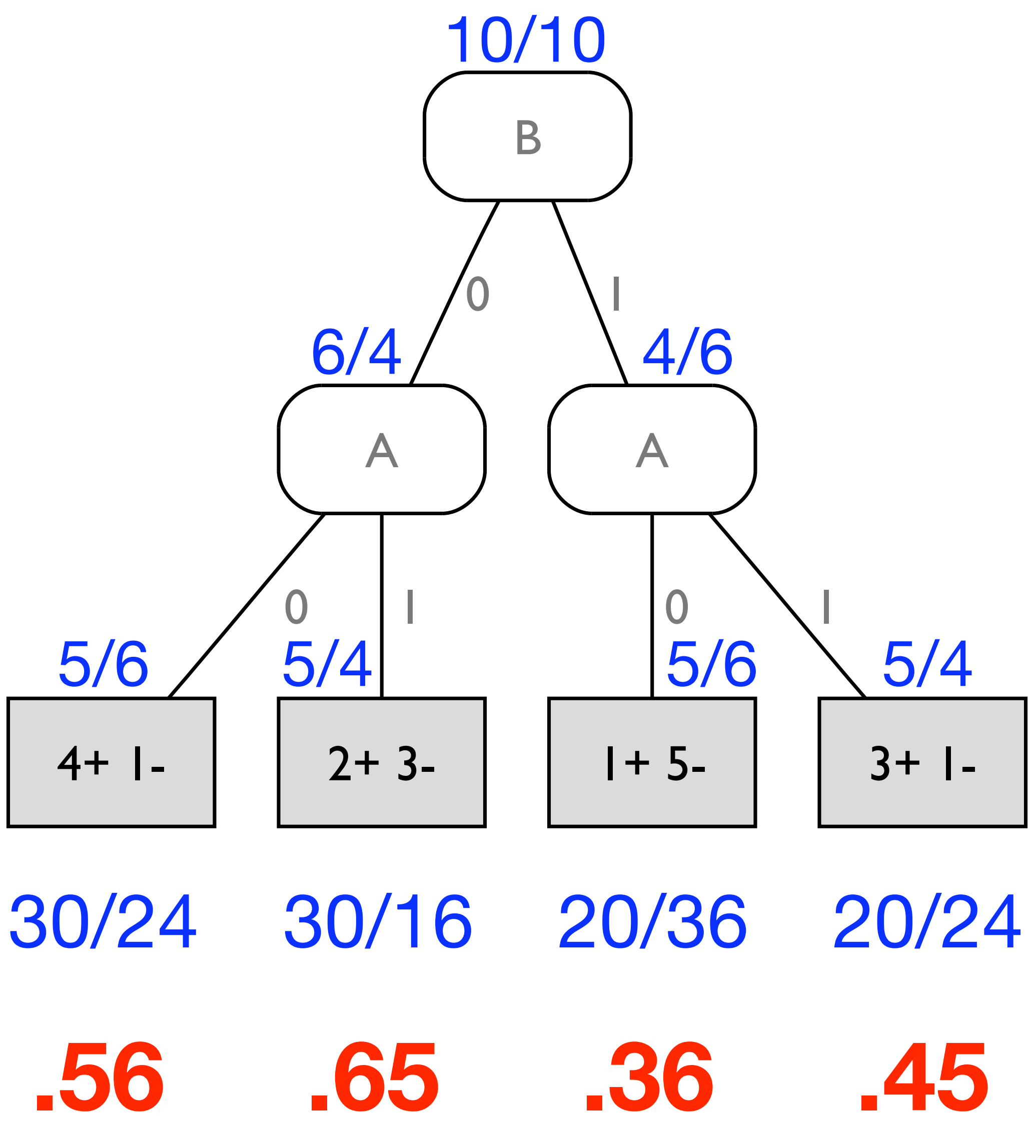


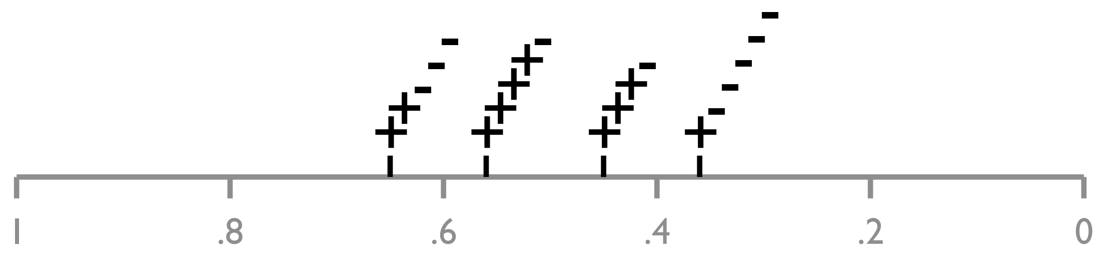


The *joint* probabilities in the leaves are used to re-order the ROC segments, resulting in a convex ROC curve.

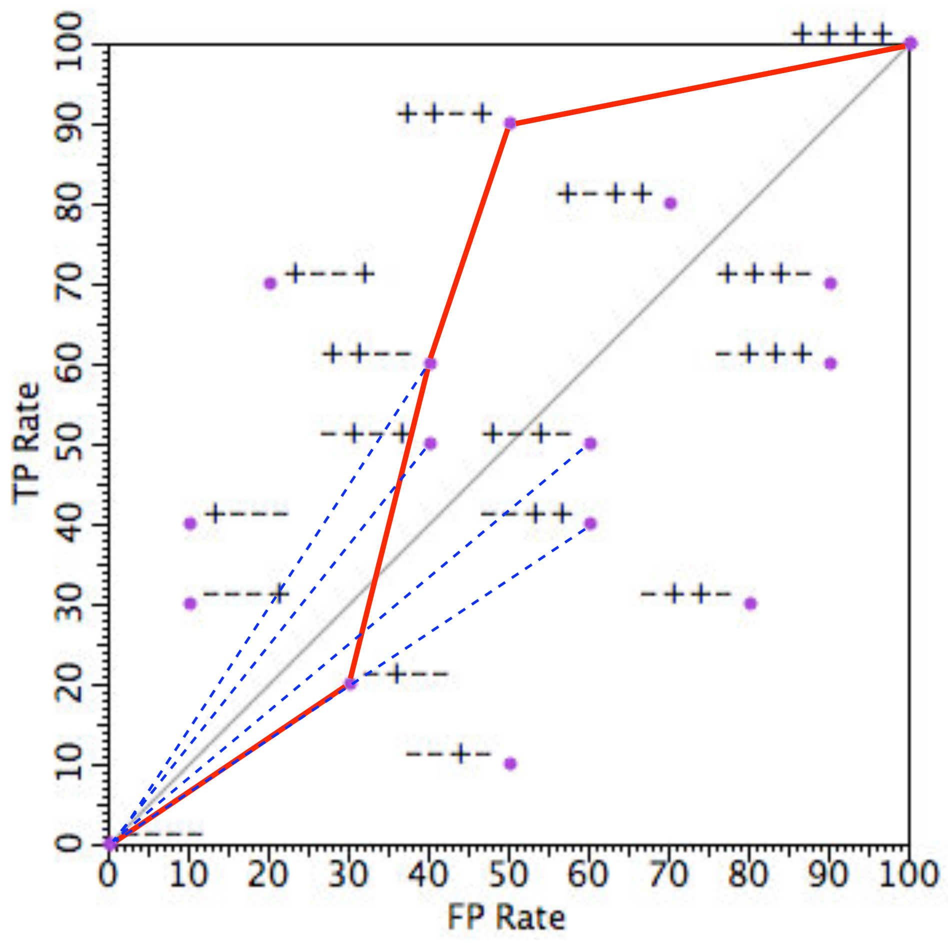
Naive Bayes probability estimator

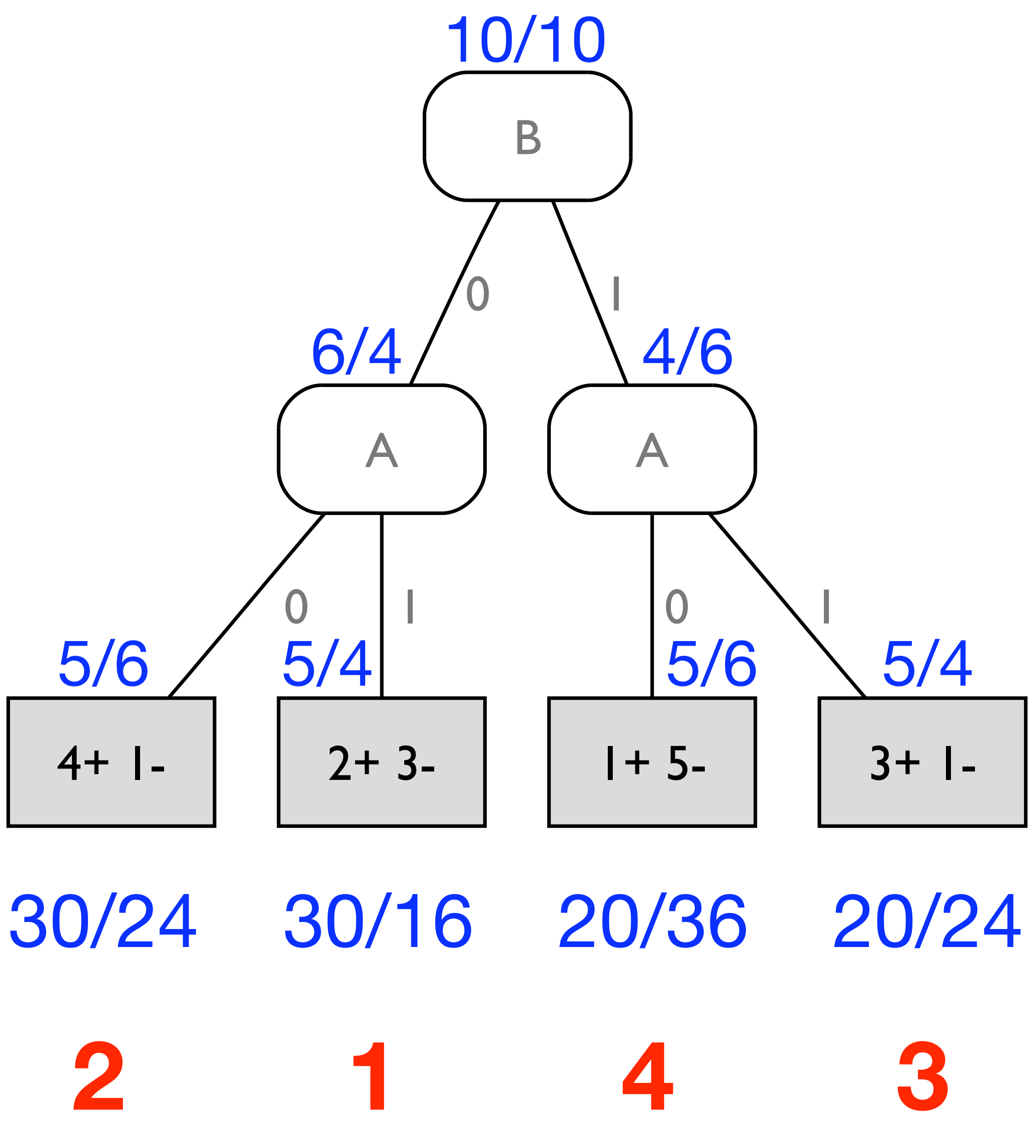






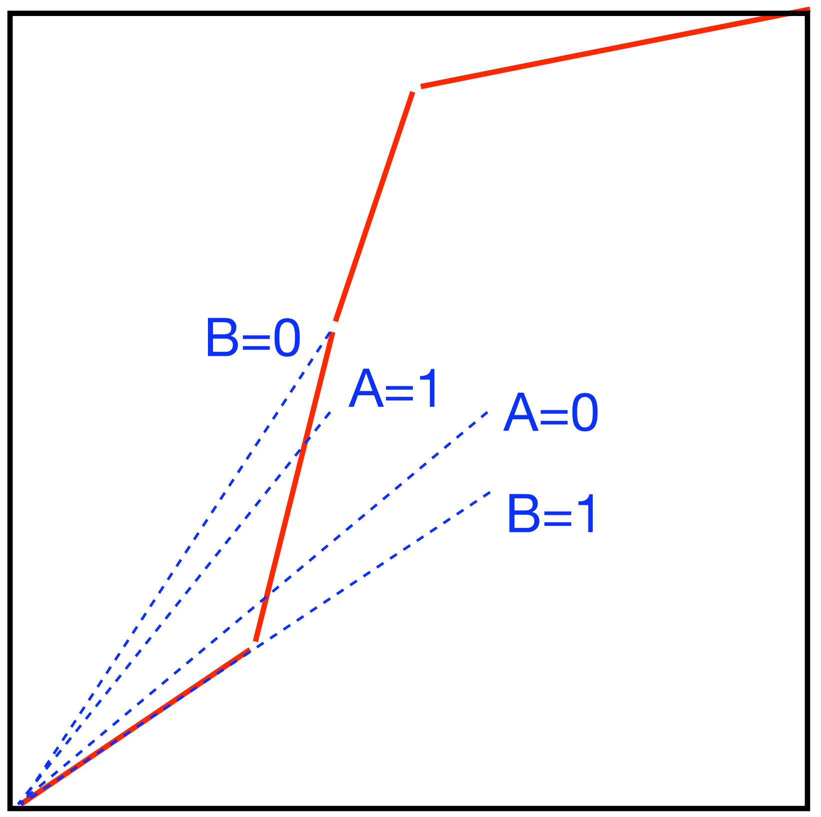
Naive Bayes ROC curve

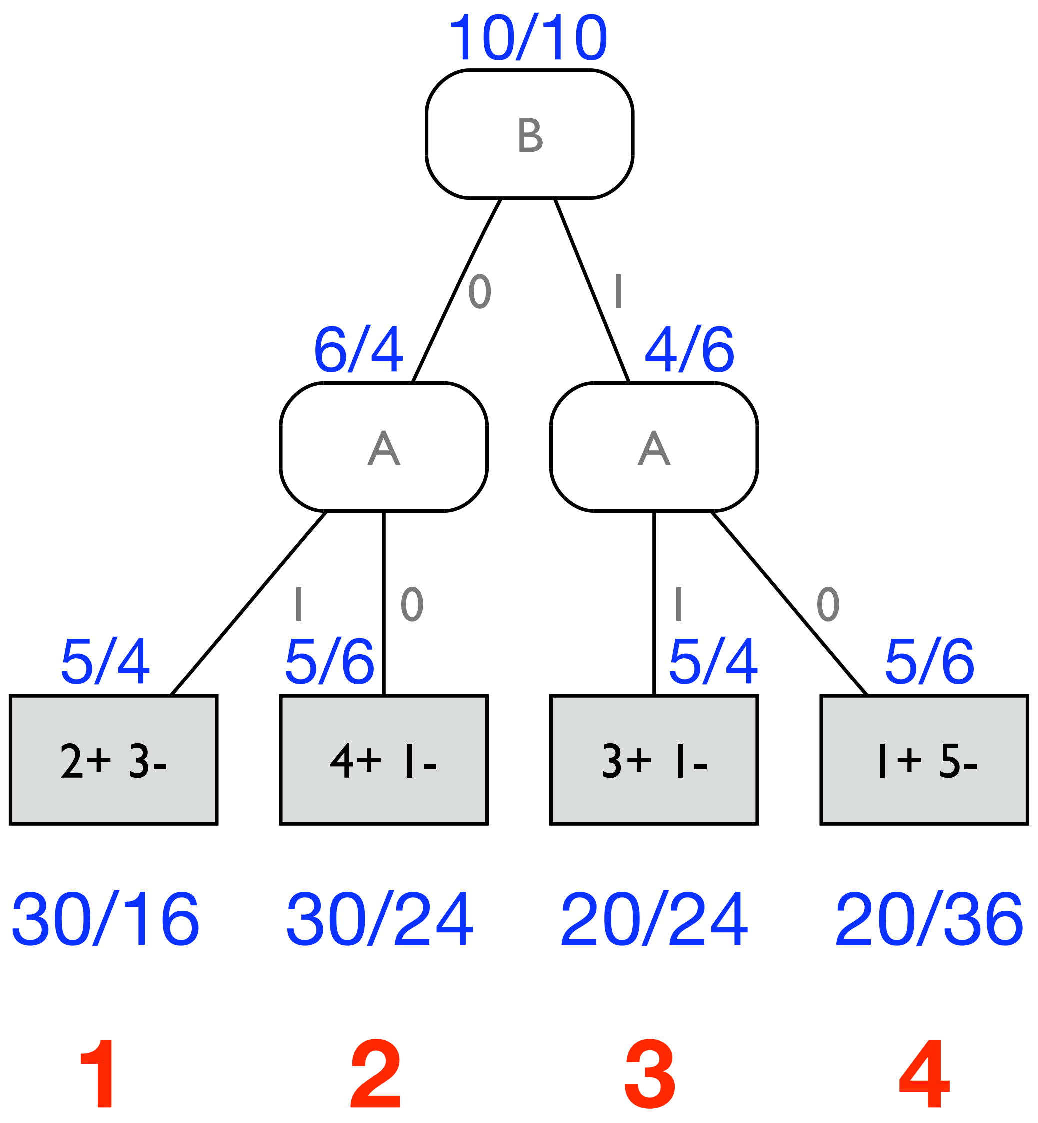




The concavity is caused by misleading marginal probabilities (*cf*. A=1, B=0). Repairing this would require access to the true joint probabilities.

Lexicographic ranking

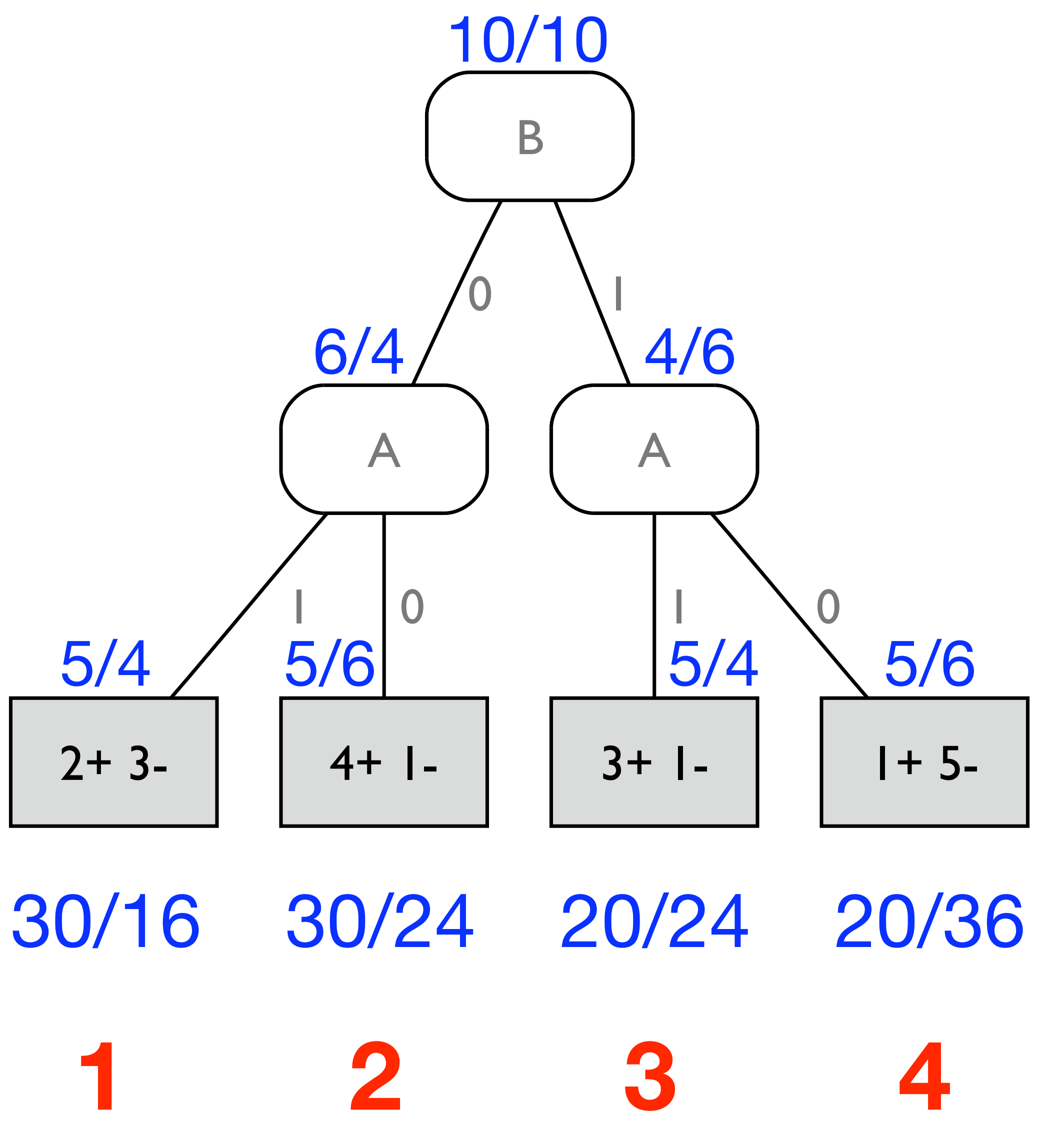




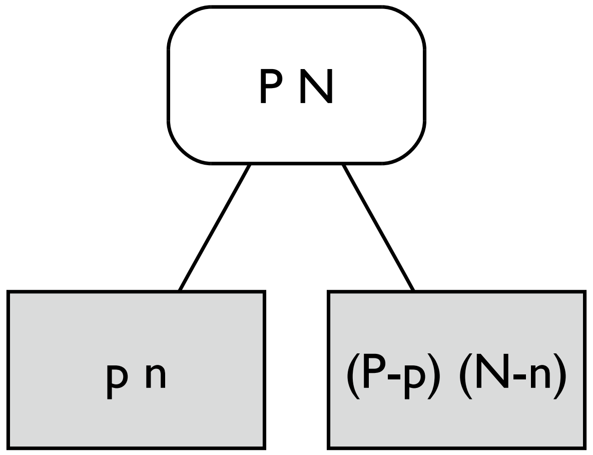
The ranking corresponds to a simple left-to-right ordering of the leaves of the tree that results from putting B before A, B=0 before B=1, and A=1 before A=0.

Lexicographic ranking (2)

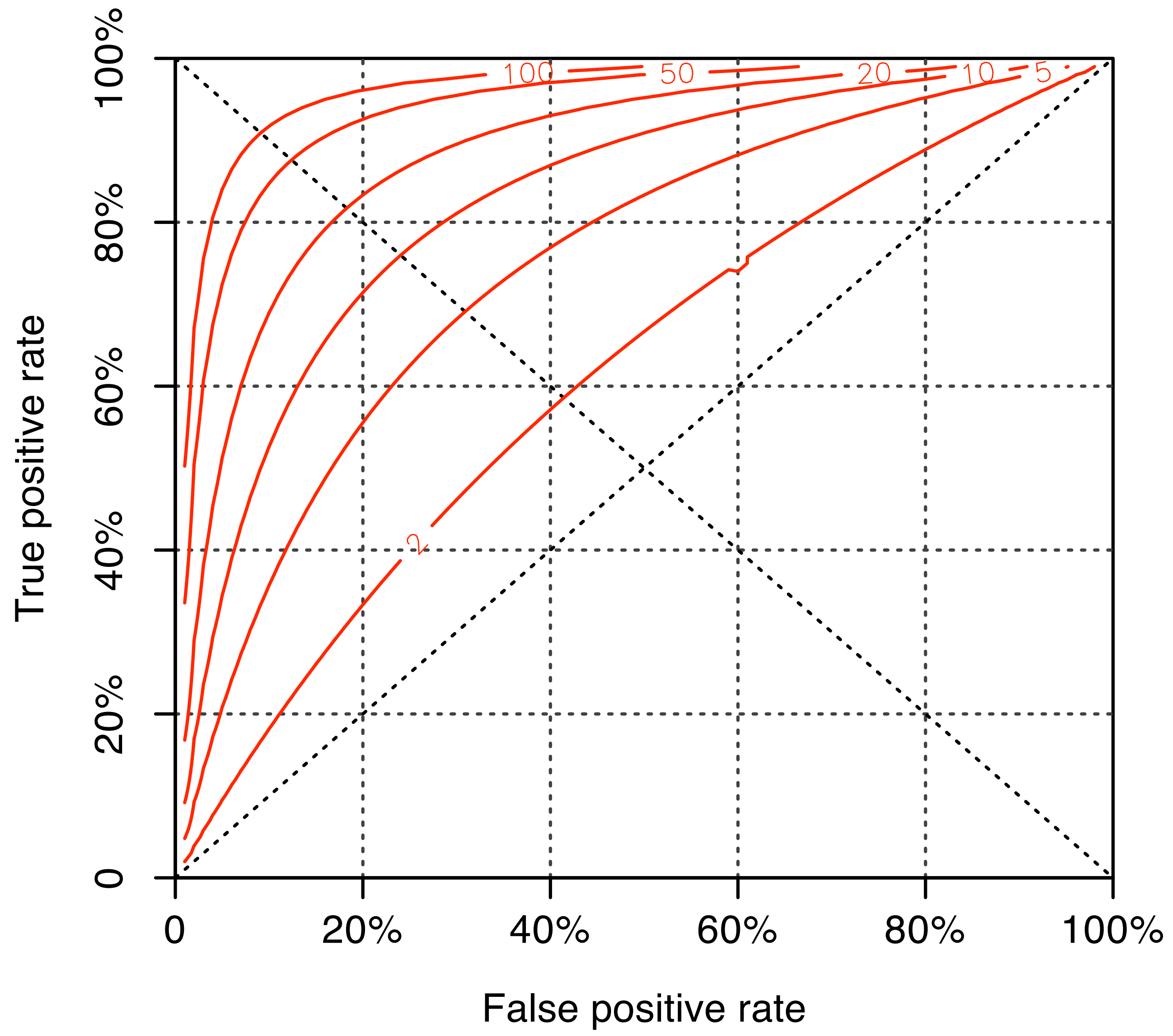
- To rank two instances
  - find the first attribute (top-down) in which the instances differ
  - the instance which has the left attribute value is ranked before the other instance
  - (in practice, trees are not used since they require exponential space; see Flach & Matsubara, ECML'07)
- Attributes are ranked by odds ratio



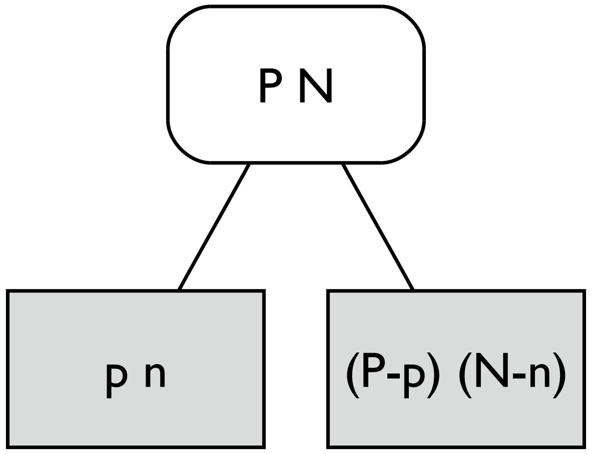
Odds ratio splitting criterion



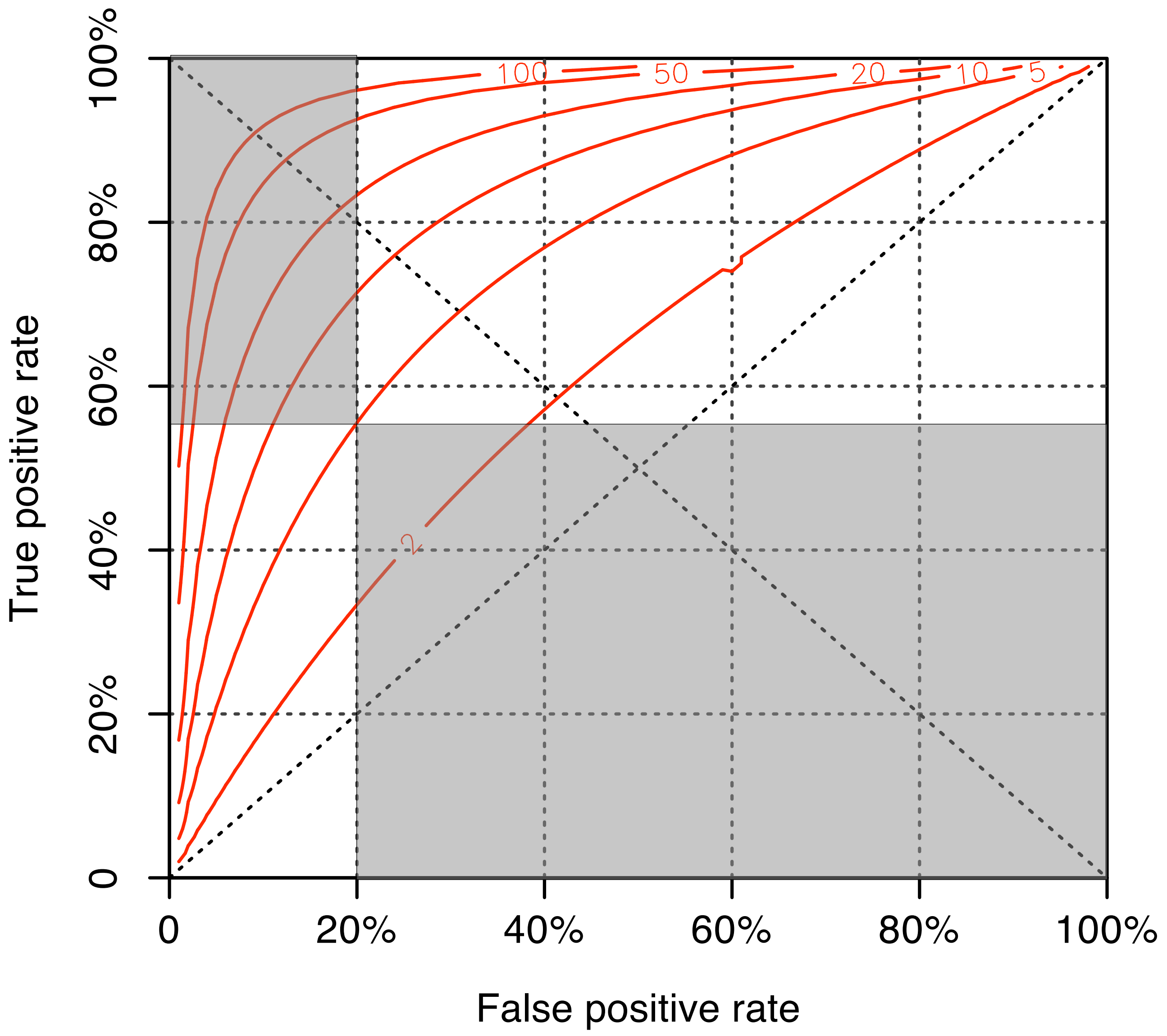
$$OR = \frac{p/n}{(P-p)/(N-n)} = \frac{p(N-n)}{(P-p)n}$$



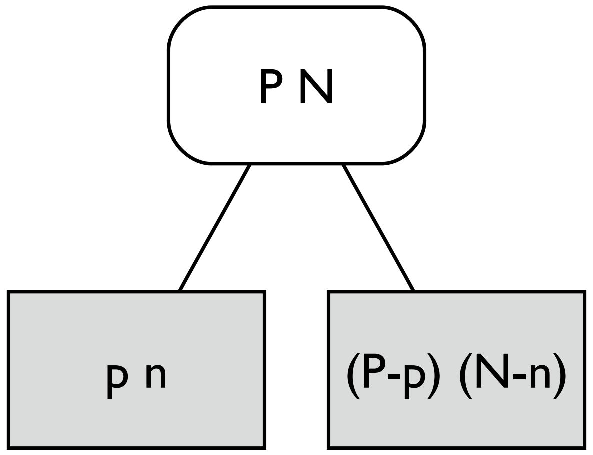
Odds ratio splitting criterion



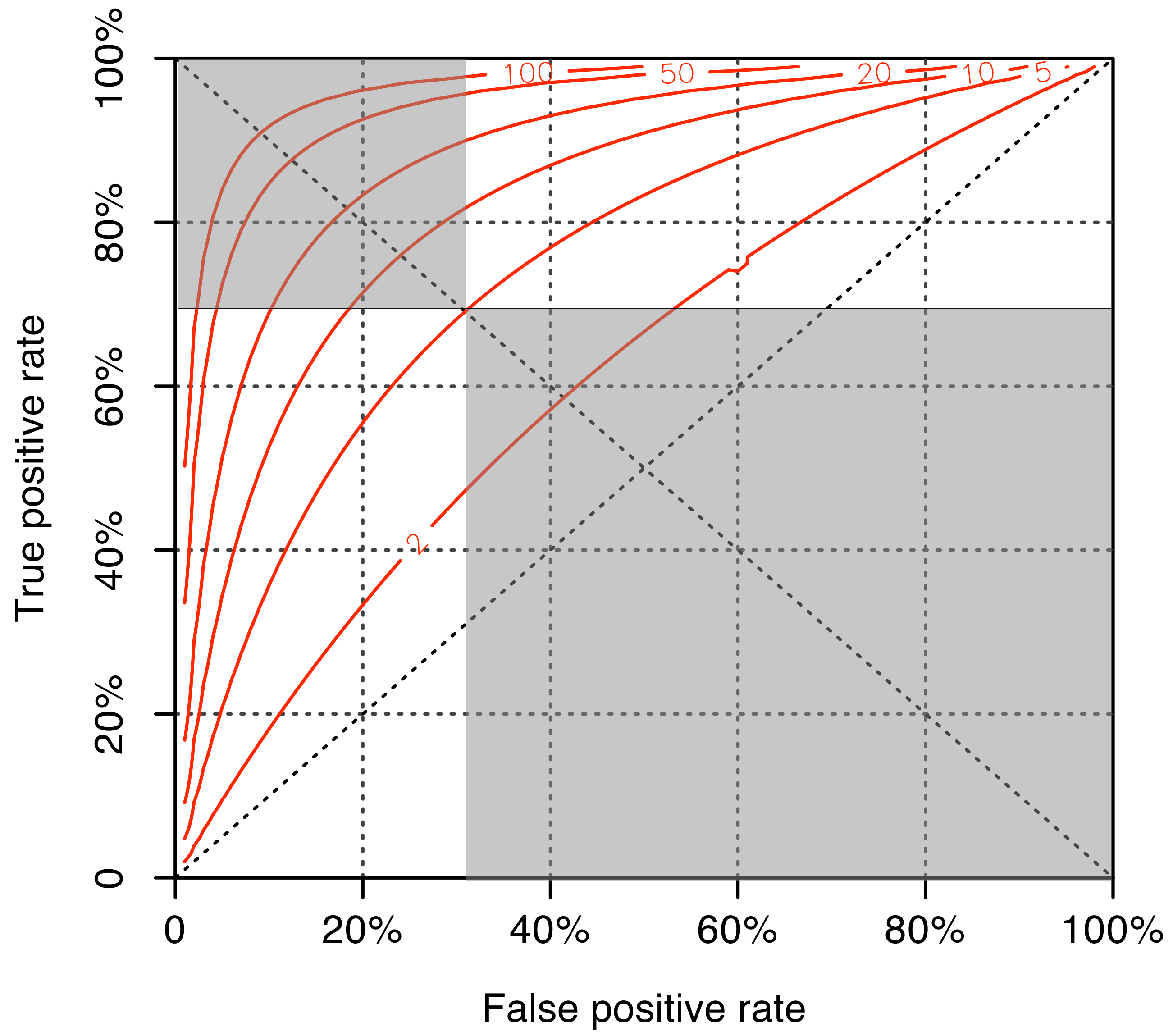
$$OR = \frac{p/n}{(P-p)/(N-n)} = \frac{p(N-n)}{(P-p)n}$$



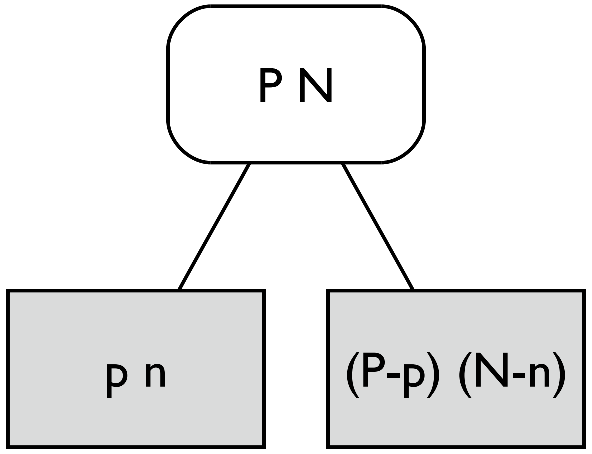
Odds ratio splitting criterion



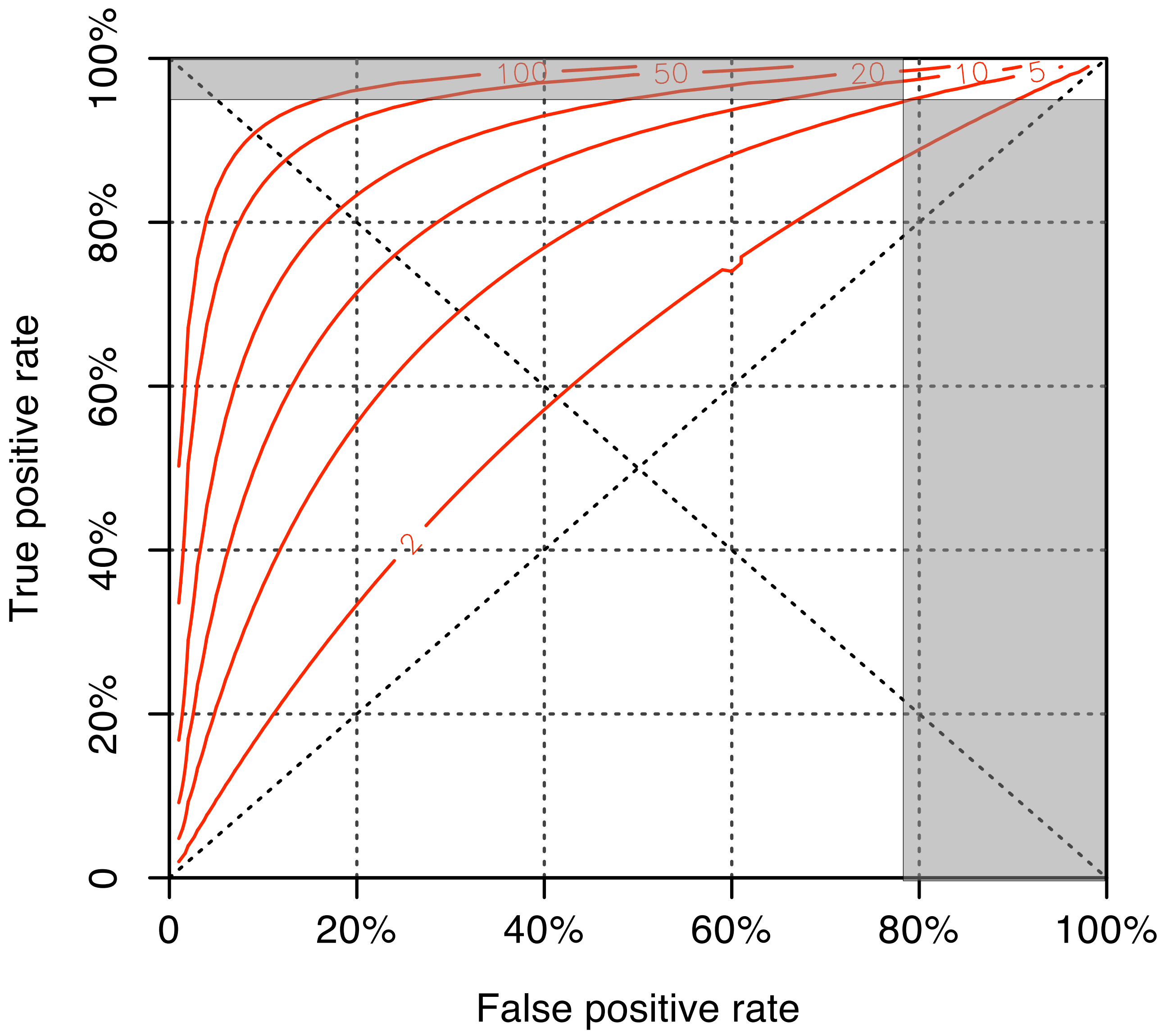
$$OR = \frac{p/n}{(P-p)/(N-n)}$$
$$= \frac{p(N-n)}{(P-p)n}$$



Odds ratio splitting criterion

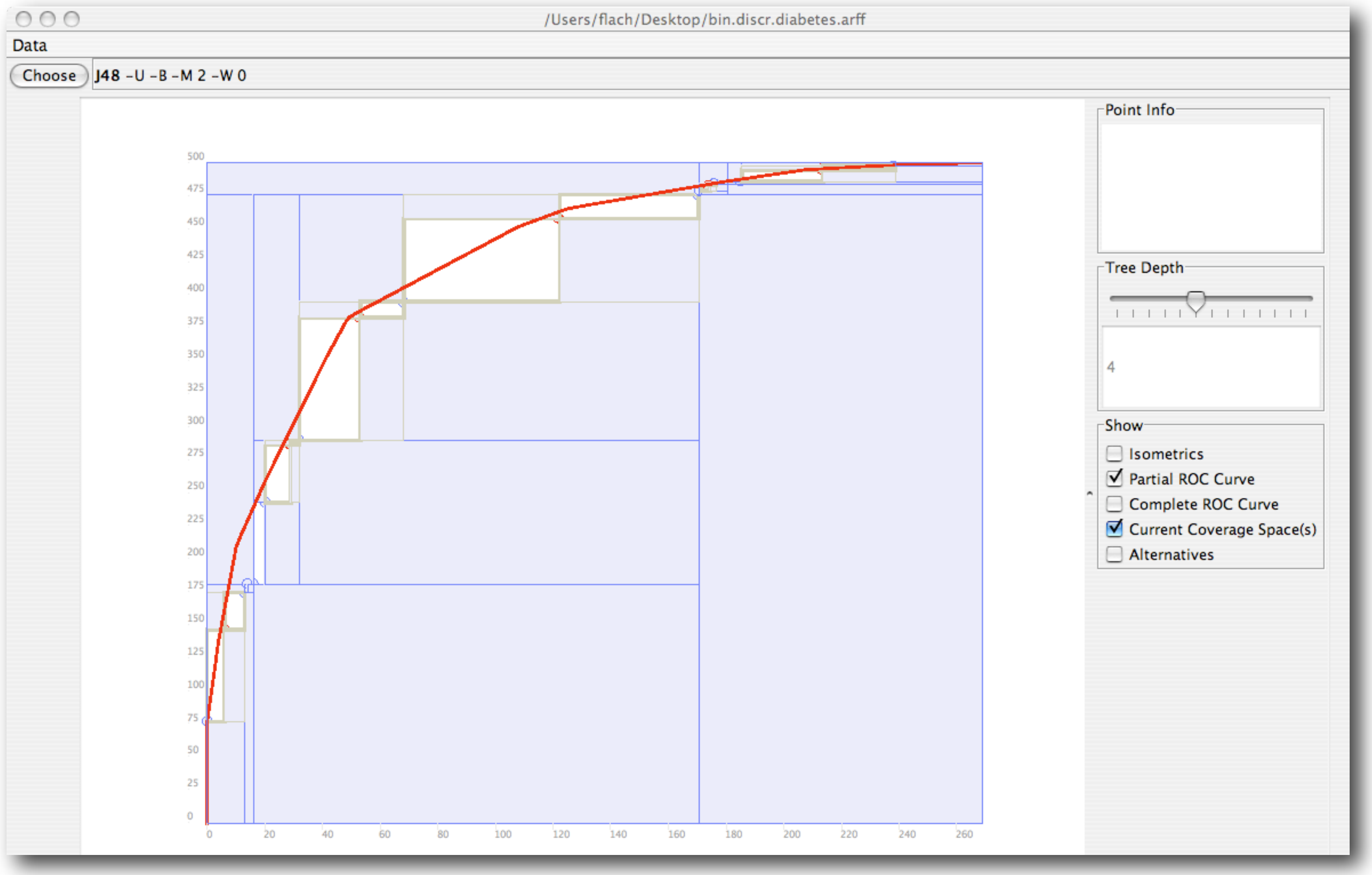


$$OR = \frac{p/n}{(P-p)/(N-n)} = \frac{p(N-n)}{(P-p)n}$$



Cue: ProgRoc

- ProgRoc is a program to visualise tree-based models in ROC space
  - http://www.cs.bris.ac.uk/Research/MachineLearning/ProgRoc/
  - training data in ARFF format, model trained by Weka
  - we can cut off the tree at each depth to show the recursive partitioning
  - in addition, we can see the partial ROC curve obtained from the ranking
  - many thanks to Tarek Abudawood and Edson Takashi Matsubara



/Users/flach/Desktop/bin.discr.diabetes.arff

Data

Choose J48 -U -B -M 2 -W 0

Point Info

Tree Depth

4

Show

- [ ] Isometrics
- [x] Partial ROC Curve
- [ ] Complete ROC Curve
- [x] Current Coverage Space(s)
- [ ] Alternatives

Building models — summary

- Decision tree (training set) ROC curve is always convex *because the tree has access to the joint probabilities in its leaves*

- At the other extreme, lexicographic ranking is purely syntactic in that it strictly follows the recursive structure of the constructed tree

- Naive Bayes estimates the joint probabilities from the marginals which gives it somewhat more flexibility than lexicographic ranking

  - e.g., 000 > 001 > 010 > 100 > 011 > 101 > 110 > 111 is not lexicographic but achievable by naive Bayes

II Classification and ranking

- Classification performance is measured by accuracy, ranking performance is measured by area under the ROC curve (AUC, see further)
  - these clearly measure different things: for n examples, accuracy is measured in O(n) steps and AUC in O(n log n) steps
- On the other hand, we often encounter statements such as “AUC aggregates the model’s behaviour for all possible decision thresholds”
- It is not entirely clear what this actually means —
  - is AUC some kind of expected value of accuracy?
  - if there is such a linear relation, does it follow that optimising accuracy and optimising AUC lead to the same model?

Some notation

	Predicted		*Total*
Actual	TP	FN	Pos
	FP	TN	Neg

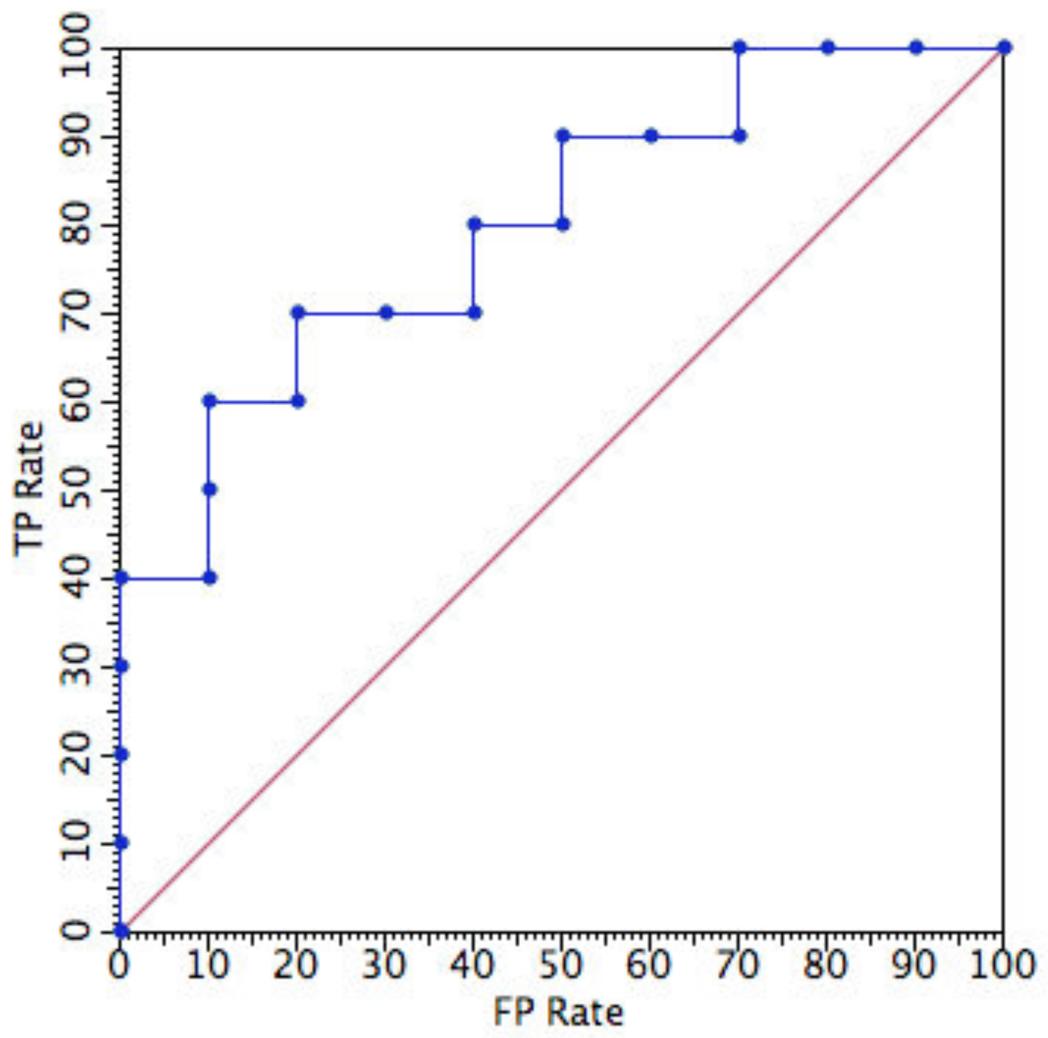
- Pos: number of positives
- Neg: number of negatives
- Class ratio c = Pos/Neg
- TP, FP, TN, FN: number of true/false positives/negatives
- Acc: number of correctly classified examples (Acc = TP+TN)
- Err: number of incorrectly classified examples (Err = FP+FN)

- pos: proportion of positives (pos = Pos/(Pos+Neg)
- neg: proportion of negatives (neg = 1–pos)
- c = pos/neg
- tpr, fpr, tnr, fnr: true/false positive/negative rates (tpr = TP/Pos, fpr = FP/Neg, tnr = 1–fpr, fnr = 1–tpr)
- acc: proportion of correctly classified examples (acc = pos*tpr + neg*tnr)
- err = 1–acc

From a ranking to a ROC curve

+ + + + - + + - + - - + - + - - + - - -

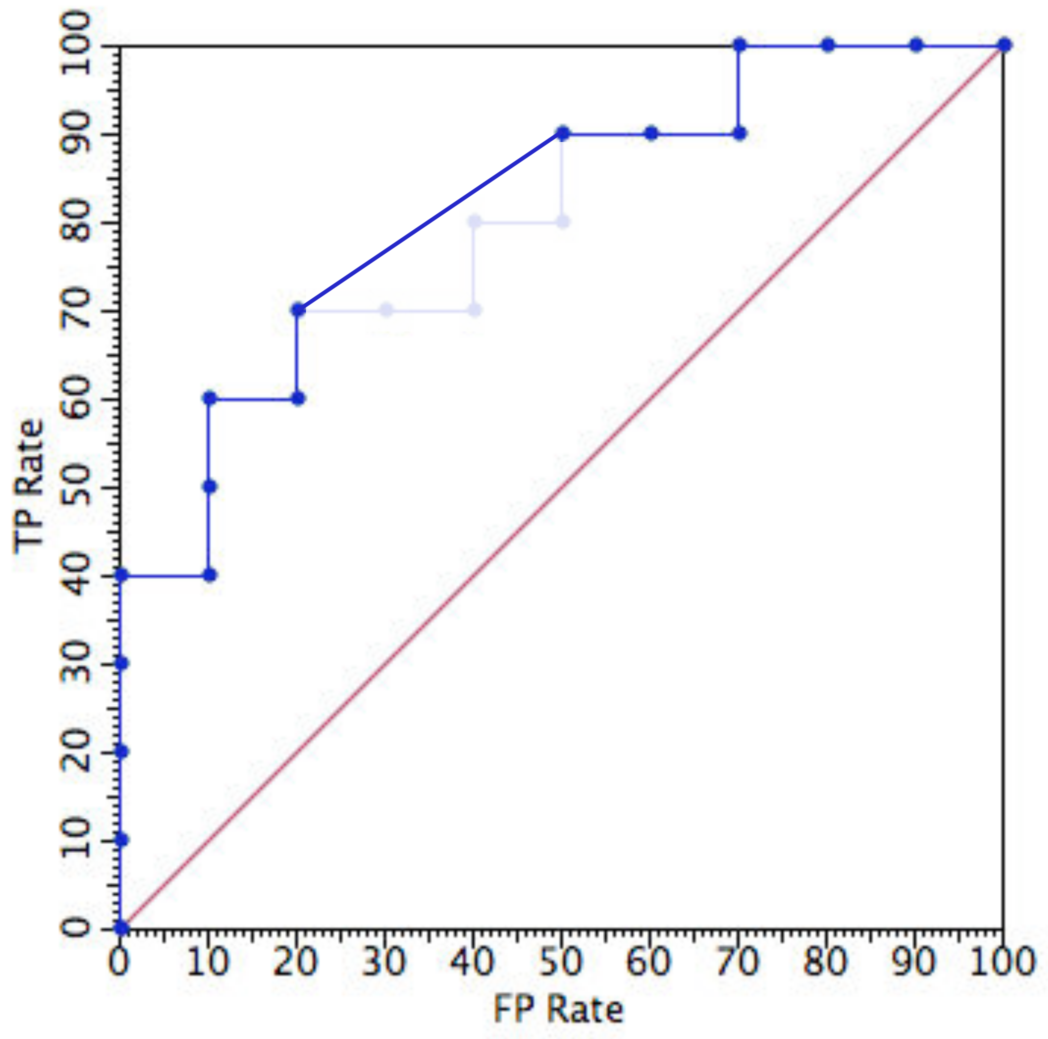
- start in (0,0)
- get the next instance in the ranking
    - if it is positive, move 1/Pos up
    - if it is negative, move 1/Neg right



From a ranking to a ROC curve

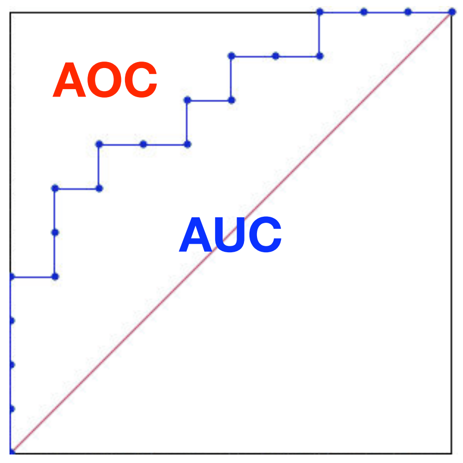
+ + + + - + + - +   - - + - +   - - + - - -

- start in (0,0)
- get the next instance in the ranking
  - if it is positive, move 1/Pos up
  - if it is negative. move 1/Neg right
  - make diagonal move in case of ties



Machine Learning 101 Exam, Q42. AUC is ...

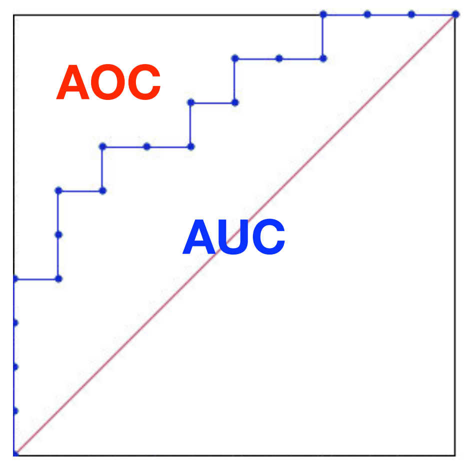
+ + + + - + + - + - - + - + - - + - - -



Machine Learning 101 Exam, Q42. AUC is ...

(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

+ + + + - + + - + - - + - + - - + - - -

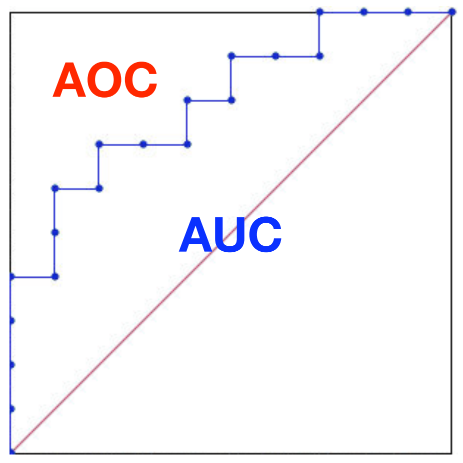


Machine Learning 101 Exam, Q42. AUC is ...

(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

(b) The expected proportion of positives ranked before a uniformly drawn random negative.

+ + + + - + + - + - - + - + - - + - - -



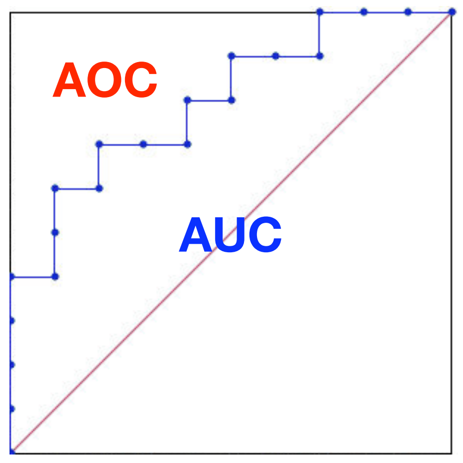
Machine Learning 101 Exam, Q42. AUC is ...

(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

(b) The expected proportion of positives ranked before a uniformly drawn random negative.

(c) The expected true positive rate if the ranking is split just before a uniformly drawn random negative.

+ + + + - + + - + - - + - + - - + - - -



Machine Learning 101 Exam, Q42. AUC is ...

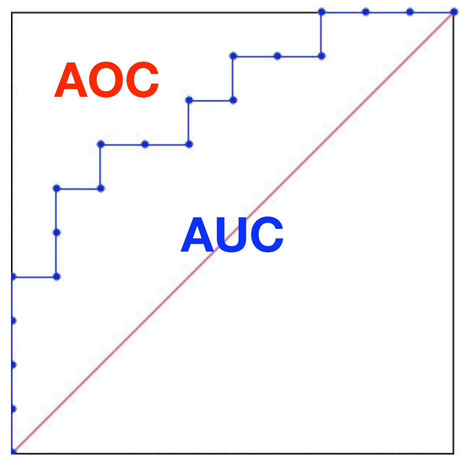
(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

(b) The expected proportion of positives ranked before a uniformly drawn random negative.

(c) The expected true positive rate if the ranking is split just before a uniformly drawn random negative.

(d) The expected proportion of negatives ranked after a uniformly drawn random positive.

+ + + + - + + - + - - + - + - - + - - -



Machine Learning 101 Exam, Q42. AUC is ...

(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

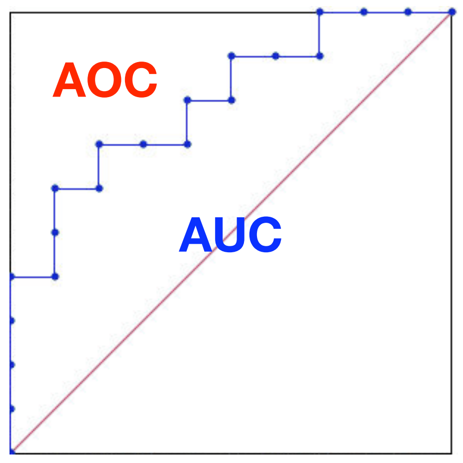
(b) The expected proportion of positives ranked before a uniformly drawn random negative.

(c) The expected true positive rate if the ranking is split just before a uniformly drawn random negative.

(d) The expected proportion of negatives ranked after a uniformly drawn random positive.

(e) 1 – the expected false positive rate if the ranking is split just after a uniformly drawn random positive.

+ + + + - + + - + - - + - + - - + - - -



Machine Learning 101 Exam, Q42. AUC is ...

(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

(b) The expected proportion of positives ranked before a uniformly drawn random negative.

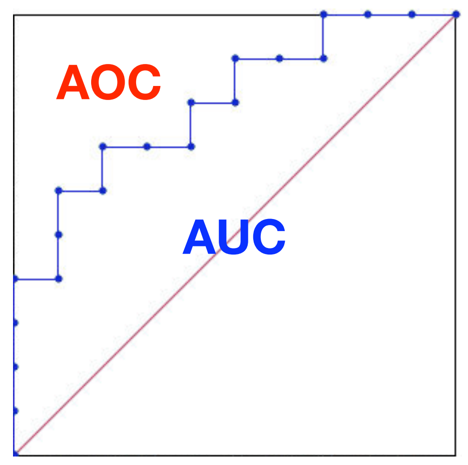
(c) The expected true positive rate if the ranking is split just before a uniformly drawn random negative.

(d) The expected proportion of negatives ranked after a uniformly drawn random positive.

(e) 1 – the expected false positive rate if the ranking is split just after a uniformly drawn random positive.

(f) All of the above.

+ + + + - + + - + - - + - + - - + - - -



Machine Learning 101 Exam, Q42. AUC is ...

(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

(b) The expected proportion of positives ranked before a uniformly drawn random negative.

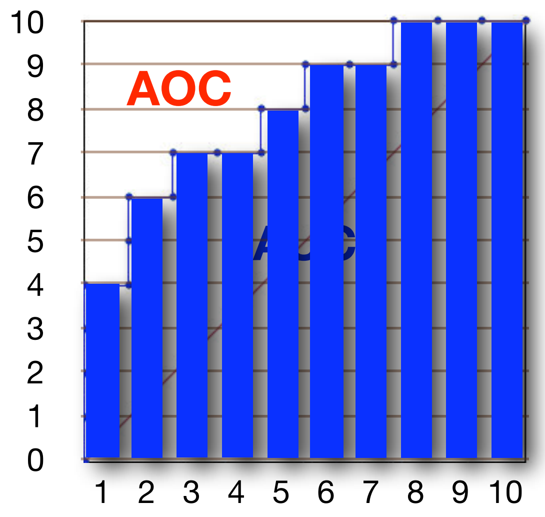
(c) The expected true positive rate if the ranking is split just before a uniformly drawn random negative.

(d) The expected proportion of negatives ranked after a uniformly drawn random positive.

(e) 1 – the expected false positive rate if the ranking is split just after a uniformly drawn random positive.

(f) All of the above.

+ + + + - + + - + - - + - + - - + - - -



Machine Learning 101 Exam, Q42. AUC is ...

(a) The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

(b) The expected proportion of positives ranked before a uniformly drawn random negative.

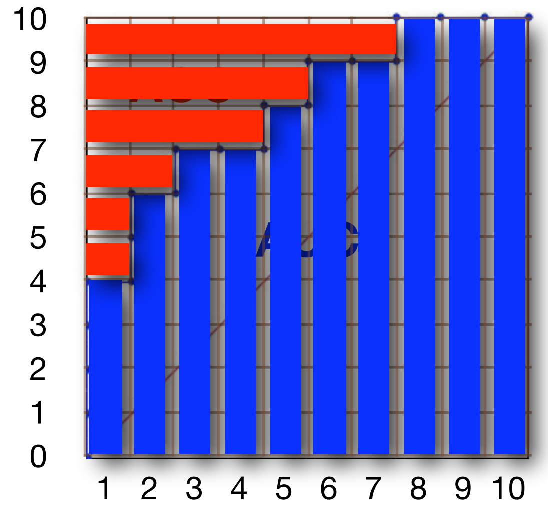
(c) The expected true positive rate if the ranking is split just before a uniformly drawn random negative.

(d) The expected proportion of negatives ranked after a uniformly drawn random positive.

(e) 1 – the expected false positive rate if the ranking is split just after a uniformly drawn random positive.

(f) All of the above.

+ + + + - + + - + - - + - + - - + - - -

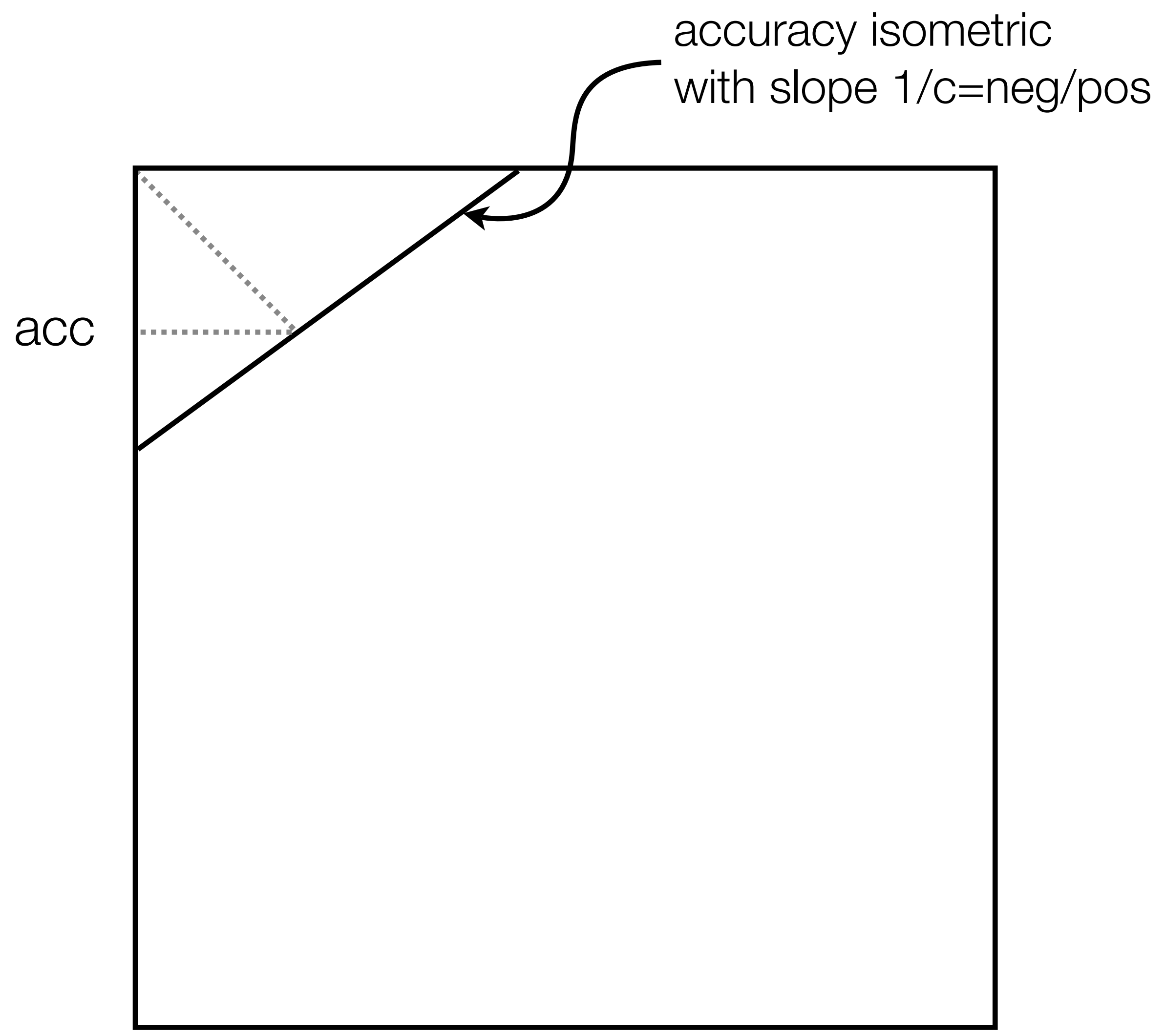


From AUC to accuracy

- Given a ranking, uniformly pick a random split point
  - NB. *Not* uniformly over the range of scores!
- The expected value of acc is
  - $E[acc] = \frac{1}{2}(pos^2 + neg^2) + 2pos \cdot neg \cdot auc$
- For uniform class distributions, this reduces to
  - $E[acc] = \frac{1}{4} + \frac{1}{2}auc$
- But this has high variance (e.g., we can always achieve majority class)

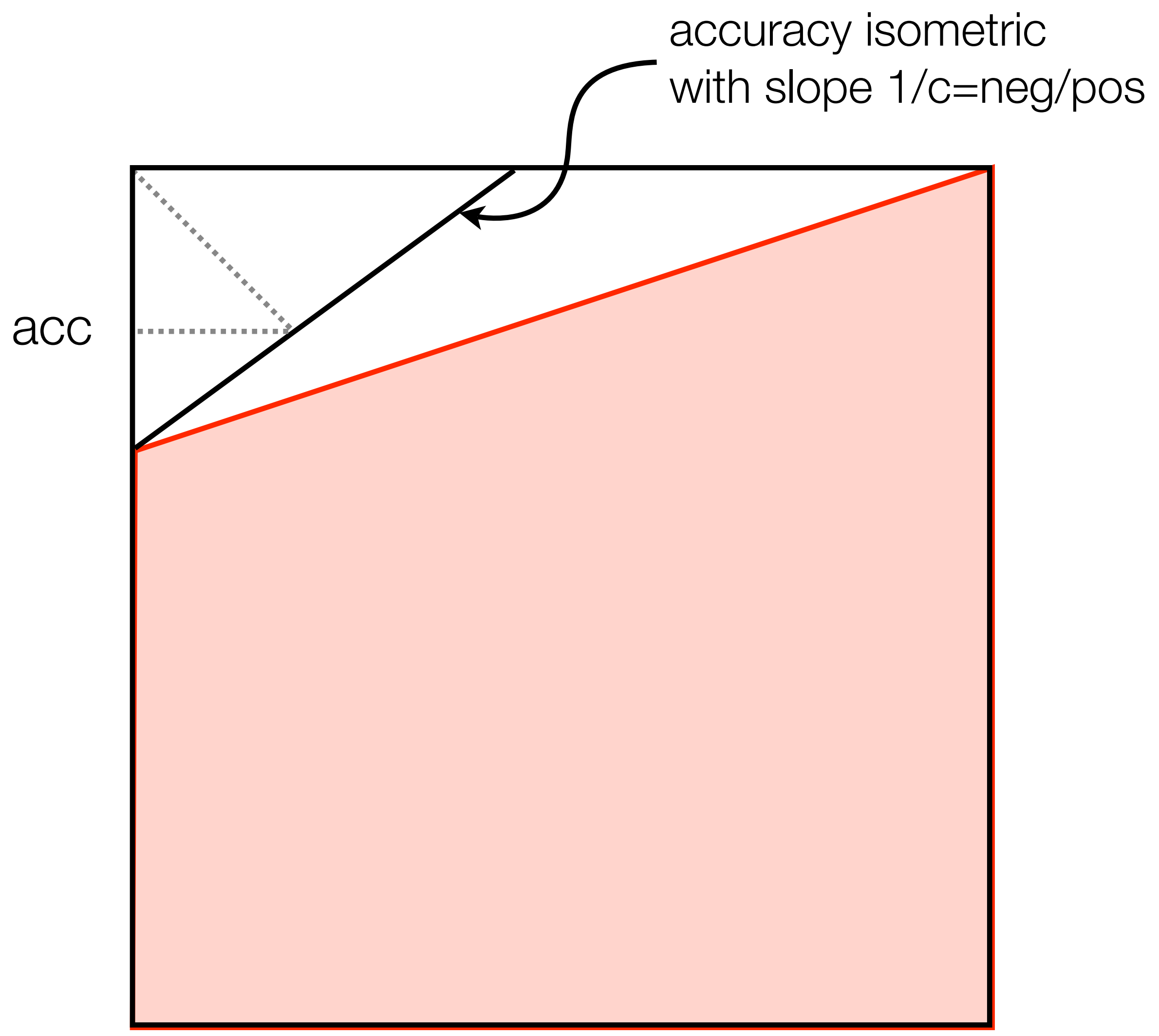
From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve



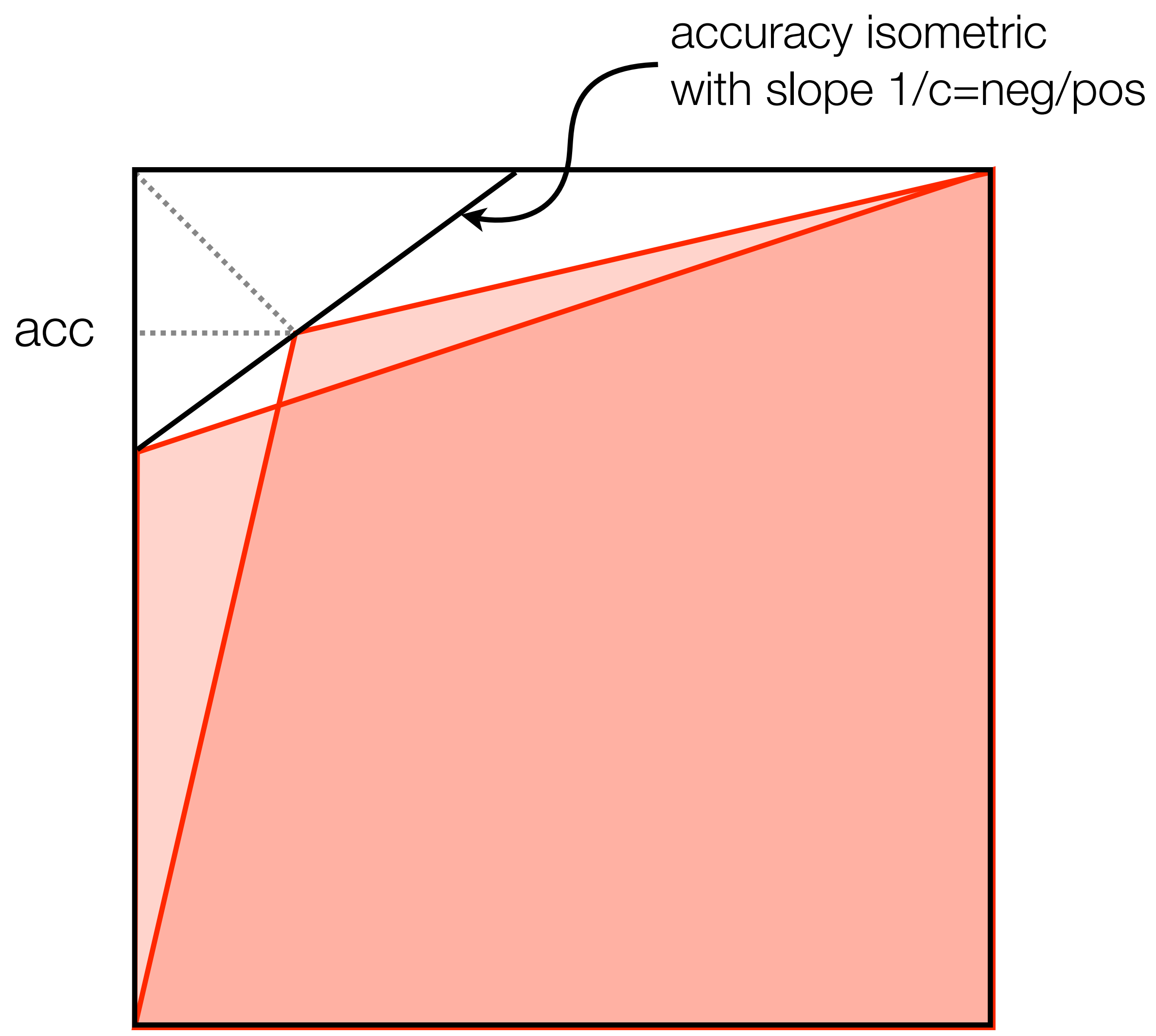
From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve



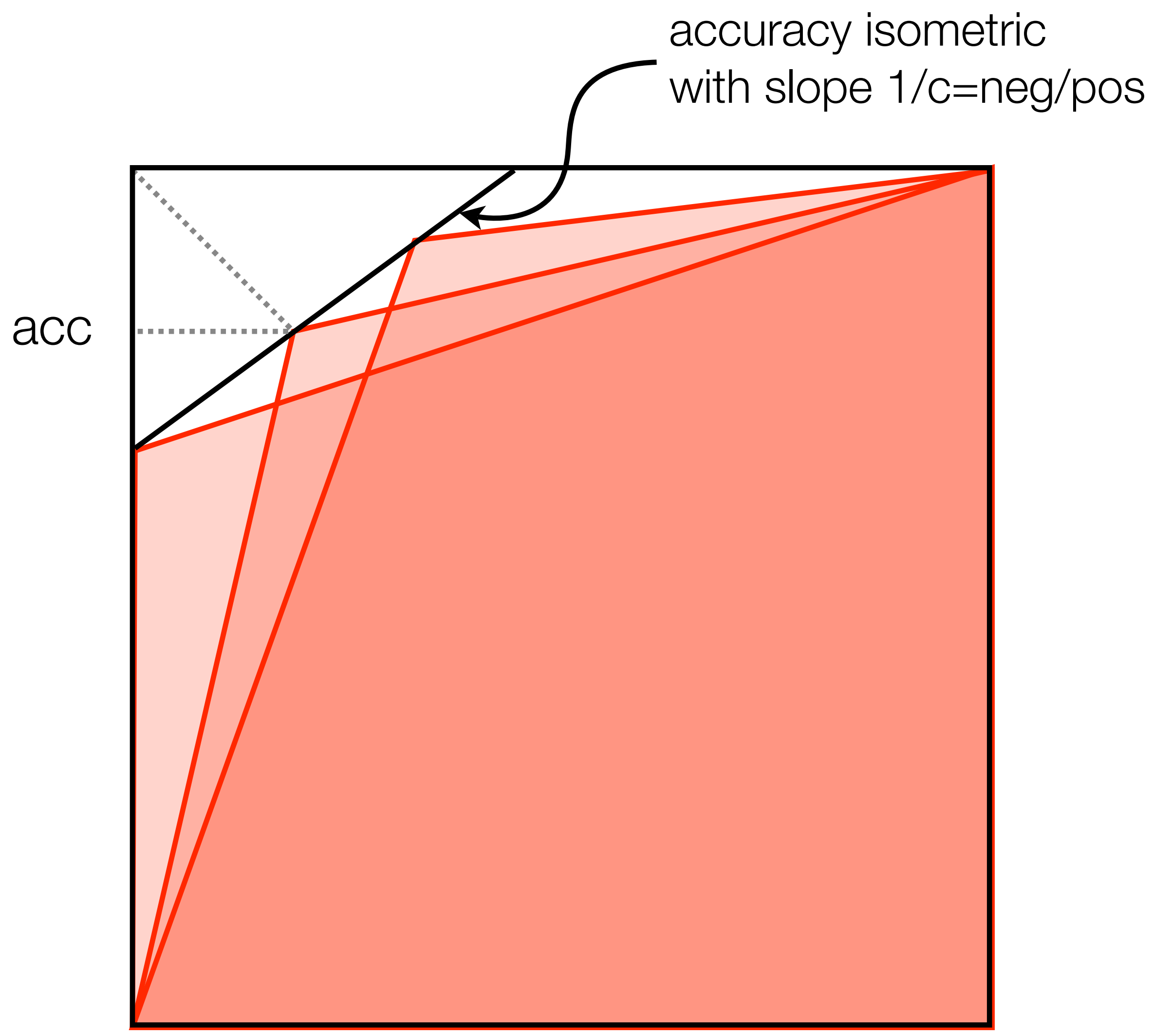
From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve



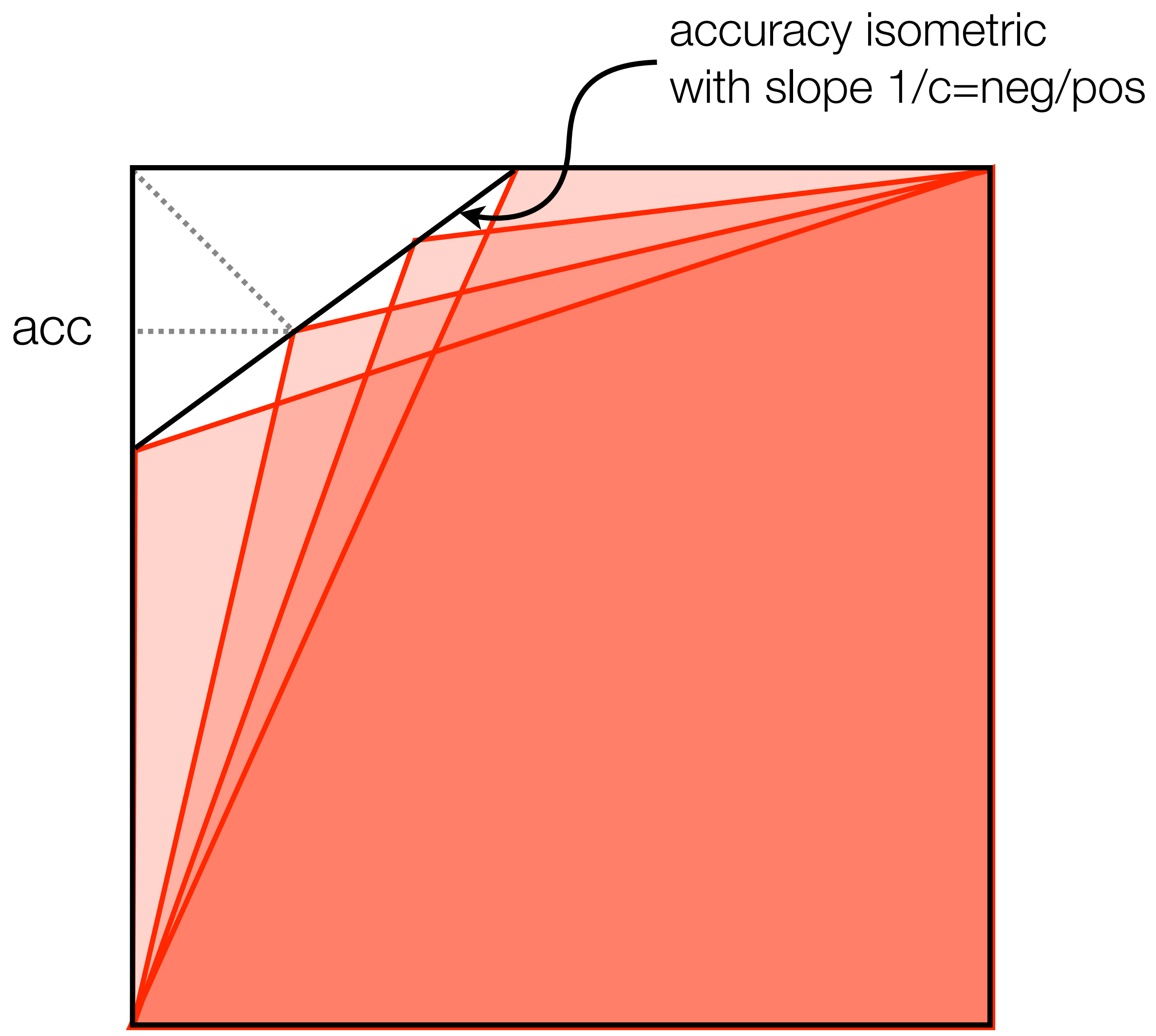
From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve



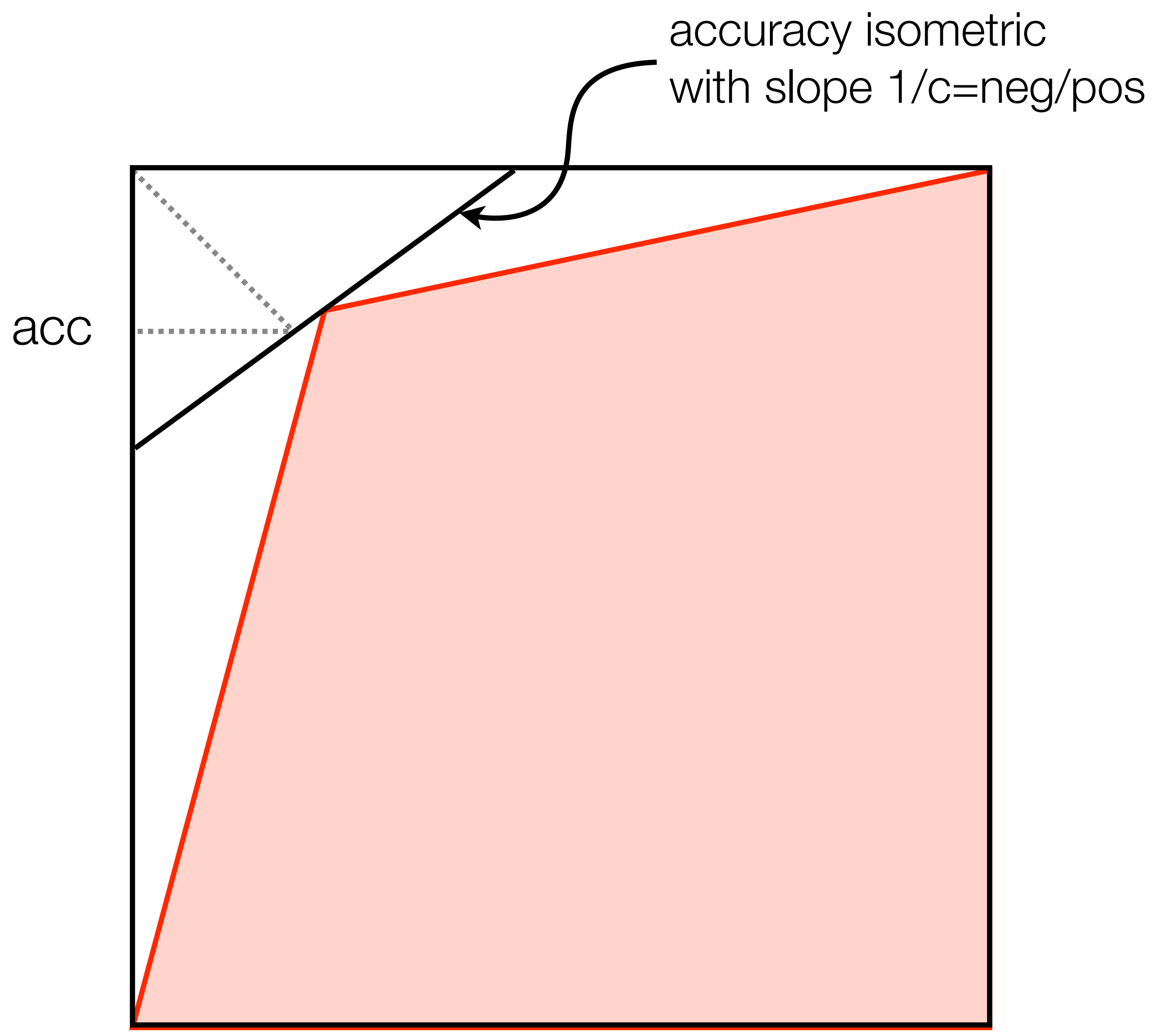
From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve



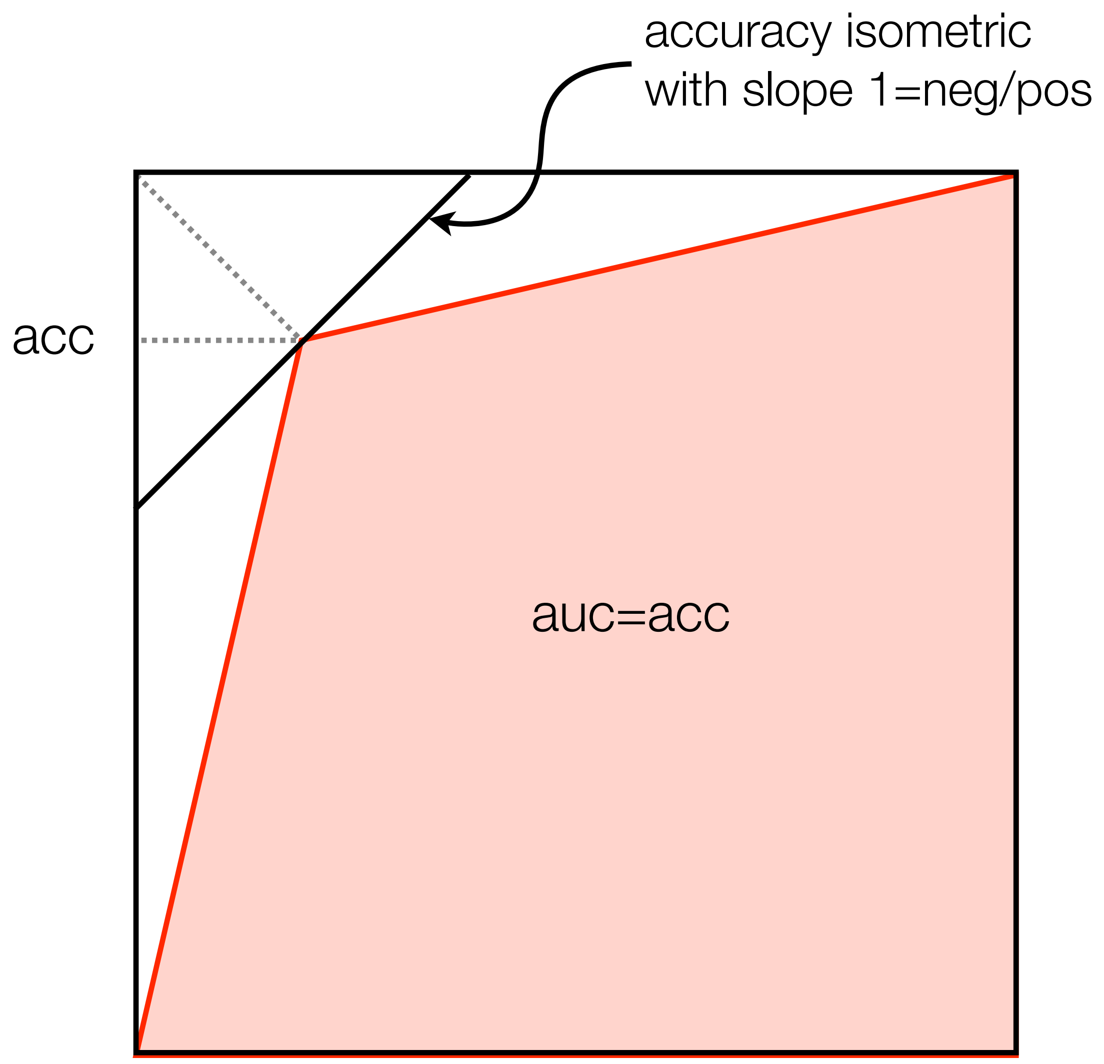
From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve
- The expected value of AUC is
  - $E[auc] = \frac{acc-(pos-neg)^2}{4pos \cdot neg}$



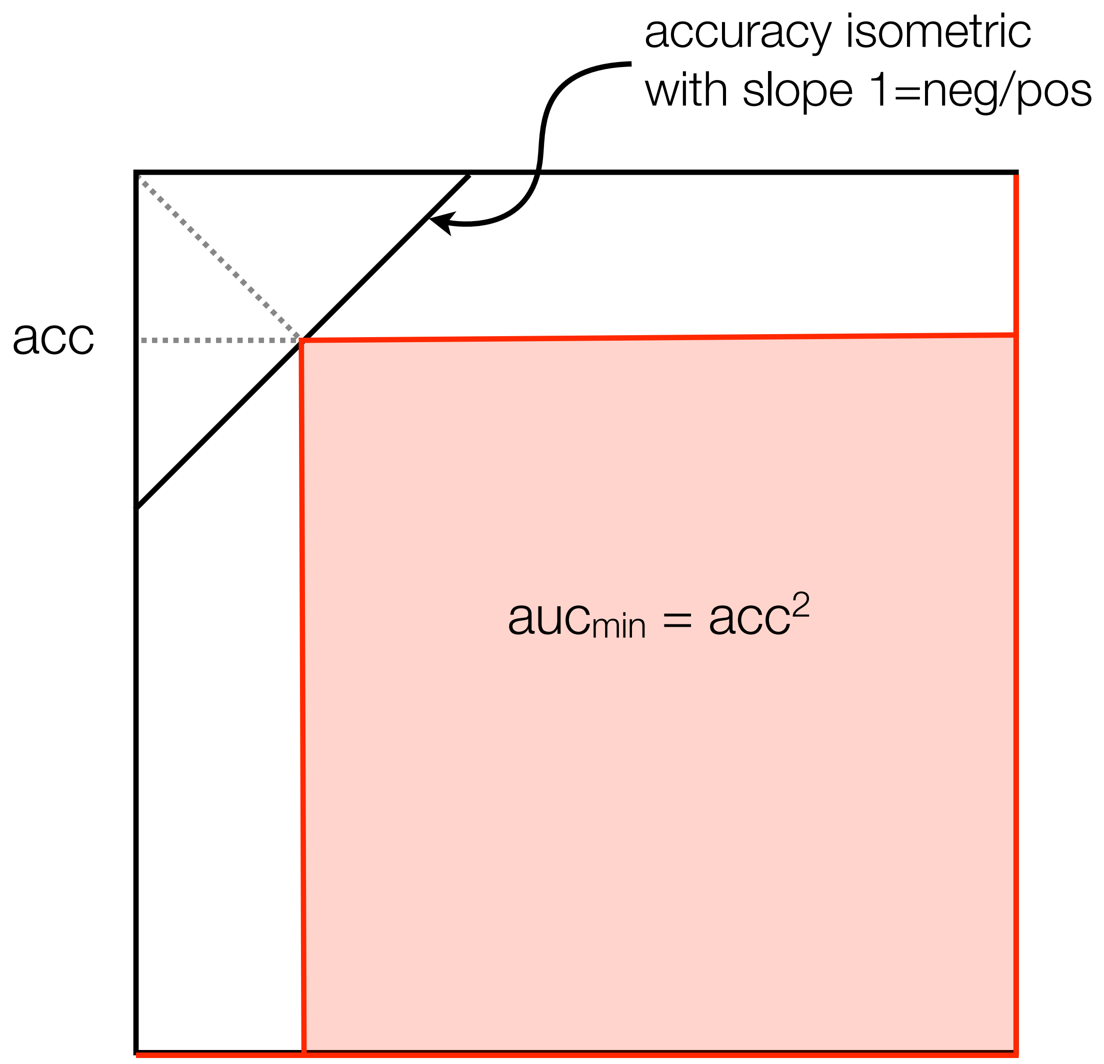
From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve
- The expected value of AUC is
  - $E[auc] = \frac{acc-(pos-neg)^2}{4pos \cdot neg}$
- For uniform class distributions:
  - $E[auc] = acc$



From accuracy to AUC

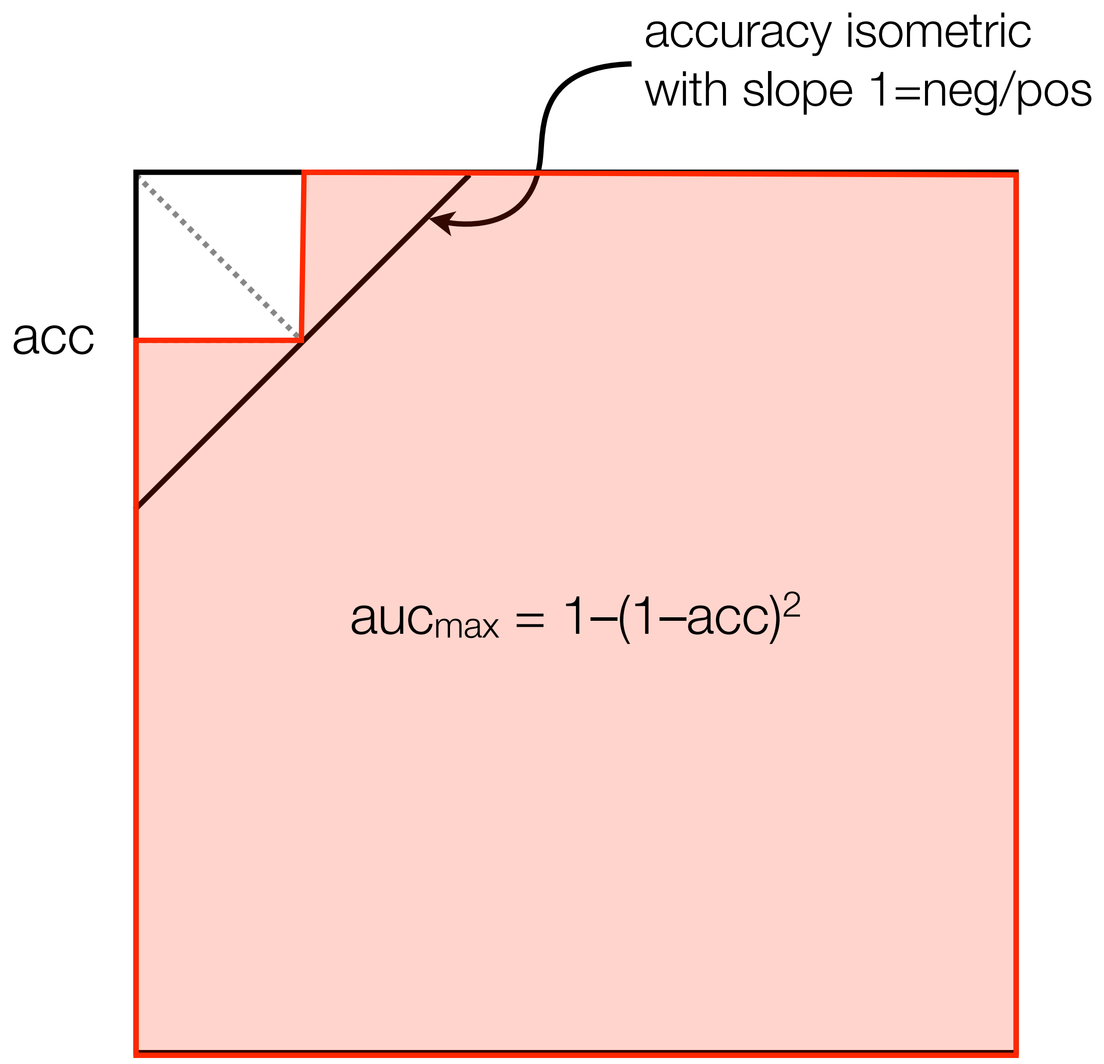
- Given an accuracy value, uniformly construct a random 3-point ROC curve
- The expected value of AUC is
  - $E[auc] = \frac{acc-(pos-neg)^2}{4pos \cdot neg}$
- For uniform class distributions:
  - $E[auc] = acc$
- Again, this has considerable variance:



See also *AUC Optimization vs. Error Rate Minimization*, Corinna Cortes & Mehryar Mohri, NIPS 2003

From accuracy to AUC

- Given an accuracy value, uniformly construct a random 3-point ROC curve
- The expected value of AUC is
    - $E[auc] = \frac{acc-(pos-neg)^2}{4pos \cdot neg}$
- For uniform class distributions:
    - $E[auc] = acc$
- Again, this has considerable variance:


accuracy isometric with slope 1=neg/pos
acc
auc$_{max}$ = 1–(1–acc)^2

See also *AUC Optimization vs. Error Rate Minimization*, Corinna Cortes & Mehryar Mohri, NIPS 2003

Classification and ranking — summary

- While we obtain linear relationships between accuracy (AUC) and the *expected value* of AUC (accuracy), the variances can be large
  - hence optimising these objective functions may still lead to different results
- If certain models are found to optimise both accuracy and AUC, the reason is model-specific
  - e.g., loss functions approximately coincide (SVMs, AdaBoost/RankBoost)
  - e.g., model produces convex ROC curve (decision tree)

III Ranking and probability estimation

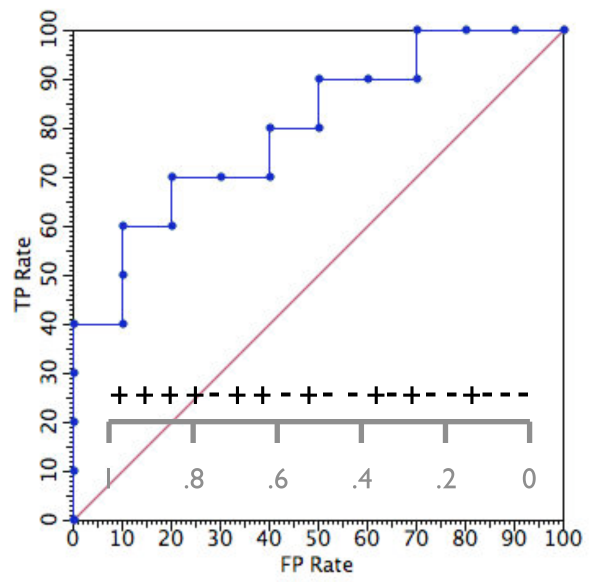
- Uncalibrated probability estimators, such as naive Bayes, are just rankers
  - e.g., we cannot assign any particular significance to p=0.5, and must learn the threshold from the data
  - scores being normalised between 0 and 1 doesn’t imply that they represent meaningful probabilities
- So, good probability estimators need to be well-calibrated *in addition to* being good rankers
  - notice that MSE is O(n), AUC is O(n log n)

Calibration

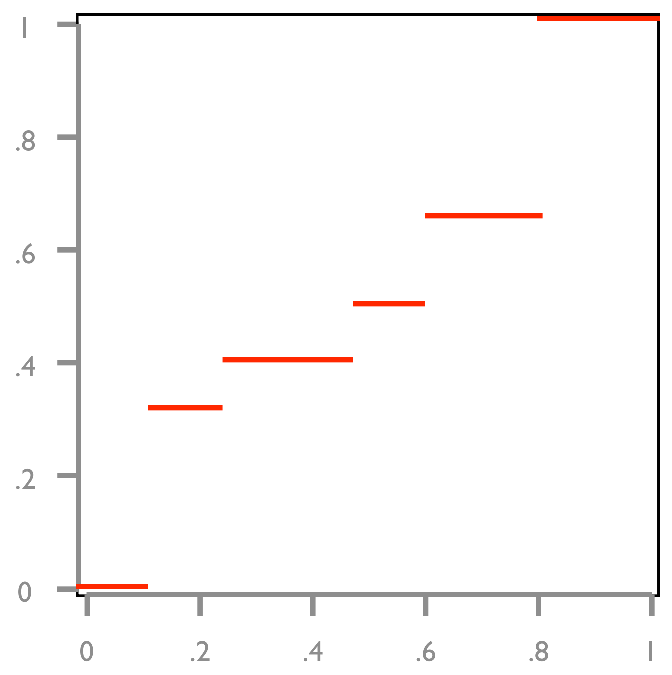
- Well-calibrated probabilities have the following property:
  - conditioning a test sample on predicted probability p, the expected proportion of positives is close to p
- Thus, the predicted likelihood ratio approximates the slope of the ROC curve
  - perfect calibration implies convex ROC curve
- This suggests a simple calibration procedure:
  - discretise scores using convex hull and derive probability in each bin from ROC slope (times prior odds)
    - = isotonic regression (Zadrozny & Elkan, ICML'01; Fawcett & Niculescu-Mizil, MLJ'07; Flach & Matsubara, ECML'07)
    - notice that this is exactly what decision trees do

Calibration through the ROC convex hull

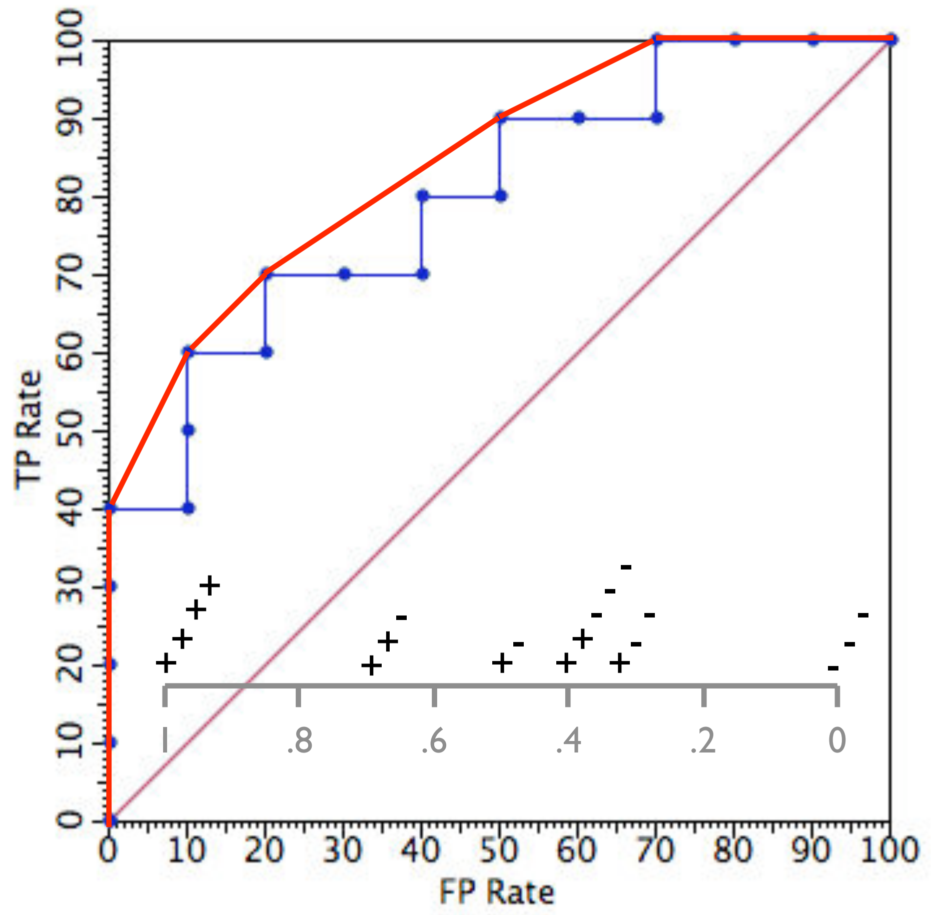
Original scores



Calibration map



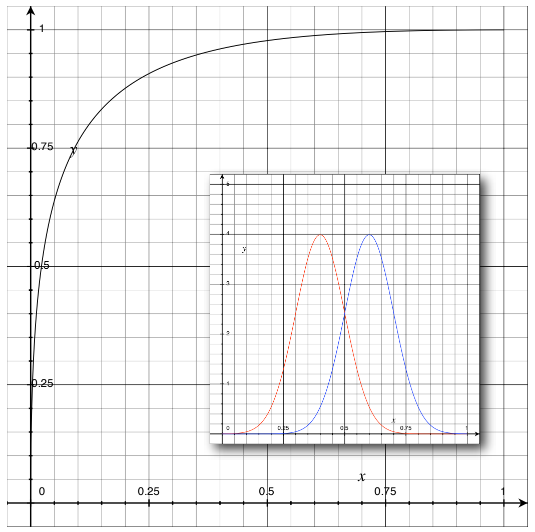
Calibrated scores



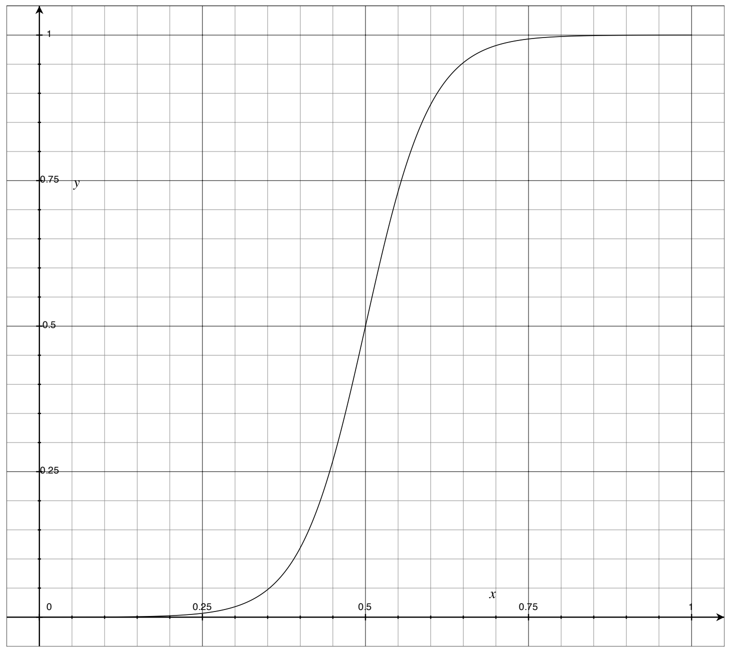
Piecewise constant calibration map leads to more ties in the ranking.

Alternative: logistic calibration

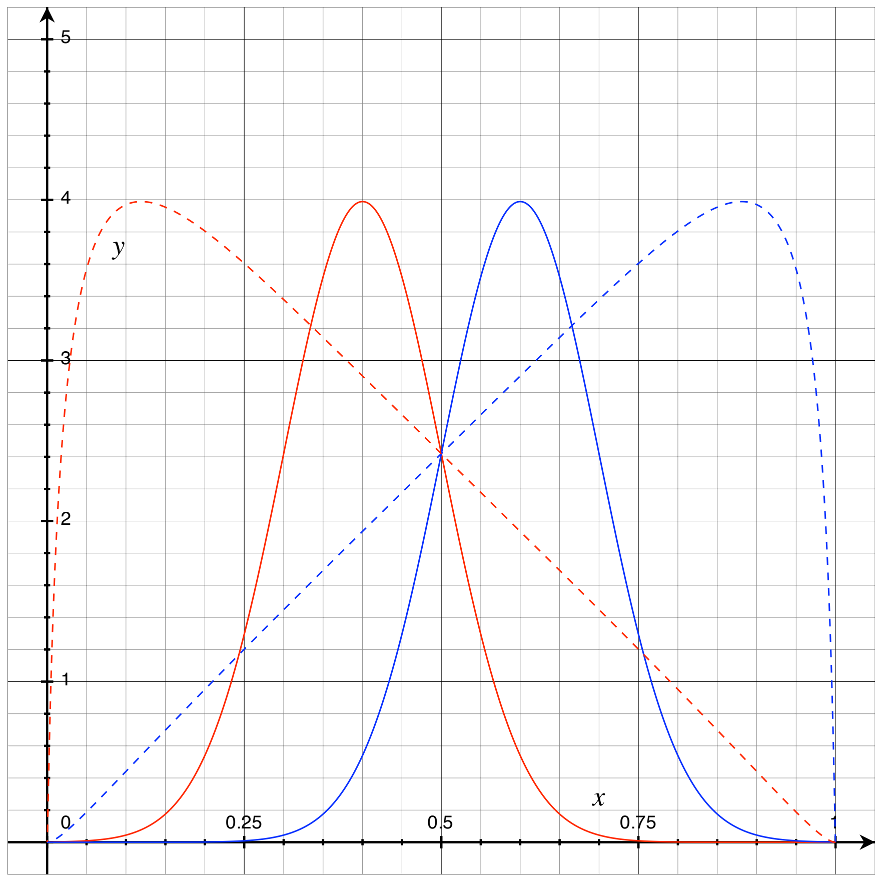
Normally distributed scores



Calibration map: logistic function



Score distributions after calibration



Logistic regression and neural networks have this built in.

Decomposing the Brier score

- The Brier score (mean squared error) measures the average squared deviation from the *ideal* scores 0 and 1
  - ideal scores ≠ true scores
- If p positives and n negatives all receive the same probability score s, their contribution to the Brier score can be decomposed as below (s′=p/(p+n)):

$$\begin{aligned} & p(1-s)^2 + ns^2 \\ = \quad & (p+n)s^2 - 2ps + p \\ = \quad & (p+n)(s^2 - 2ss' + s') \\ = \quad & (p+n)((s-s')^2 + s'(1-s')) \end{aligned}$$

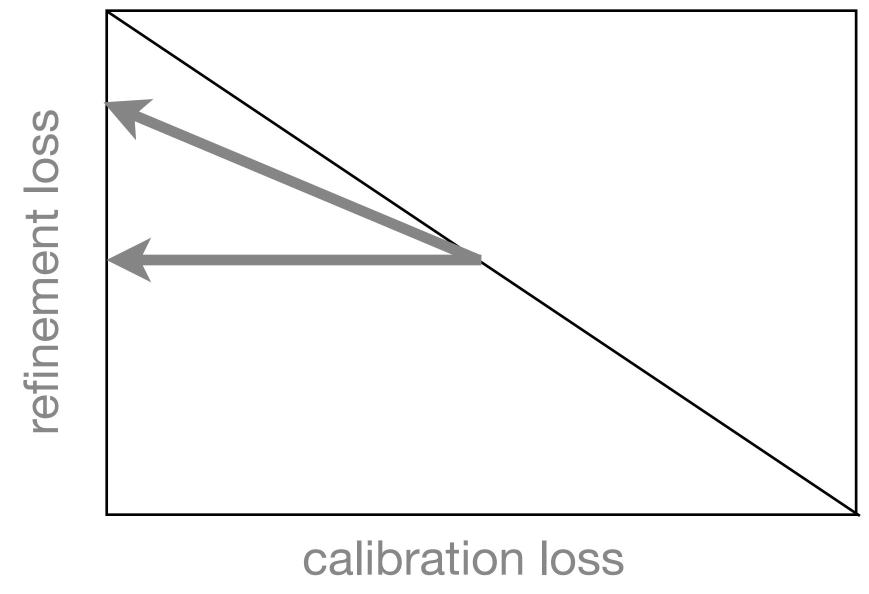
- The first term measures the squared deviation of the predicted probability from the proportion of positives (‘true’ scores); the second is an impurity term independent of the predicted probability

Decomposing the Brier score (Flach & Matsubara, ECML'07)

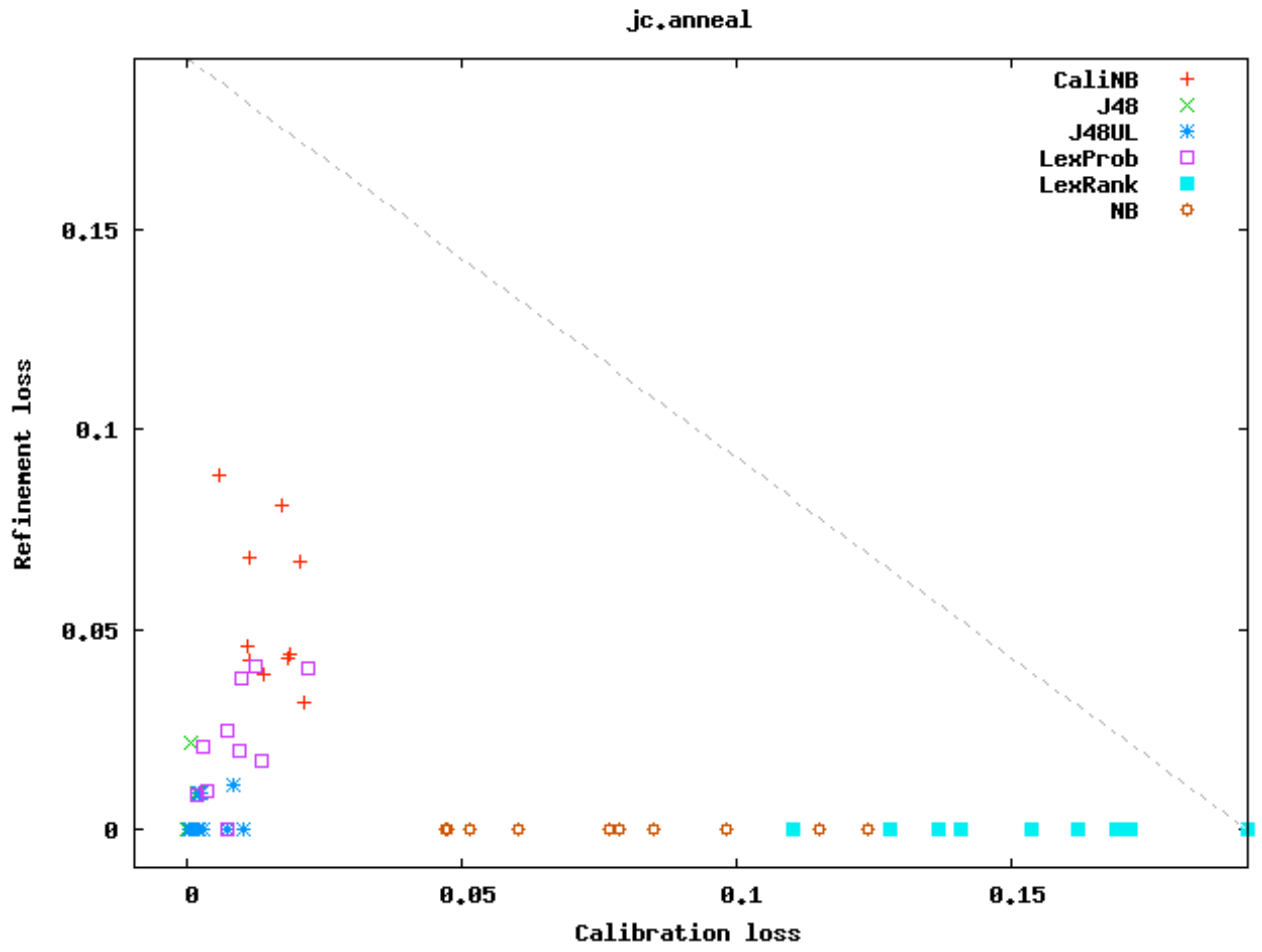
- We can perform this decomposition on each segment of the ROC curve to obtain the following *exact* decomposition of the Brier score:

$$BS = \underbrace{\frac{1}{|X|}\sum_i |X_i|(s_i - s_i')^2}_{\text{calibration loss}} + \underbrace{\frac{1}{|X|}\sum_i |X_i| s_i'(1-s_i')}_{\text{refinement loss}}$$

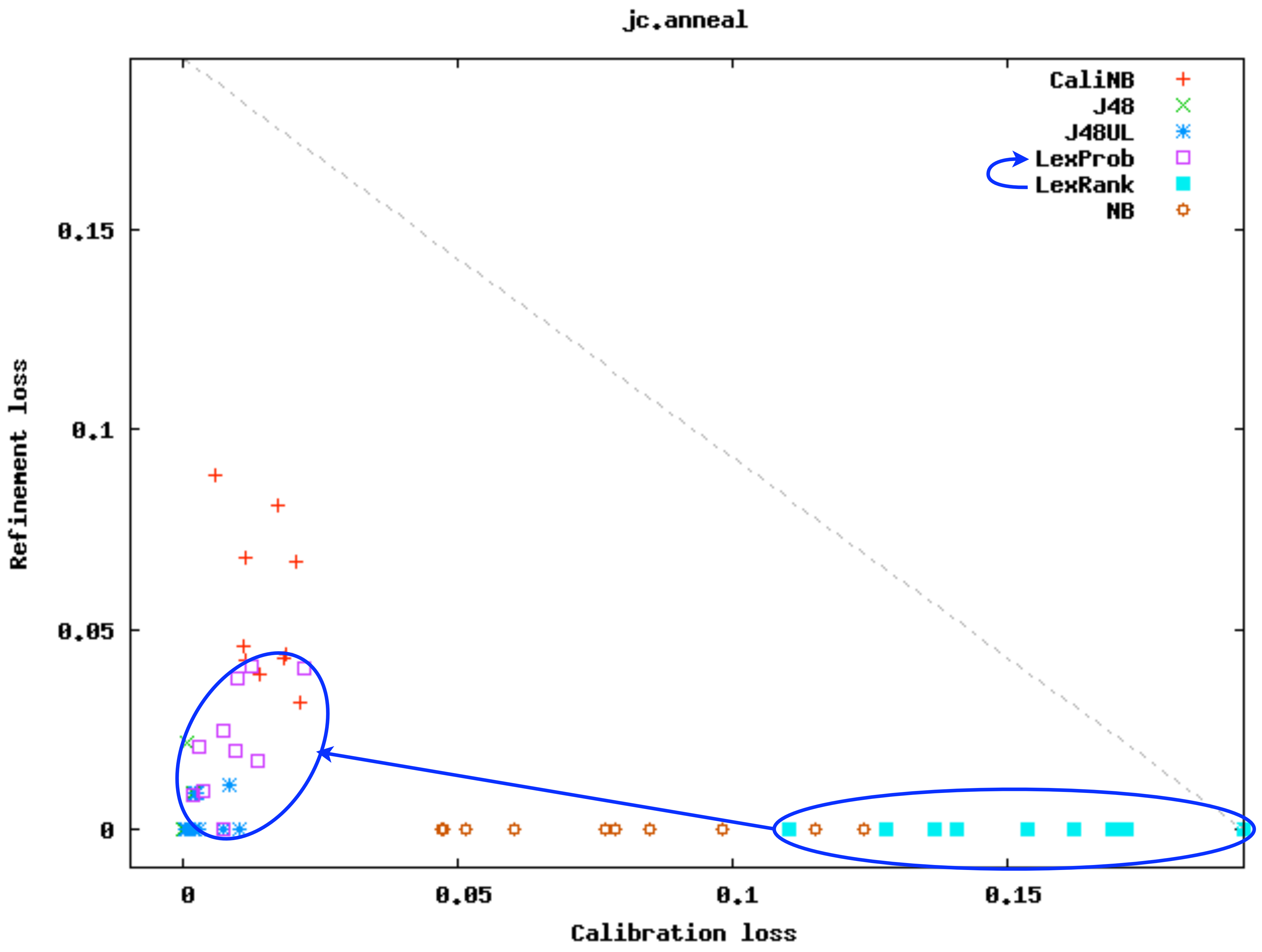
- A similar but inexact decomposition is known from forecasting theory
- The calibration loss relates scores to ROC slopes (0 only for convex curve)
- Refinement loss is incurred by segments that are not horizontal or vertical: it measures the amount of ties in the ranking



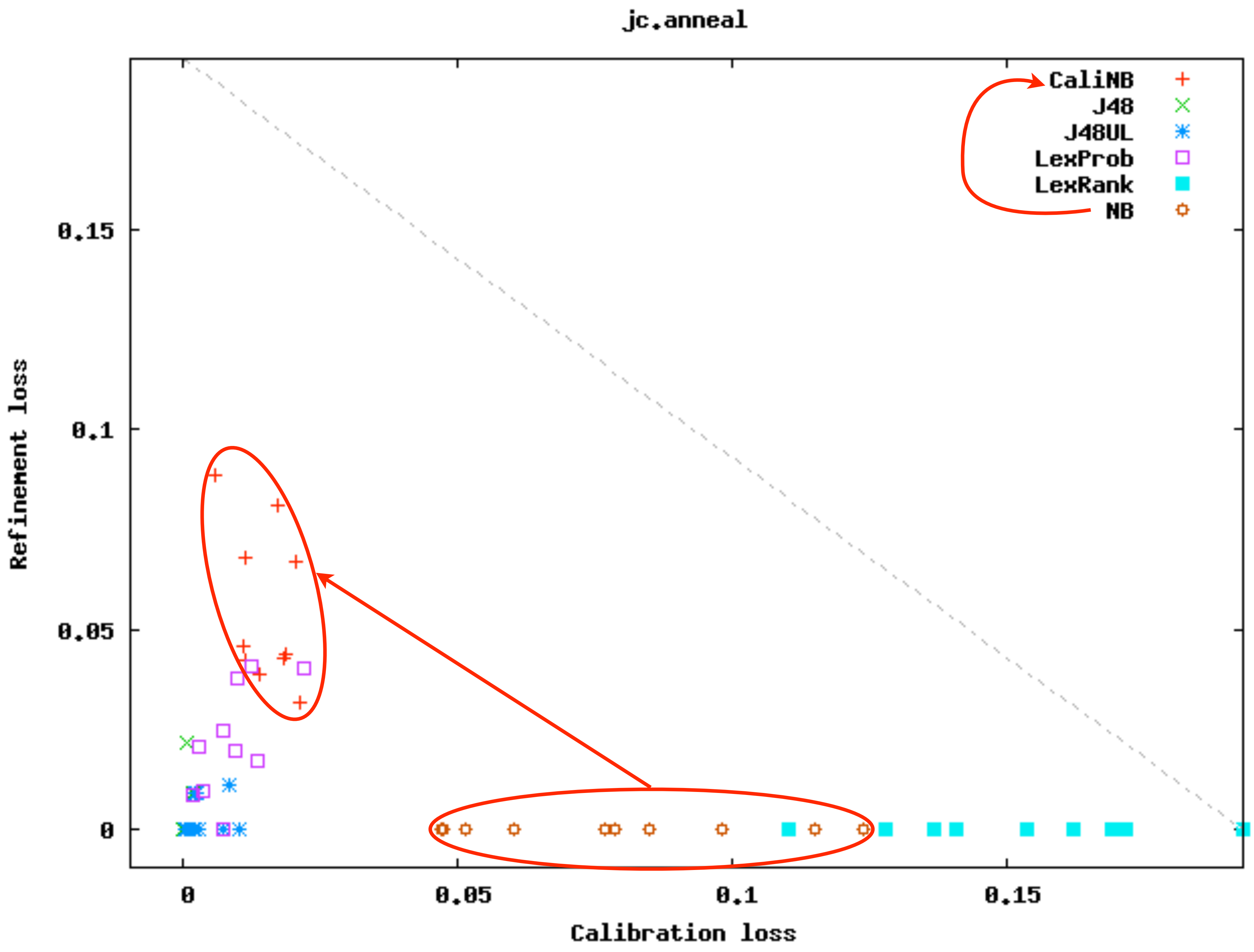
Refinement vs. calibration plot



Refinement vs. calibration plot



Refinement vs. calibration plot



Ranking and probability estimation — summary

- Uncalibrated “probability estimators” are just rankers whose scores happen to be in the range [0, 1]
- Only with calibrated probability estimators can we
  - threshold the score at 0.5 to set the decision threshold at posterior odds 1
  - more generally, threshold the score at p if we desire posterior odds p/(1-p)
- Probability estimators need to be evaluated using a combination of AUC to measure ranking performance and calibration loss to measure quality of the probability estimates

Concluding remarks

- Ranking is a fundamental notion underlying both classification and probability estimation
  - the ROC curve, its convex hull, a calibration map, calibration & refinement loss before and after calibration, can all be obtained by a single sweep through a ranked list of labelled examples
- For obtaining calibrated probability estimates we have a choice between parametric (logistic) and non-parametric (isotonic) methods
  - logistic calibration doesn't change the ranking, hence refinement loss stays the same; but calibration loss may increase if scores are not normally distributed; being a parametric method it generally requires less data
  - isotonic calibration improves the ranking through the ROC convex hull, which discretises the scores; but estimating the bin boundaries requires more data and is also brittle

Some open questions

- Calibration is linked to convexity of the ROC curve — what is the right measure of convexity?
- Is there advantage in smoothing the bin boundaries in isotonic calibration?
- Is there advantage in incorporating predicted scores in the ROC curve? (Wu, Flach & Ferri, ECML'07)
- (as always with ROC analysis) How to extend the analysis to more than two classes?