Introducción al aprendizaje automático

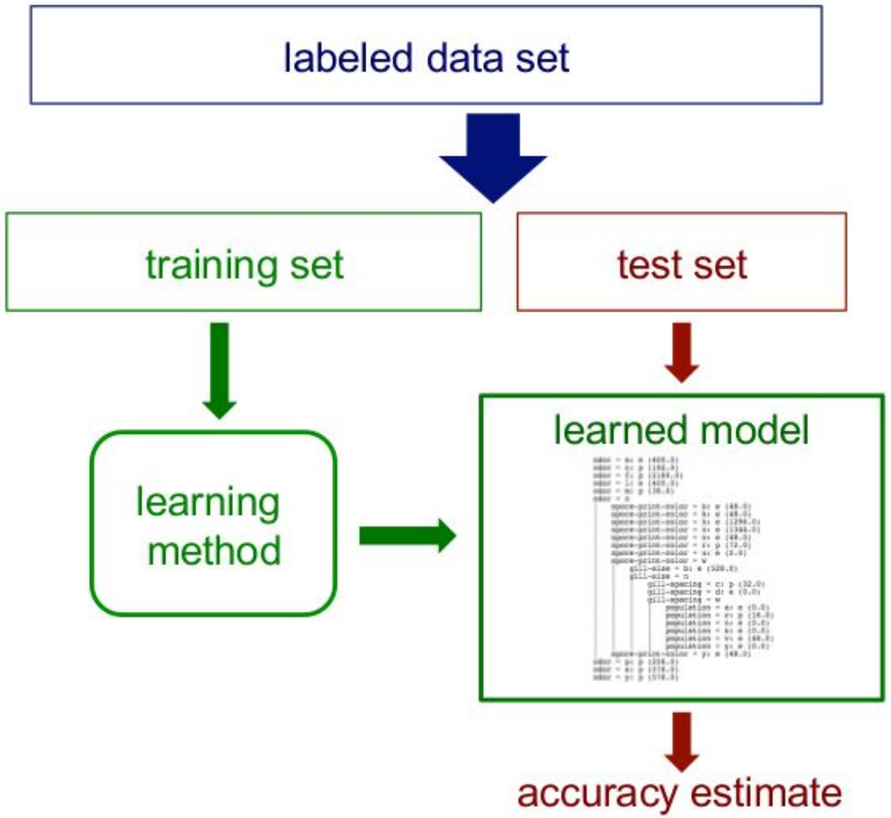
•••

#4. Métricas y medidas de performance

Slides: D. Page, P. Flach

Test sets revisited

How can we get an unbiased estimate of the accuracy of a learned model?



Test sets revisited

How can we get an unbiased estimate of the accuracy of a learned model?

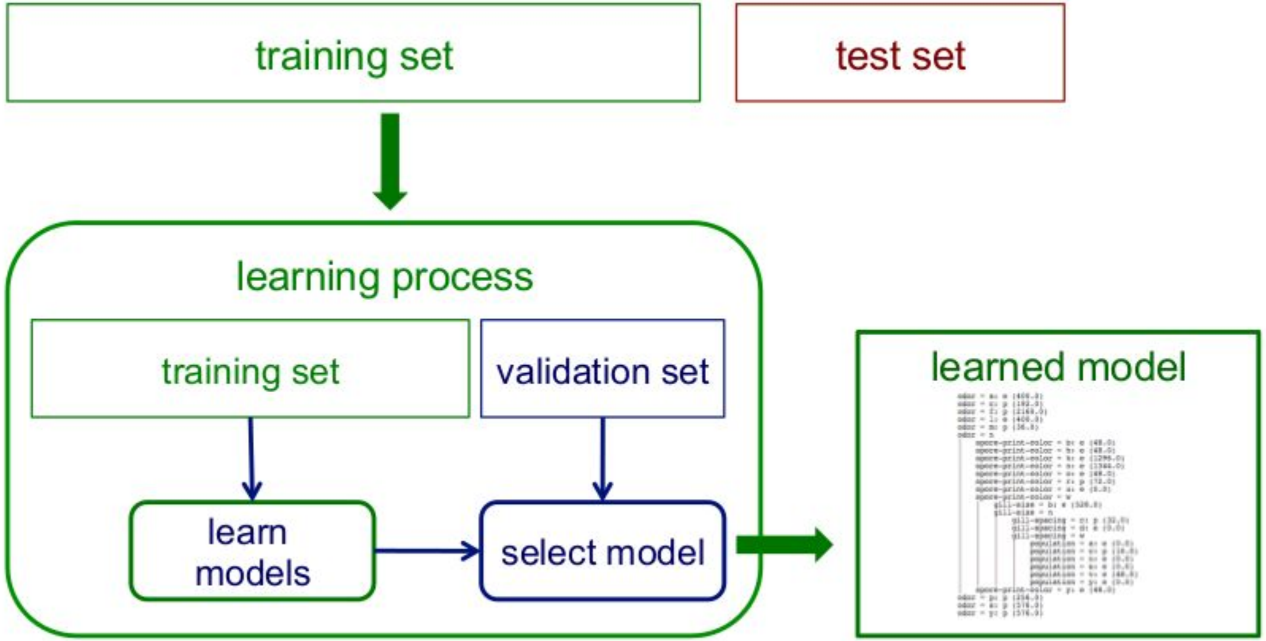
- when learning a model, you should pretend that you don't have the test data yet (it is "in the mail")*

- if the test-set labels influence the learned model in any way, accuracy estimates will be biased

* In some applications it is reasonable to assume that you have access to the feature vector (i.e. $\boldsymbol{x}$) but not the y part of each test instance.

Validation (tuning) sets revisited

Suppose we want unbiased estimates of accuracy during the learning process (e.g. to choose the best level of decision-tree pruning)?



```
odor = a: e (400.0)
odor = c: p (192.0)
odor = f: p (2160.0)
odor = l: e (400.0)
odor = m: p (36.0)
odor = n
|   spore-print-color = b: e (48.0)
|   spore-print-color = h: e (48.0)
|   spore-print-color = k: e (1296.0)
|   spore-print-color = n: e (1344.0)
|   spore-print-color = o: e (48.0)
|   spore-print-color = r: p (72.0)
|   spore-print-color = u: e (0.0)
|   spore-print-color = w
|   |   gill-size = b: e (528.0)
|   |   gill-size = n
|   |   |   gill-spacing = c: p (32.0)
|   |   |   gill-spacing = d: e (0.0)
|   |   |   gill-spacing = w
|   |   |   |   population = a: e (0.0)
|   |   |   |   population = c: p (16.0)
|   |   |   |   population = n: e (0.0)
|   |   |   |   population = s: e (0.0)
|   |   |   |   population = v: e (48.0)
|   |   |   |   population = y: e (0.0)
|   spore-print-color = y: e (48.0)
odor = p: p (256.0)
odor = s: p (576.0)
odor = y: p (576.0)
```

Partition training data into separate training/validation sets

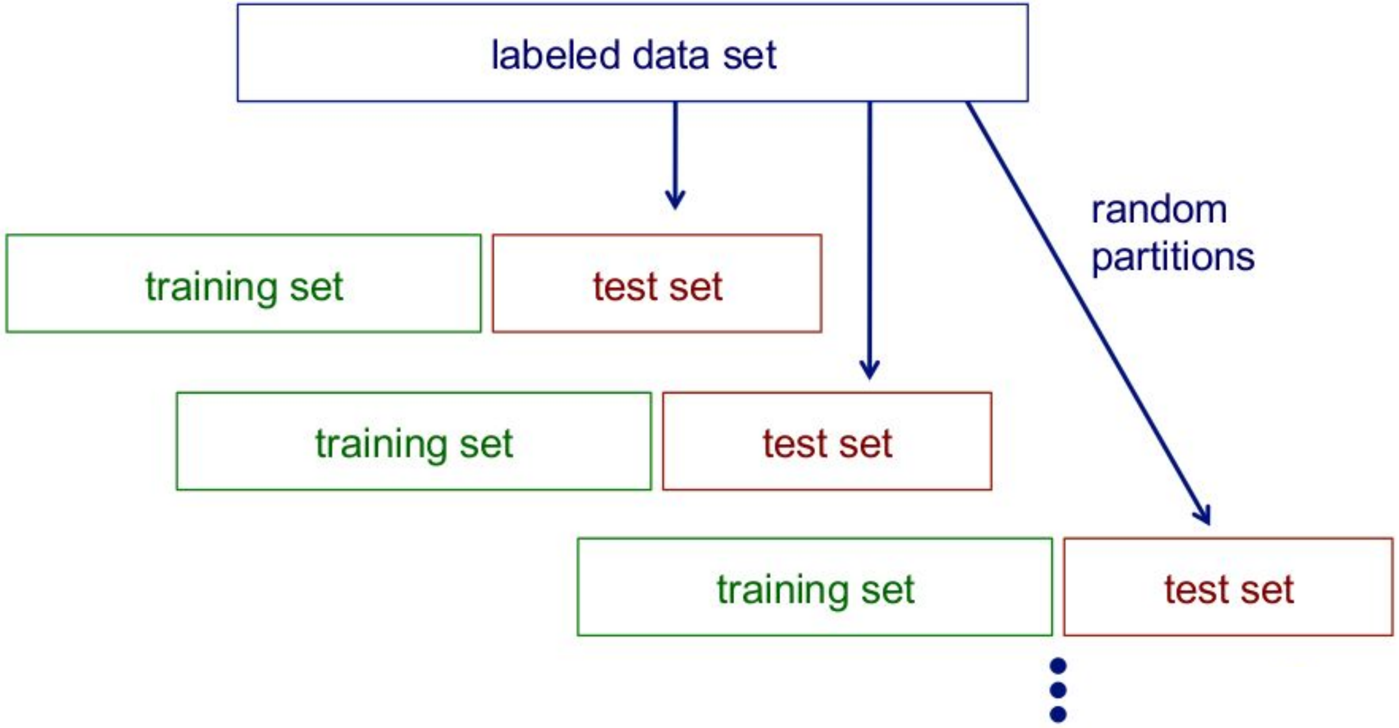
Limitations of using a single training/test partition

- we may not have enough data to make sufficiently large training and test sets
    - a <u>larger test set</u> gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
    - but… a <u>larger training set</u> will be more representative of how much data we actually have for learning process

- a single training set doesn’t tell us how sensitive accuracy is to a particular training sample

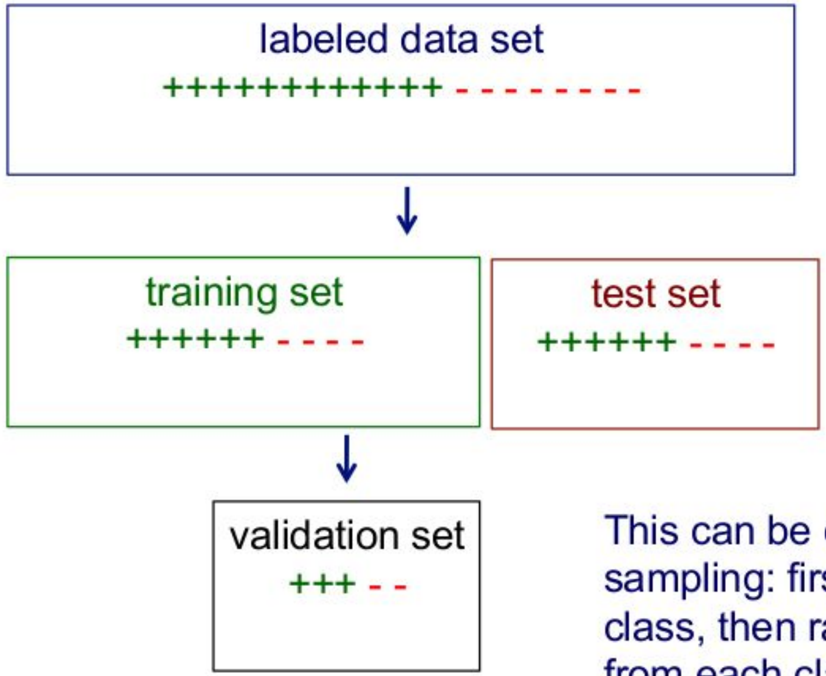
Random resampling

We can address the second issue by repeatedly randomly partitioning the available data into training and set sets.



Stratified sampling

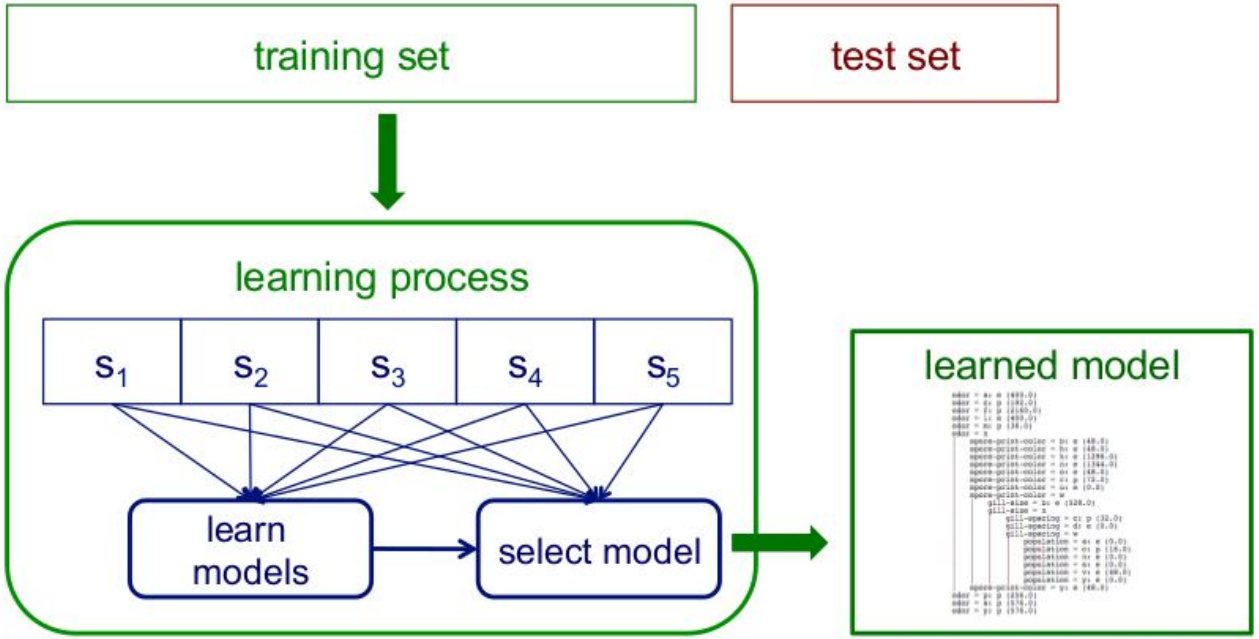
When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set



This can be done via stratified sampling: first stratify instances by class, then randomly select instances from each class proportionally.

Internal cross validation

Instead of a single validation set, we can use cross-validation within a training set to select a model (e.g. to choose the best level of decision-tree pruning)

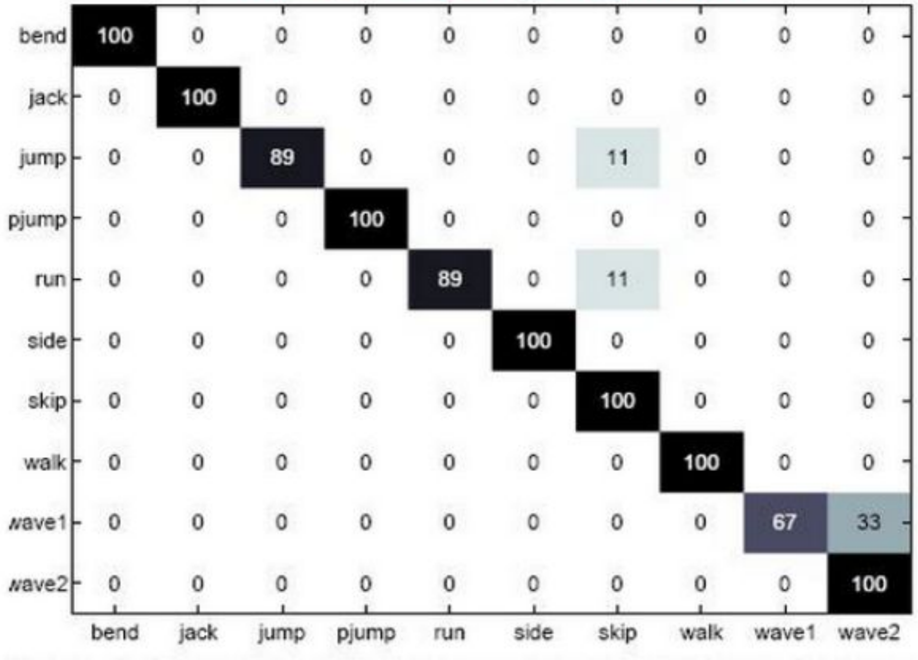


```
odor = a: e (400.0)
odor = c: p (192.0)
odor = f: p (2160.0)
odor = l: e (400.0)
odor = m: p (36.0)
odor = n
|   spore-print-color = b: e (48.0)
|   spore-print-color = h: e (48.0)
|   spore-print-color = k: e (1296.0)
|   spore-print-color = n: e (1344.0)
|   spore-print-color = o: e (48.0)
|   spore-print-color = r: p (72.0)
|   spore-print-color = u: e (0.0)
|   spore-print-color = w
|   |   gill-size = b: e (528.0)
|   |   gill-size = n
|   |   |   gill-spacing = c: p (32.0)
|   |   |   gill-spacing = d: e (0.0)
|   |   |   gill-spacing = w
|   |   |   |   population = a: e (0.0)
|   |   |   |   population = c: p (16.0)
|   |   |   |   population = n: e (0.0)
|   |   |   |   population = s: e (0.0)
|   |   |   |   population = v: e (48.0)
|   |   |   |   population = y: e (0.0)
|   spore-print-color = y: e (48.0)
odor = p: p (256.0)
odor = s: p (576.0)
odor = y: p (576.0)
```

Confusion matrices

How can we understand what types of mistakes a learned model makes?

activity recognition from video



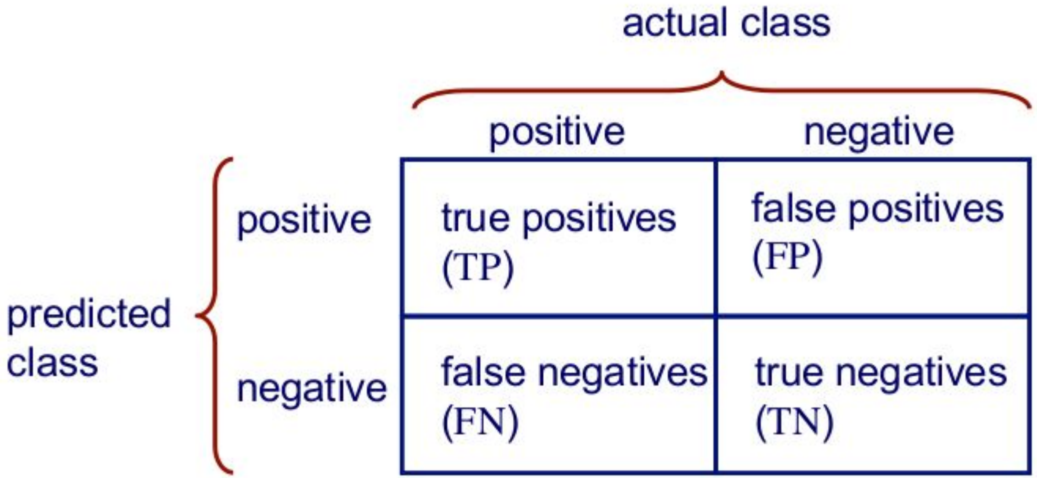
	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2	0	0	0	0	0	0	0	0	0	100

actual class

predicted class

figure from vision.jhu.edu

Confusion matrix for 2-class problems



		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

FP = false alarm = Type I error
FN = miss = Type II error
P/N: prediction
F/N: correctness

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

accuracy = POS * TPR + NEG * (1-FPR)

TPR = true positive rate = TP / POS = TP / (TP + FN)
FPR = true positive rate = FP / NEG = FP / (FP + TN)

Is accuracy an adequate measure of predictive performance?

- accuracy may not be useful measure in cases where
  - there is a large class skew
    - Is 98% accuracy good if 97% of the instances are negative?
  - there are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong
    - Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease
  - we are most interested in a subset of high-confidence predictions

Other accuracy metrics

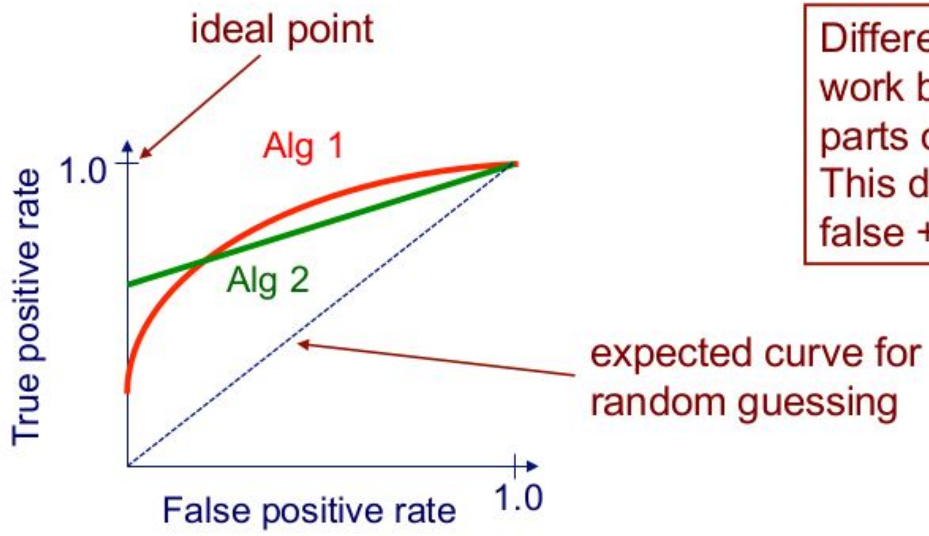
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

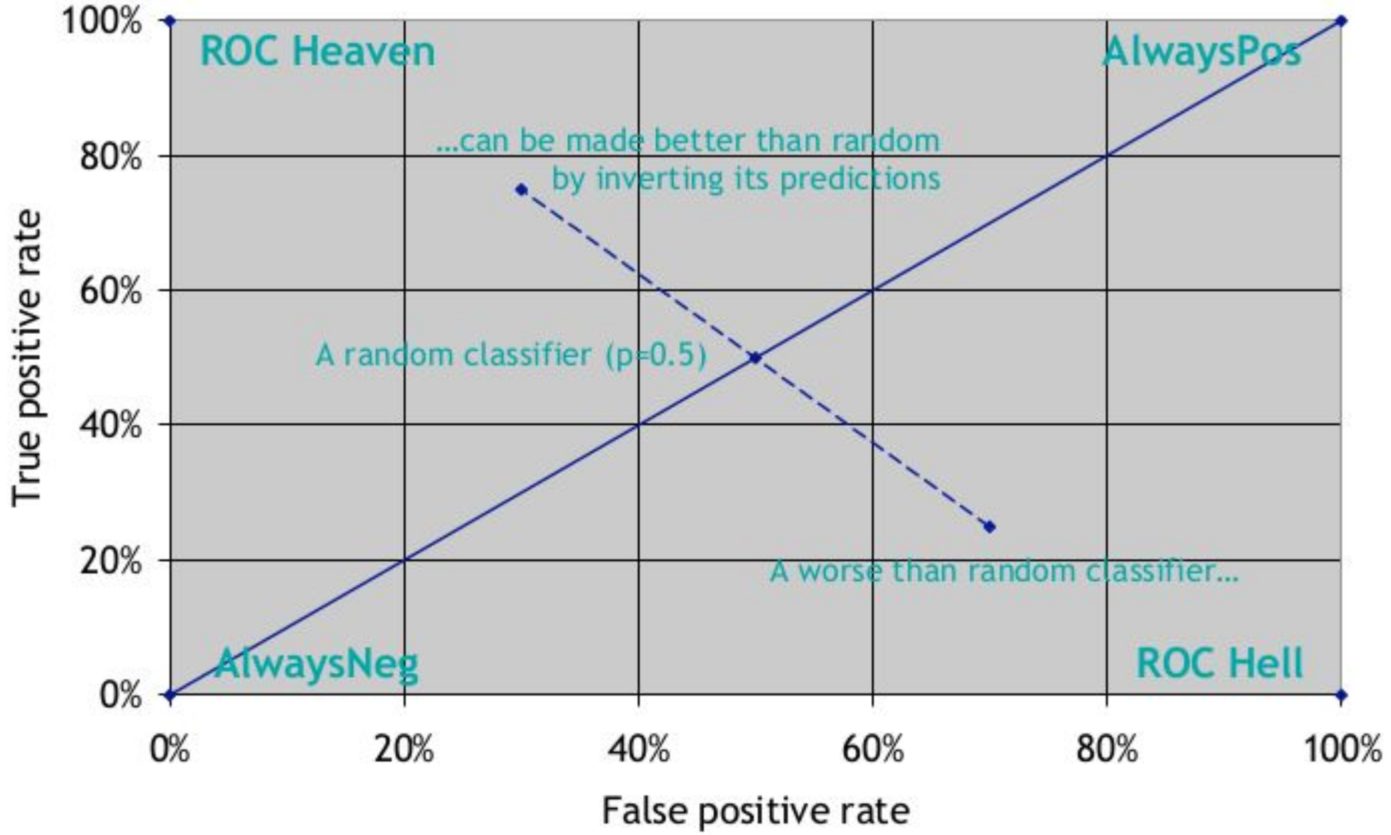
$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

ROC curves

A *Receiver Operating Characteristic* (*ROC*) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied



Different methods can work better in different parts of ROC space. This depends on cost of false + vs. false -

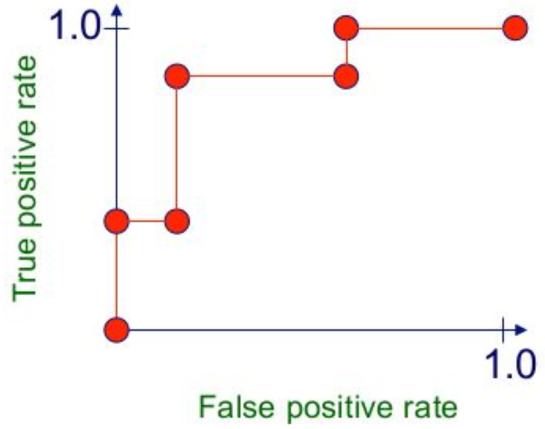

ROC Heaven
AlwaysPos
...can be made better than random by inverting its predictions
A random classifier (p=0.5)
A worse than random classifier...
AlwaysNeg
ROC Hell
True positive rate
False positive rate
0%
20%
40%
60%
80%
100%

Algorithm for creating an ROC curve

1. sort test-set predictions according to confidence that each instance is positive
2. step through sorted list from high to low confidence
   1. locate a *threshold* between instances with opposite classes (keeping instances with the same confidence value on the same side of threshold)
   2. compute TPR, FPR for instances above threshold
   3. output (FPR, TPR) coordinate

Plotting an ROC curve

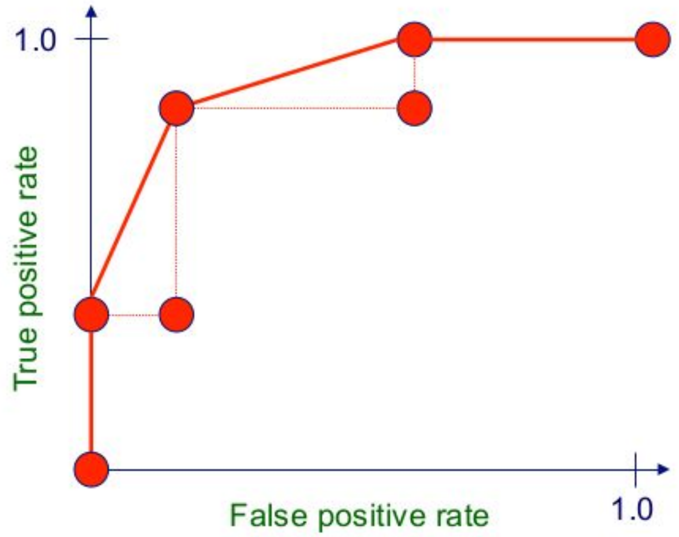
instance	confidence positive		correct class
Ex 9	.99		+
Ex 7	.98	TPR= 2/5, FPR= 0/5	+
Ex 1	.72	TPR= 2/5, FPR= 1/5	-
Ex 2	.70		+
Ex 6	.65	TPR= 4/5, FPR= 1/5	+
Ex 10	.51		-
Ex 3	.39	TPR= 4/5, FPR= 3/5	-
Ex 5	.24	TPR= 5/5, FPR= 3/5	+
Ex 4	.11		-
Ex 8	.01	TPR= 5/5, FPR= 5/5	-



Plotting an ROC curve

can interpolate between points to get *convex hull*

- convex hull: repeatedly, while possible, perform interpolations that skip one data point and discard any point that lies below a line
- interpolated points are achievable in theory: can flip weighted coin to choose between classifiers represented by plotted points



Other accuracy metrics

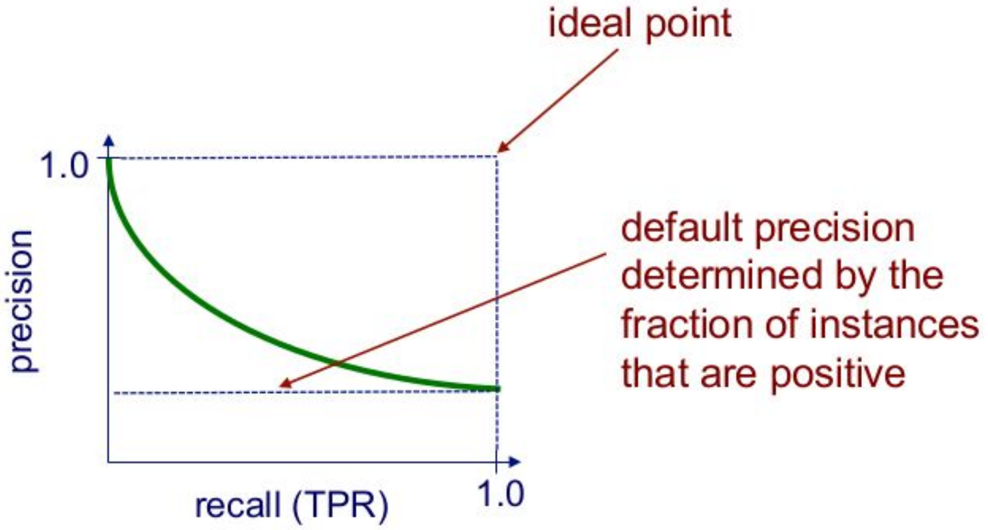
		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

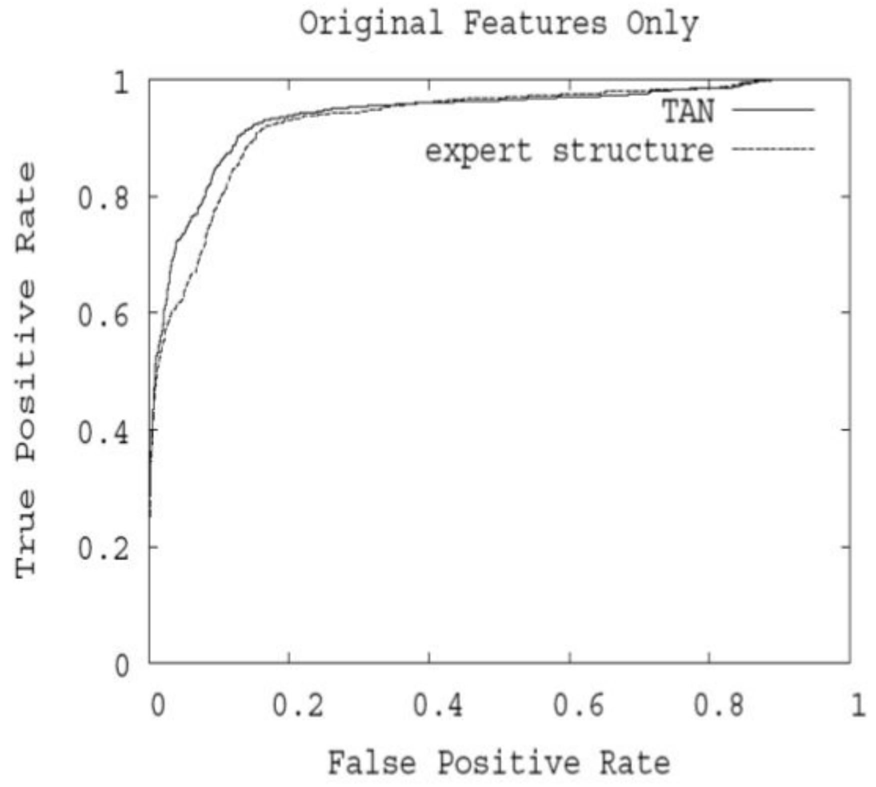
$$\text{precision} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision/recall curves

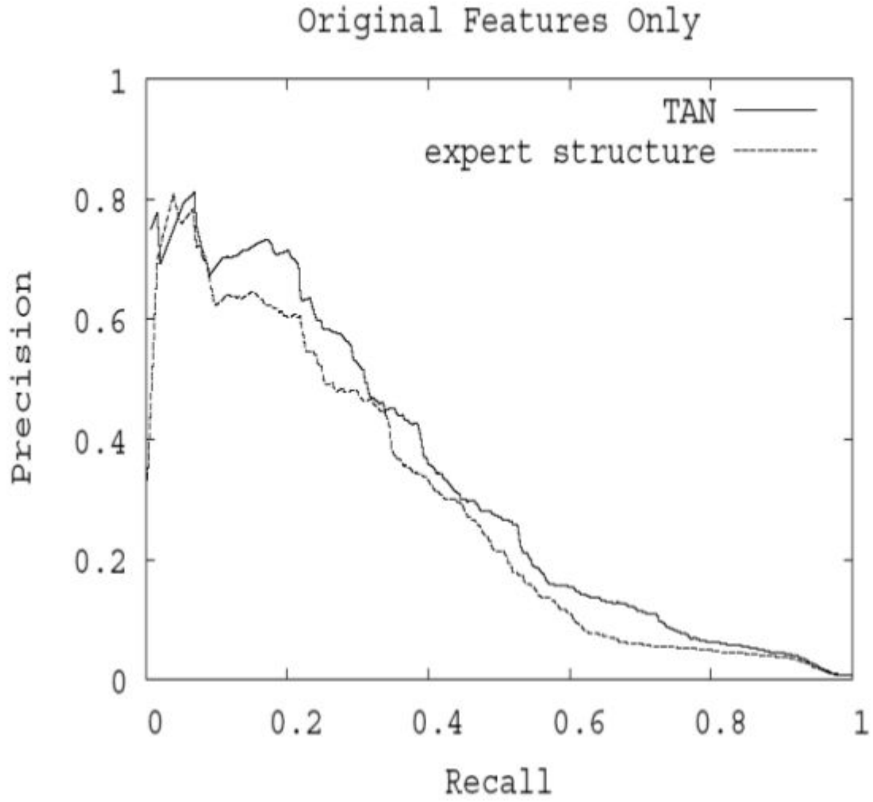
A *precision/recall curve* plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied



Mammography Example: ROC



Mammography Example: PR



How do we get one ROC/PR curve when we do cross validation?

Approach 1

- make assumption that confidence values are comparable across folds
- pool predictions from all test sets
- plot the curve from the pooled predictions

Approach 2 (for ROC curves)

- plot individual curves for all test sets
- view each curve as a function
- plot the average curve for this set of functions

Comments on ROC and PR curves

both

- allow predictive performance to be assessed at various levels of confidence
- assume binary classification tasks
- sometimes summarized by calculating *area under the curve*

ROC curves

- insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
- can identify optimal classification thresholds for tasks with differential misclassification costs

precision/recall curves

- show the fraction of predictions that are false positives
- well suited for tasks with lots of negative instances

The AUC metric

- The Area Under ROC Curve (AUC) assesses the ranking in terms of separation of the classes
  - all the +ves before the –ves: AUC=1
  - random ordering: AUC=0.5
  - all the –ves before the +ves: AUC=0
- Equivalent to the Mann-Whitney-Wilcoxon sum of ranks test
  - estimates probability that randomly chosen +ve is ranked before randomly chosen –ve
  - $\frac{S_+ - Pos(Pos+1)/2}{Pos \cdot Neg}$ where S_+ is the sum of ranks of +ves
- Gini coefficient = 2*AUC–1 (area above diag.)
  - NB. not the same as Gini index!

Averaging ROC curves

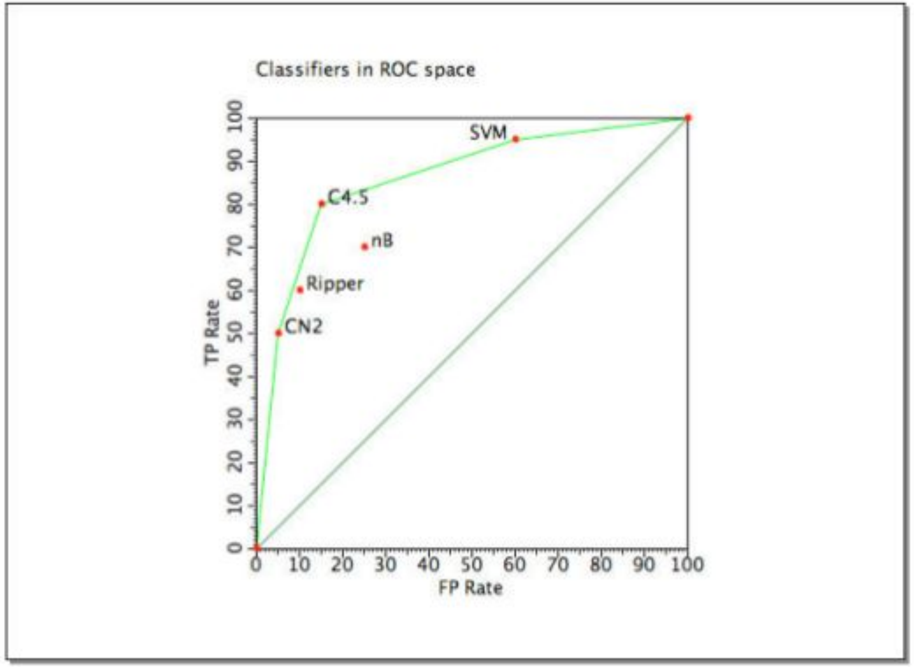
- To obtain a cross-validated ROC curve
  - just combine all test folds with scores for each instance, and draw a single ROC curve
- To obtain cross-validated AUC estimate with error bounds
  - calculate AUC in each test fold and average
  - or calculate AUC from single cv-ed curve and use bootstrap resampling for error bounds
- To obtain ROC curve with error bars
  - vertical averaging (sample at fixed fpr points)
  - threshold averaging (sample at fixed thresholds)
  - see (Fawcett, 2004)

Comparing learning systems

How can we determine if one learning system provides better performance than another

- for a particular task?
- across a set of tasks / data sets?

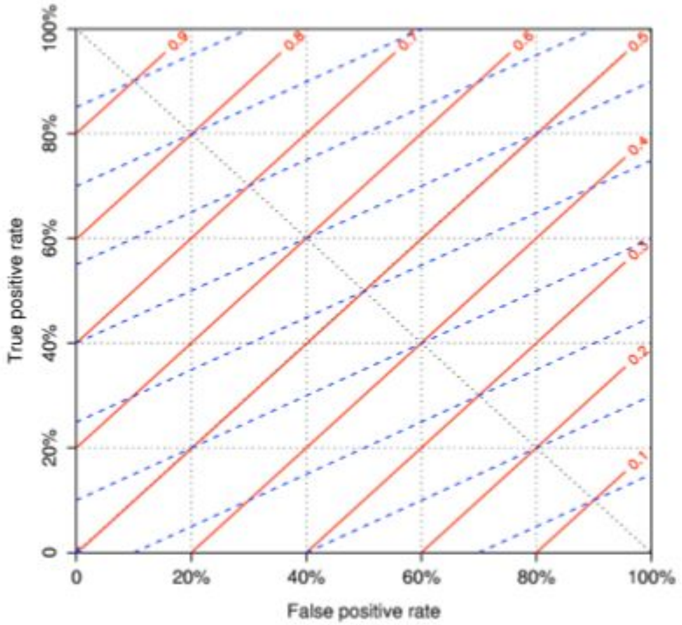
The ROC convex hull



- Classifiers on the convex hull achieve the best accuracy for some class distributions
- Classifiers below the convex hull are always sub-optimal

Iso-accuracy lines

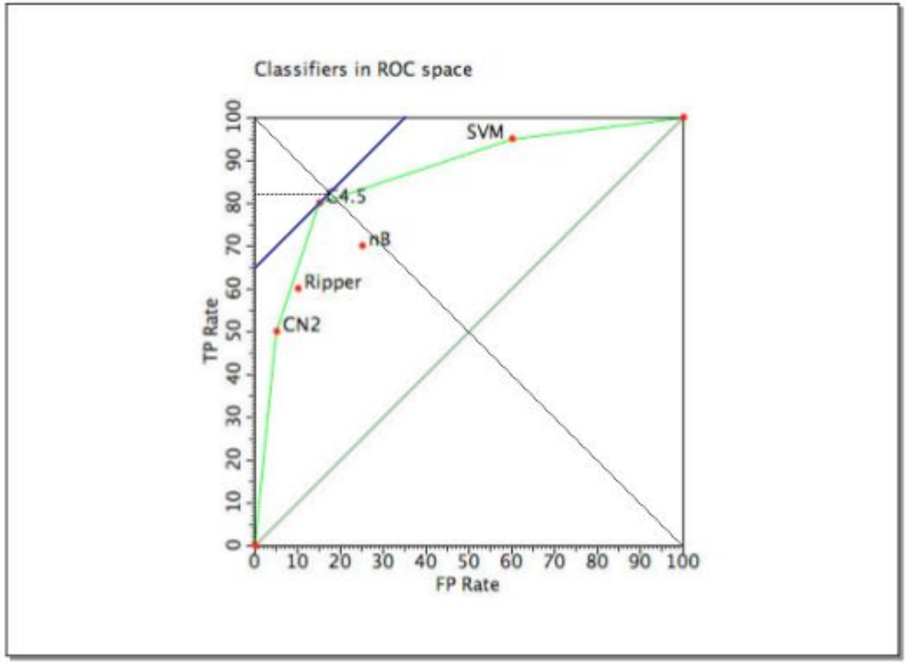
- Iso-accuracy line connects ROC points with the same accuracy
  - $pos * tpr + neg * (1 - fpr) = a$
  - $tpr = \frac{a - neg}{pos} + \frac{neg}{pos} fpr$
- Parallel ascending lines with slope neg/pos
  - higher lines are better
  - on descending diagonal, $tpr = a$



Iso-accuracy & convex hull

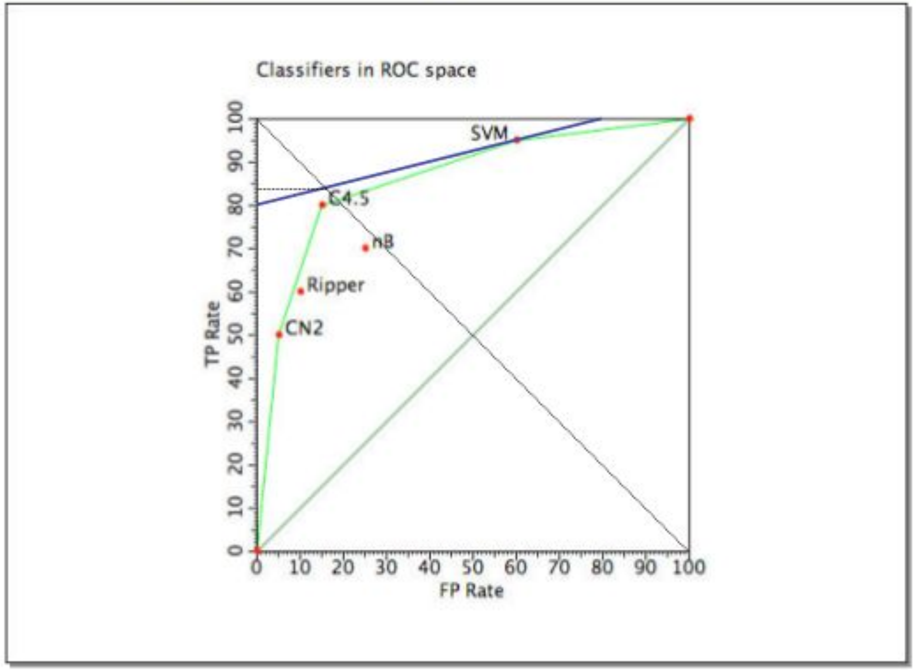
- Each line segment on the convex hull is an iso-accuracy line for a particular class distribution
  - under that distribution, the two classifiers on the end-points achieve the same accuracy
  - for distributions skewed towards negatives (steeper slope), the left one is better
  - for distributions skewed towards positives (flatter slope), the right one is better
- Each classifier on convex hull is optimal for a specific range of class distributions

Selecting the optimal classifier



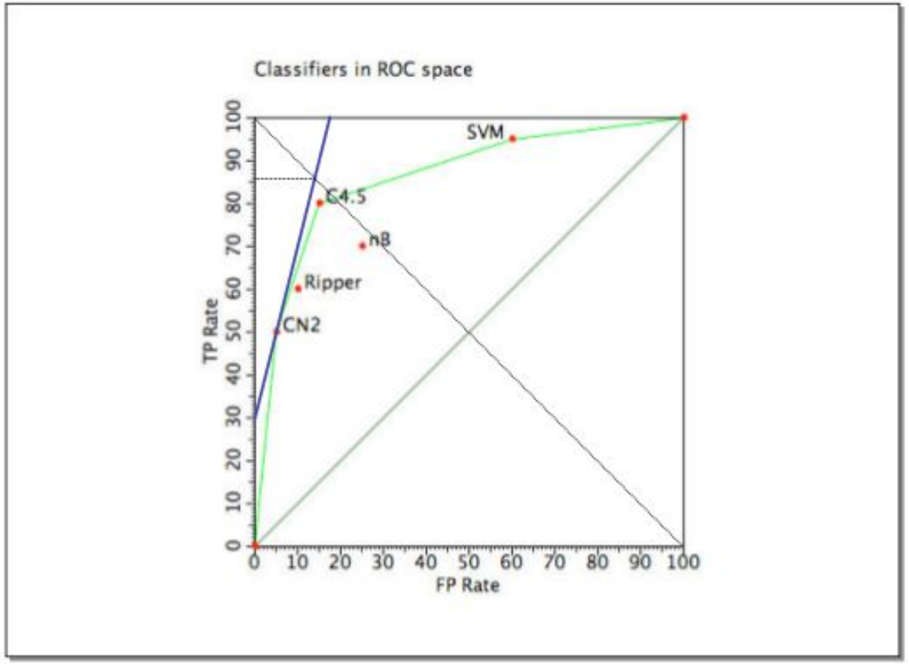
- For uniform class distribution, C4.5 is optimal
  - and achieves about 82% accuracy

Selecting the optimal classifier



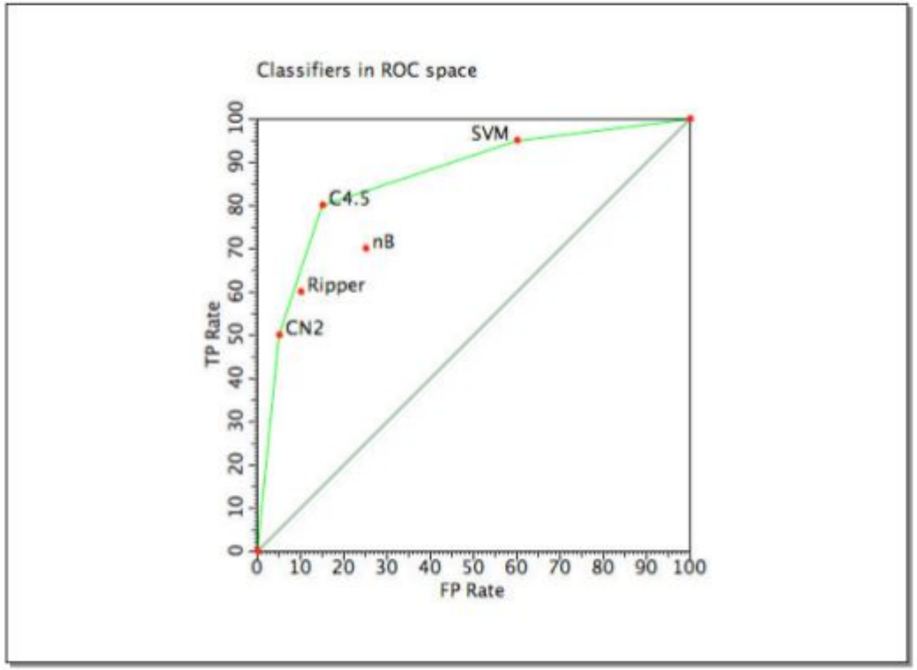
- With four times as many +ves as –ves, SVM is optimal
  - and achieves about 84% accuracy

Selecting the optimal classifier



- With four times as many –ves as +ves, CN2 is optimal
  - and achieves about 86% accuracy

Selecting the optimal classifier



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

Lesion studies

We can gain insight into what contributes to a learning system's performance by removing (lesioning) components of it

The ROC curves here show how performance is affected when various feature types are removed from the learning representation

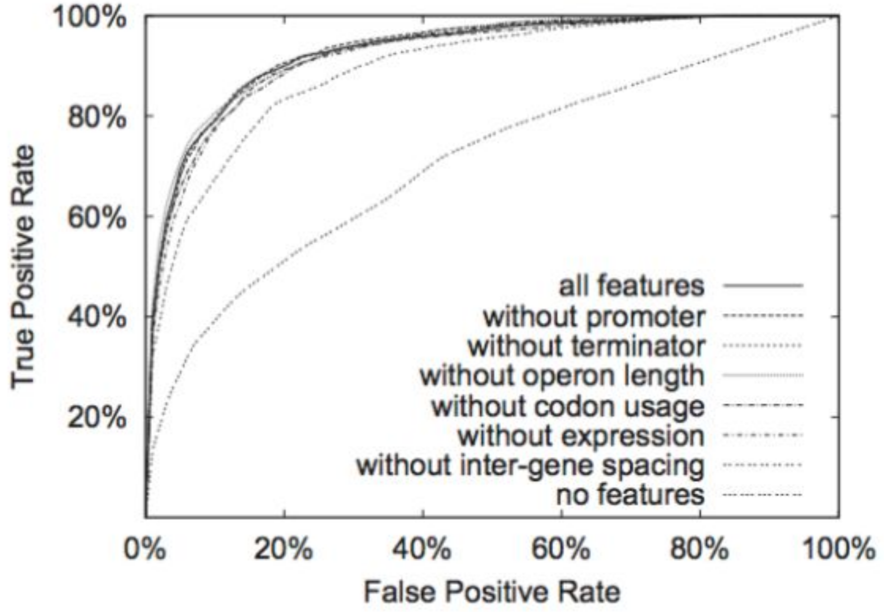


figure from Bockhorst et al., *Bioinformatics* 2003