CN550: Neural and Computational Models of Recognition, Memory, and Attention

Evaluation Methods
Algorithm Independent Strategies

Lecture 4

Instructors: Anatoli Gorchetchnikov <anatoli@cns.bu.edu>
Heather Ames <starfly@cns.bu.edu>
Teaching Fellow: Karthik Srinivasan <skarthik@bu.edu>

Which Algorithm to Choose?

Compare the performance using some metrics

If there is a tie

- choose the simplest algorithm
- choose the smoothest algorithm

Are there general “laws” of pattern classification?

Are there some techniques that can benefit all classifiers?

Off-training Set Error

General desire is to minimize error on all data points:

- Independent identically distributed (IID) error

Since any powerful algorithm can learn the training set, we rather concentrate on minimization of the error for points not in the training set

No Free Lunch Theorem

If generalization performance is our goal then there is no problem-independent reasons to choose one algorithm over the other

Superiority of a specific algorithm stems from

- Nature of the task
- Distribution of the data

Theorem:

- Expected IID error for any two algorithms is the same
- Even for a given training set, off-set error for any two algorithms is the same

Consequence – for good performance on some problems the algorithm has to have equally bad performance on other problems

After Lunch

Key assumption of pattern classification: your chosen algorithm is good for your problem at hand

Expertise in a few powerful methods does not mean you can solve any problem; diversity counts

Even the preference for simpler classifiers does not have theoretical foundation

The empirical success of Occam's razor is due

- to the problems we are usually solving
- to our design approach – increase complexity only if the previous approach does not work

What Is Similarity?

Similar theorem (Ugly Duckling theorem) exists for patterns and attributes:

- There is no general reason to prefer one feature or dimension over the other and one distance metrics over the other
- In the absence of assumptions even the notion of similarity does not make sense

Error and Bias-Variance Trade-off

In regression mean square error is a reasonable estimate of performance

MSE is additive in terms of bias squared and variance

In classification MSE is not so good due to discrete decisions

Boundary error in decision space can still be expressed I terms of bios and variance, but the relationship is multiplicative

The nature of this relationship makes variance to dominate the bias

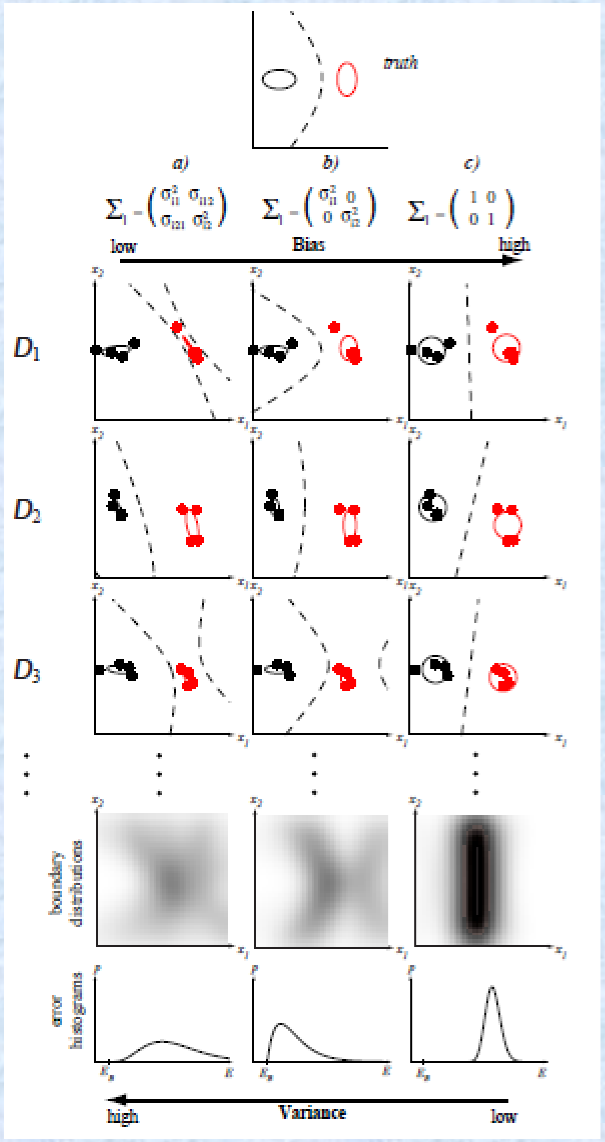
Recall that large bias means poor match and large variance means weak match

So we should favor a system with lower variance despite higher bias – better be sure even if wrong

Error and Bias-Variance Trade-off

- More parameters – lower bias but higher variance
- More data points for a given bias – lower variance
- As the size of the training set approaches infinity the error approaches its theoretical limit
- But at the same time we need more parameters to fit the data

- The only way to get zero bias and zero variance is to know the true data distribution ahead of time



Resampling

How can we estimate bios and variance without infinite data sets?

Previous slide suggests that we can train a system on different data samples and look at the resulting decision boundaries

Then if we have fewer parameters we must have larger bios, and if we have a tighter distribution of decision boundaries from different samples we must have a lower variance

Furthermore, analysis of these boundaries and possibly their combination can improve the resulting performance

Resampling

Resampling – choosing data subsets; e.g. to extract relevant information without overfitting

- How to use a given training set to maximize the performance?
- Which methods improve performance across many classifiers?
- Are there rules to estimate which methods are likely to improve a given system?

E.g. does the order of presentation matter?

- systems based on category boxes with fast learning depend critically on this order, so train multiple times with different orders and vote or take average

Resampling Methods

Bootstrapping – assigning measure of accuracy by using multiple resamples of the original data set (and equal in size to the original set) by drawing from it randomly with replacement

Jackknifing – similar approach when measure of accuracy are computed from multiple resamples of the original set which are created by leaving out one or more data points

Permutation tests – shuffle the class labels in the data and compare the original class prediction distribution with the shuffled one, can give an estimate if the current properties actually allow a reasonable classification

Cross-validation – use different subsets of training data for validation

Jackknifing

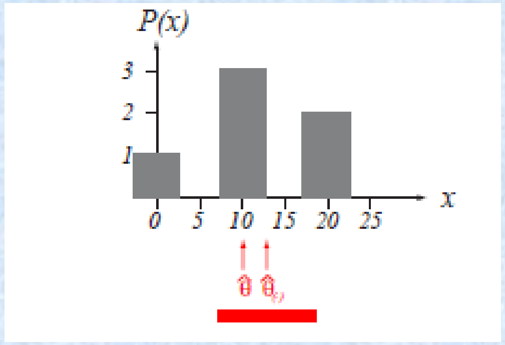
The key point is that the variance is hard to compute for any other statistic than the mean if you use a whole set

If we have a set of size *n* and use it *n* times, each time leaving one element out, then we have *n* estimates of any statistic

Then we can apply common formulas for mean and standard deviation to these *n* estimates

As a result we get not only a value for any statistic (mode, median, etc.) but also an estimate of error for it

Furthermore, sometimes the value can be "better" then the traditional value from the full set



Bootstrapping

Due to the sampling there will be repetitions in each bootstrap set

Instead of getting one output we get multiple outputs and can even gather statistics of these outputs

Implicit assumption is that data points are mutually independent

Increasing the number of bootstrap samples reduces the errors inherent in the bootstrapping procedure, so the more the better

In the limit of infinite samples the bootstrap statistics approach true statistics

On the other hand you can stop at any time, unlike jackknifing that requires exactly n runs

Permutation Tests

Similar to the techniques we discussed in 510 for detecting synchrony and oscillatory patterns

Shuffling class labels randomly allows you to create a “baseline” for a given data set

Permutation tests are a subset of exact tests in statistics that help to verify the significance of the result

In our case the hypothesis is that a given data set can be classified in distinct classes using a given classifier

Fisher’s exact test is one of the commonly used permutation tests

Bootstrap Aggregating (BAGGing)

Use multiple classifiers (usually of the same type) – component classifiers

Create *m* training sets of size $n' \leq n$ by sampling with replacement

Train *m* systems, one on each set

When testing: vote (for classification) or average (for regression) between these systems

Bagging can help to pinpoint "unstable" classifiers – ones that develop drastically different decision boundaries due to small differences in training sets or presentation order

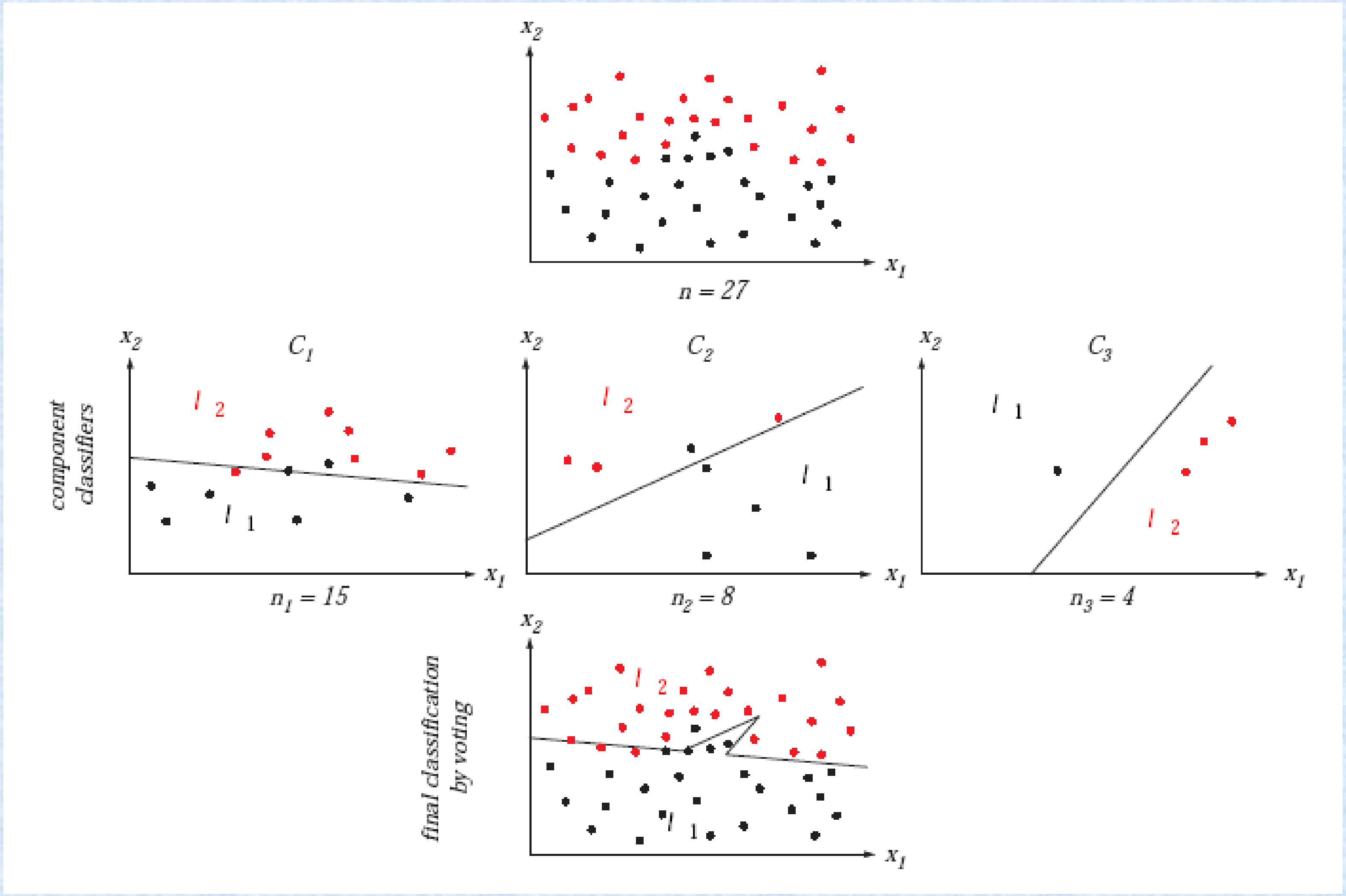
Boosting

Uses multiple classifiers, but this time data selection is not fully random

1. Select the first training subset randomly with equal probabilities without replacement
2. Train the classifier
3. See which data points are classified correctly, decrease their chance of being selected, and increase the chances of misclassified points
4. Select the next training subset using these sampling rules and train the next classifier
5. Repeat 3-4 until criteria is reached

Each next classifier pays more attention to misclassified points

Example Boosting from DH&S



Major decision – size of the subset used for training

Boosting and No Free Lunch

On one hand each added component classifier in boosting decreases the training error

In fact the error goes down exponentially with the number of components

But this is training error, not the generalization error

Furthermore, it only works if component classifiers can perform better than chance on a given problem, which according to NFL theorem is not necessarily true

Voting Strategies

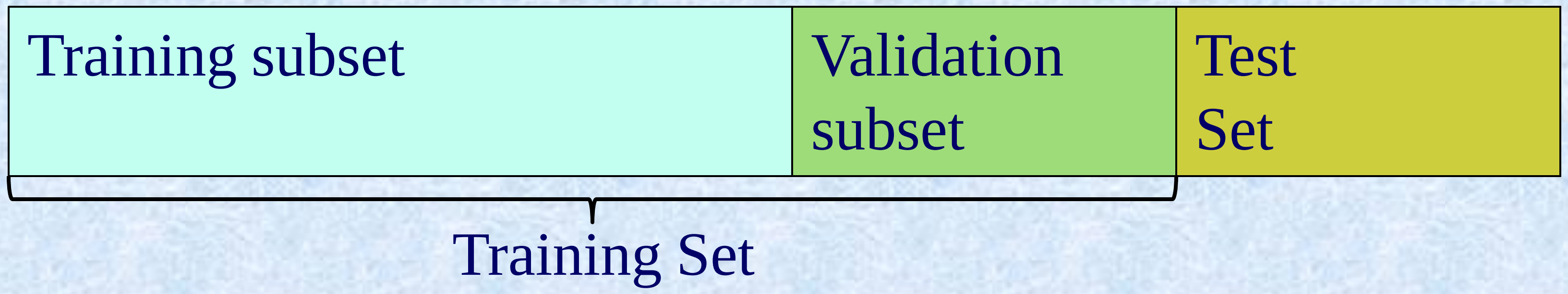
Direct voting: all voters are equal

- Good if classifiers are identical and trained on similar data sets

Mixture models

- Weight the voters: good if using different algorithms or boosting
- Can adapt classifiers and weights separately or simultaneously

Validation



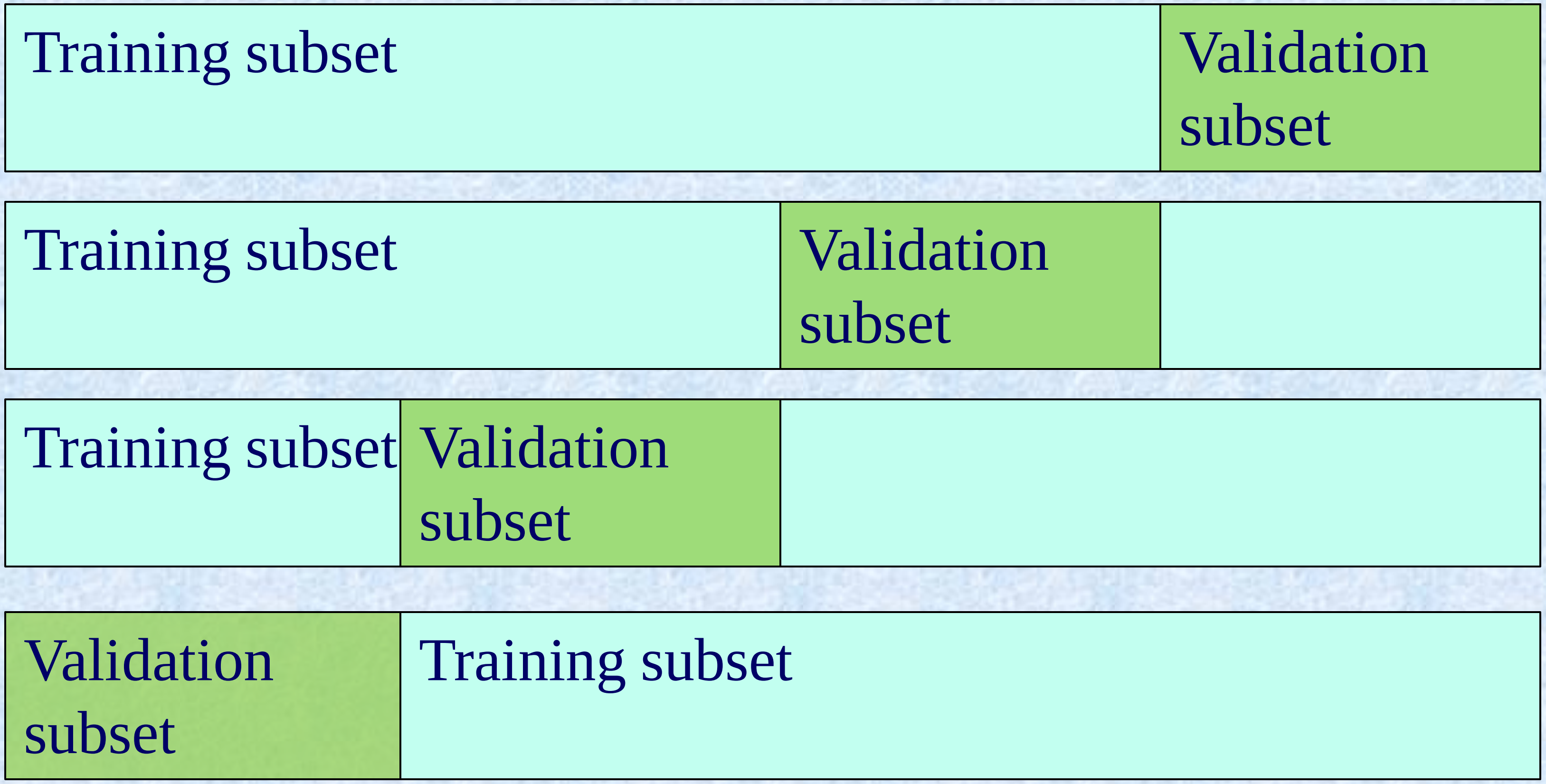
For empirical choice:

- Split the training set into two – for training and for validation
- Train on the first subset, test on the second subset

For better results do it multiple times with different splits

After parameters are chosen retrain the system on complete training set before testing

N-fold Cross Validation



Divide training set into N equal subsets

Use each of subsets as a validation set

Compare results

Choosing N

The more parameters the classifier has, the larger portion of the data is needed for training – decrease the validation set size and increase N

If the number of parameters is small comparing to the number of training points, then N does not matter much

N=10 is a usual default, but you might want odd number in case of voting on two-class problem

Note that if N is equal to the training set size, then we get jackknifing effectively leaving one item out for each of N runs

N>30 would be a good choice for combining cross-validation and jackknifing statistical analysis

Note that jackknifing here will give you stats on validation points too

Evaluation Criteria: Accuracy

Simplest measure - % correct

- Works fine on problems with approximately equal output class ratio
- Fails on real life data with one dominating class (e.g. medical databases)

Let’s say 95% of inputs give negative result, while 5% give positive and require treatment

Classifier that gives all negative output is 95% correct but totally useless

Need measure that is independent of the output class mix

Evaluation Criteria: Confusion Matrix

Take numbers of correct and incorrect and put them into table

	Actual positive	**Actual negative**	**Total**
Predicted positive	37 *True positive*	4 *False positive*	41
Predicted negative	11 *False negative*	48 *True negative*	59
Total	48	52	100

For medical classifier we might tune the system to minimize false negatives (missing a case requiring treatment is more dangerous then anything else)

Use miss rate: false negative/actual positive – 11/48=23% and minimize it

Evaluation Criteria: Confusion Matrix

	Actual positive	**Actual negative**	**Total**
Predicted positive	37 *True positive*	4 *False positive*	41
Predicted negative	11 *False negative*	48 *True negative*	59
Total	48	52	100

True positive – hit

True negative – correct rejection

False positive – false alarm, Type I error

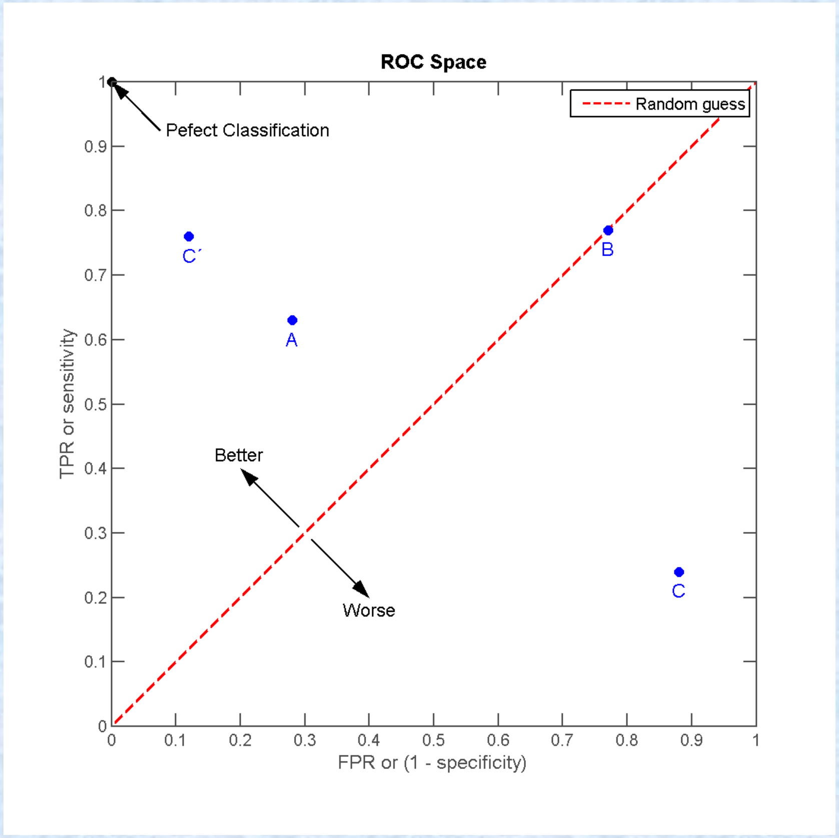
False negative – miss, Type II error

Sensitivity, true positive rate (TPR) – hit rate, recall – TPR/P

False positive rate (FPR) – fall-out – FPR/N

Confusion Matrix Definitions and ROC Space

- Use FPR and TPR as axes
- Receiver (or relative) operating characteristic
- Random guess is on diagonal
- Perfect classifier is in the upper left corner
- Note that the perfectly bad classifier in the lower right corner is also good
- Example – Pierre Richard's character in La Chevre – the guy who is always wrong

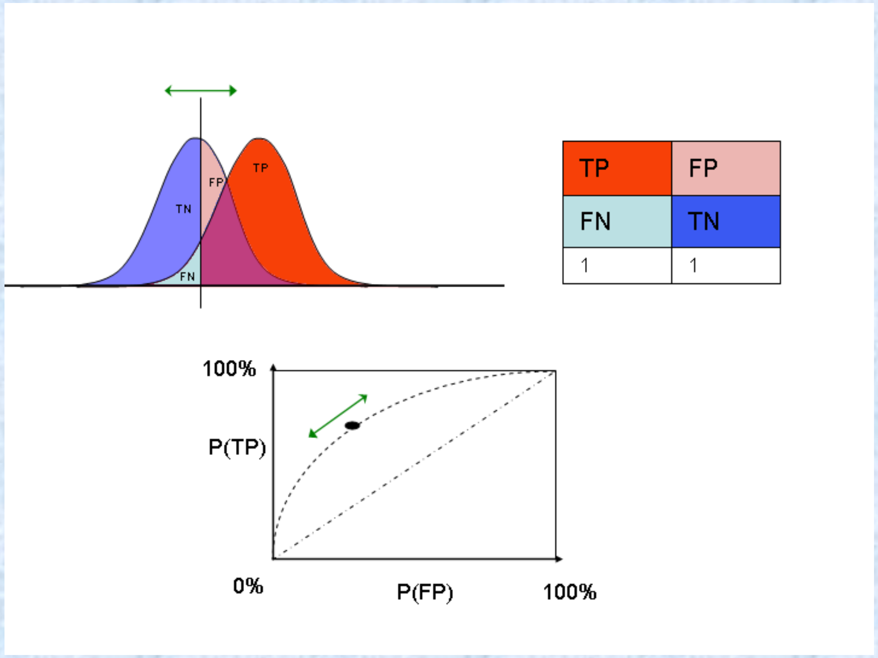


ROC Curves

Often the parameters of the classifier can be adjusted to affect the result

Simple example – threshold

As a result one can build a curve in the ROC space



Using this curve some other statistics can be derived

Often used for model comparison is area under the curve (AUC, A’, c-index)

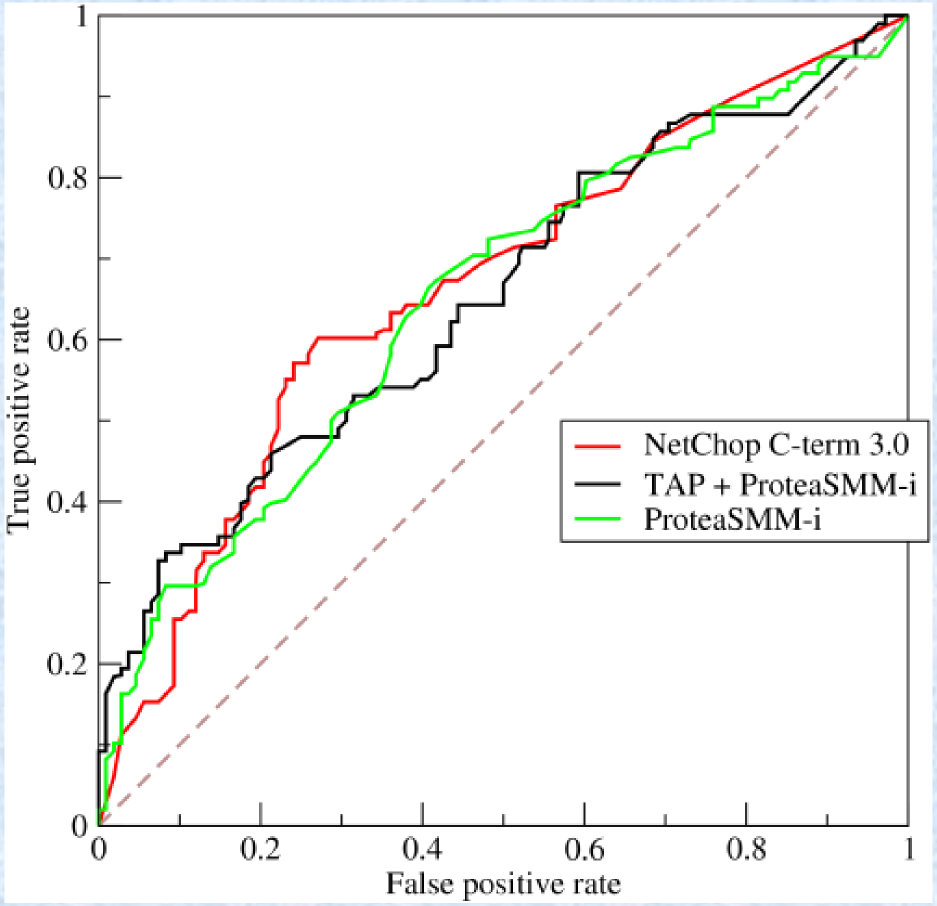
C-index corresponds to a probability that positive sample will get higher class value than negative sample

ROC Curves

Intuitive comparison:

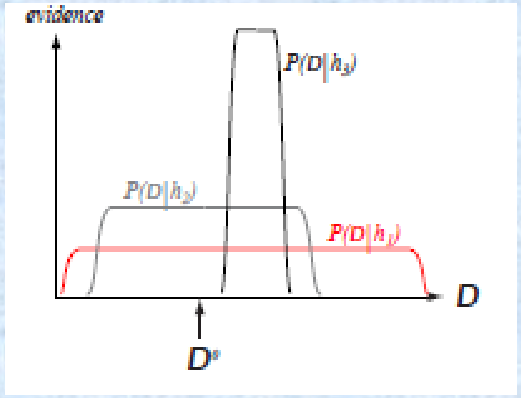
- Red curve gives the overall best classification
- Black curve might be better choice if we want to minimize false positives

C-index=1 means that a classifier for some choice of parameters finds all positive cases without false alarms



In this case the curve degenerates into a step function that jumps to TPR of 1 at FPR of 0

Maximum Likelihood and Bayesian Comparison



Find the maximum likelihood of parameters for each classifier

Calculate and compare the likelihoods

$$p(h|D) \propto p(D|h)$$

Bayesian comparison uses full analysis including the priors

The assumptions about these priors are critical here:

If you make correct assumptions, you can choose a better classifier

If you use wrong assumptions – no luck

Thus No Free Lunch still holds

Add-ons: Preprocessing

Preprocessing of input vectors – principal component analysis, dimensional reduction

Goal is to reduce the dimensionality of the input without loosing information

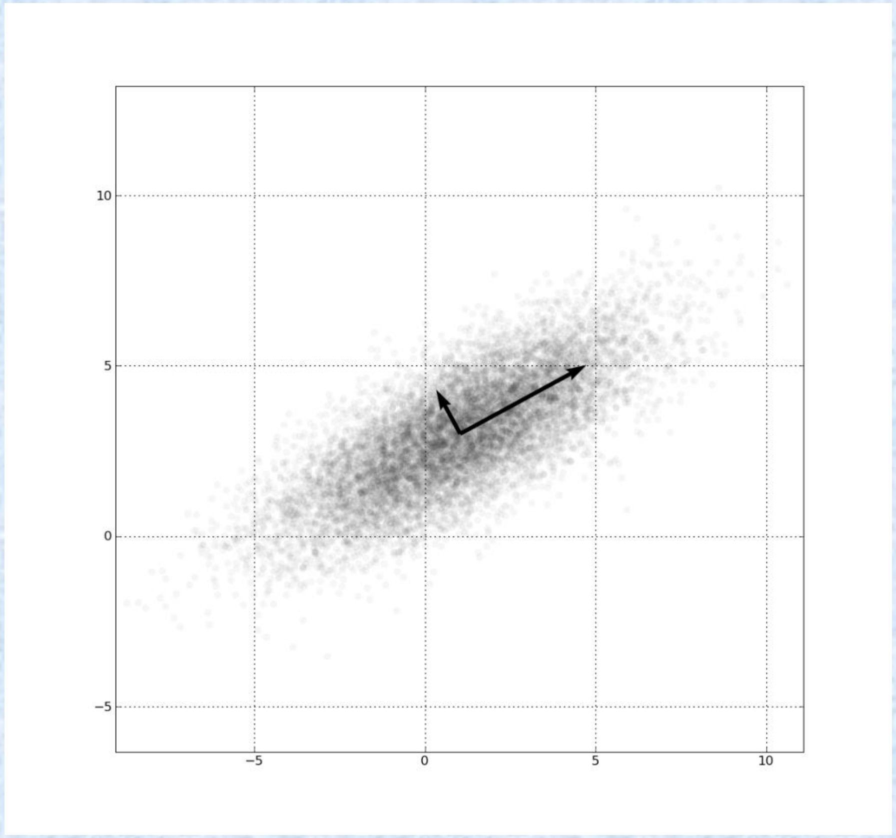
Component analysis – finding the right set of input features

Goal of PCA is to select most important components and drop the least important reducing the dimensionality of the problem

Goal of ICA is to find data dimensions that are independent

Principal Component Analysis (PCA)

- Linear transformation – new axes coalligned with data distribution
- Converts possibly correlated data to a set of uncorrelated variables
- To reduce dimensionality of the data even further – pick only the largest PC and base axes on these
- Each next PC has the highest variance given its orthogonality



PCA is the simplest of true eigenvector-based multivariate analyses

Principal Component Analysis (PCA)

PCA is dependent on the data scaling

Find the covariance matrix of the data

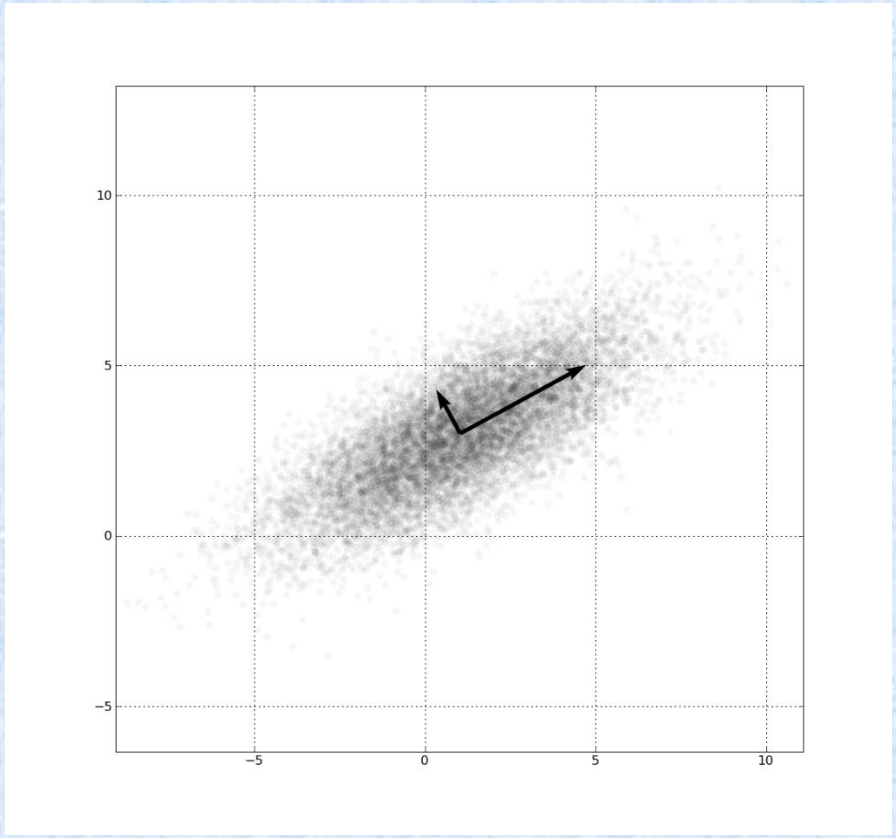
Calculate the eigenvalues of this matrix

Sort the eigenvectors by decreasing eigenvalue

Pick the number of basis eigenvectors

Convert data to z-scores

Project it on the new basis



PCA is task independent and can disregard small features even if they are important

Independent Component Analysis

If the data is noisy it is possible for PCA to pick up trends in noise rather than the data

ICA (given that categories are indeed independent) will pick up trends that lead to the better category separation even if these have small eigenvalues