Statistics for Genomic Data Analysis

http://moodle.epfl.ch/course/view.php?id=15271

Lecture 7

Outline

1 Introduction to classification

2 Classification methods

3 Comparison of methods using Iris data

4 Tree construction

5 Estimating prediction accuracy

Classification

- Historically, *objects* classified into *groups*
  - periodic table of the elements (chemistry)
  - taxonomy (zoology, botany)
- Why classify?
  - organizational convenience, summary
  - prediction
  - explanation
- *Note*: different aims may lead to different classifications; *e.g.* *SIZE* of object vs. its *USE*
- Classification *divides objects into groups* based on a set of values

Classification task

- *Task:* assign objects to classes (groups) on the basis of measurements made on the objects
- *Unsupervised:* classes unknown, want to discover them from the data (cluster analysis)
- *Supervised:* classes are predefined, want to use a (training or learning) set of labeled objects to form a classifier for classification of future observations

Example: tumor classification

- Reliable and precise classification essential for successful cancer treatment
- Current methods for classifying human malignancies rely on a variety of morphological, clinical and molecular variables
- Uncertainties in diagnosis remain; likely that existing classes are heterogeneous
- Characterize molecular variations among tumors by monitoring gene expression (microarray)
- Hope: that microarrays will lead to more reliable tumor classification (and therefore more appropriate treatments and better outcomes)
- There have been some successes in this area

Discrimination

- Objects (*e.g.* samples) are to be classified as belonging to one of a number of *predefined classes* $\{1, 2, \ldots, K\}$
- Each object associated with a *class label* (or *response*) $Y \in \{1, 2, \ldots, K\}$ and a *feature vector* (vector of predictor variables) of p measurements: $X = (X_1, \ldots, X_G)$
- *Aim:* predict Y from $\boldsymbol{X}$

Classifiers

- A *predictor* or *classifier* partitions (divides) the variable space *e.g.* gene expression profiles) into K disjoint subsets, $A_1, \ldots, A_K$, such that for a sample with expression profile $\boldsymbol{X} = (X_1, \ldots, X_p)$ in A_k the predicted class is k
- Classifiers are from a *learning set (LS)* (or *training set*) $L = (\boldsymbol{X_1}, Y_1), \ldots, (\boldsymbol{X_n}, Y_n)$
- Classifier C built from a learning set L:

$$C(\cdot, L) : \boldsymbol{X} \to \{1, 2, ..., K\}$$

- *Predicted class* for observation $\boldsymbol{X}$:

$$C(X, L) = k \text{ if } \boldsymbol{X} \text{ is in } A_k$$

Some classification methods

- (Fisher) Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA), Diagonal Discriminant Analysis (DDA)
- k-nearest neighbors (knn)
- Support Vector Machine (SVM)
- Classification trees (CART)
- Random Forests

Fisher Linear Discriminant Analysis (LDA)

First applied in 1935 by M. Barnard at the suggestion of R. A. Fisher, *linear discriminant analysis (LDA)*:

1. finds *linear combinations* of the variables $\boldsymbol{X} = X_1, \ldots, X_p$ with large ratios of between-groups to within-groups sums of squares – these are the *discriminant variables*;
2. predicts the class of an observation $\boldsymbol{X}$ by the class whose mean vector is closest to $\boldsymbol{X}$ in terms of the discriminant variables

Quadratic and Diagonal Discriminant Analysis

- LDA is derived assuming that within each class, $\boldsymbol{X}$ has a multivariate normal distribution
- In LDA assume covariance matrices in both classes are the same: $\Sigma_1 = \Sigma_2 = \Sigma$
- In *quadratic discriminant analysis (QDA)*, the covariance matrices can be *different*
- In *diagonal discriminant analysis (DDA)*, the covariance matrices are *diagonal*
- When the covariance matrices are diagonal and equal, the discriminant rule is *linear* (variables are independent)

Linear and Quadratic Discriminant Analysis

- Classical and widely used tools for classification
- Simple, *intuitive*: the predicted class for a test sample is the class with the closest mean (using Mahalanobis distance, which is scale-invariant and takes into account correlations)
- *Optimal* when the model assumptions are true
- Computationally simple
- Often has good performance in practics

Linear and Quadratic Discriminant Analysis – drawbacks

- Linear or quadratic discriminant boundaries might not be sufficiently *flexible*
- Distributional assumptions may not hold, thereby degrading performance
- Performance may also degrade in the case of too many features:
    - overfitting (more on this below)
    - highly variable parameter estimates

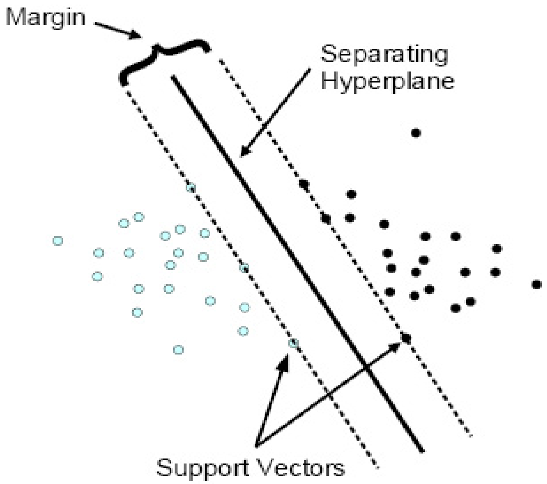
k-nearest neighbors

- Based on a measure of *distance* between observations (*e.g.* Euclidean distance)
- *k*-nearest neighbor rule classifies observation $\boldsymbol{X}$ as follows:
  - find the *k* observations in the learning set *closest* to $\boldsymbol{X}$
  - predict the class of $\boldsymbol{X}$ by *majority vote* (*i.e.*, choose the class that is *most common* among those *k* observations)
- The number of neighbors *k* can be chosen by *cross-validation* (more on this later)
- Important issues: choice of distance, selection of *relevant* features

Support Vector Machine (SVM)

- *Class separation:* look for optimal separating hyperplane between two classes by maximizing the *margin* between the classes' closest points
  - points lying on the boundaries are the *support vectors*; middle of the margin is the optimal separating hyperplane;
- *Overlapping classes:* data points on the "wrong" side of the discriminant margin are downweighted to reduce their influence ("soft margin")
- *Nonlinearity:* when no linear separator, data points projected into a (usually) higher-dimensional space where the data points effectively become linearly separable (projection realized via *kernel techniques*)
- *Find solution:* the whole task can be formulated as a quadratic optimization problem, which can be solved by known techniques

Support Vector Machine (SVM)



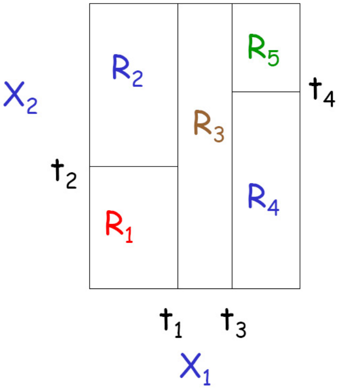
Trees

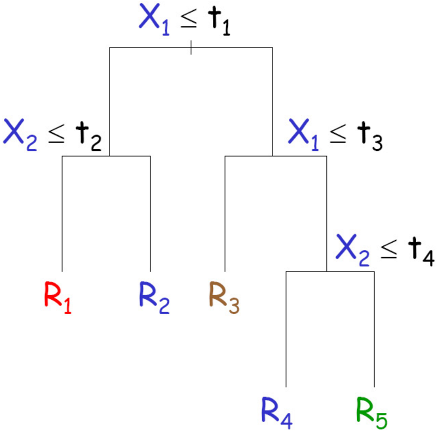
- Provide means to *express knowledge*
- Can aid in decision making
- Can be portrayed *graphically* or by means of a chart
- Response types:
  - Categorical $\Rightarrow$ Classification tree
  - Continuous $\Rightarrow$ Regression tree
  - Survival $\Rightarrow$ Survival tree
- Available `R` packages include `tree`, `rpart`, `tssa`

Classification trees (CART)

- Partition the feature space into a set of rectangles, then fit a simple model in each one
- *Binary tree structured classifiers* are constructed by repeated splits of subsets (nodes) of the measurement space $\underline{X}$ into two descendant subsets (starting with $\underline{X}$ itself)
- Each terminal subset is assigned a class label; the resulting partition of $\underline{X}$ corresponds to the classifier

CART: partition representation

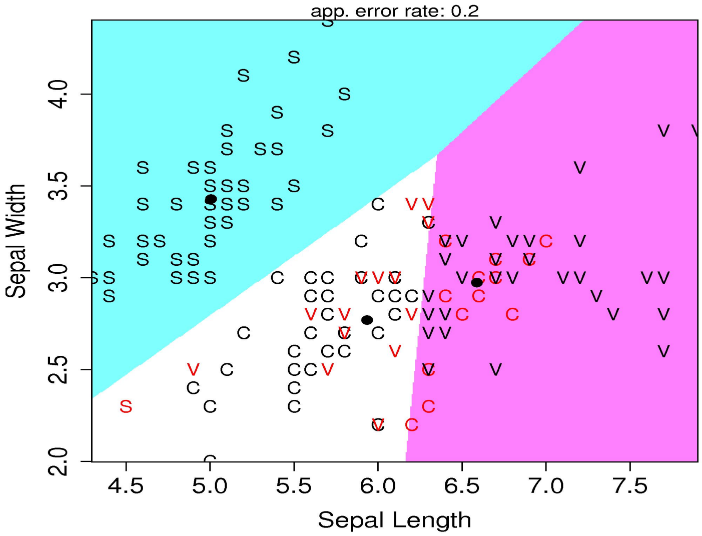




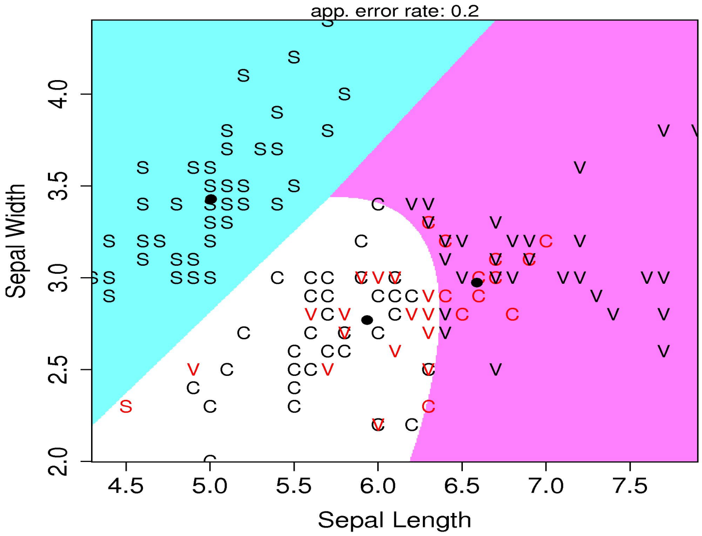
Iris data

- This classic data set gives measurements (cm) of 4 variables for 50 flowers from each of 3 species of iris:
  - sepal length
  - sepal width
  - petal length
  - petal width
- The species are *Iris setosa, versicolor and virginica*
- The data were collected by Edgar Anderson (1935)
- We will use the `iris` data to look at some of these methods

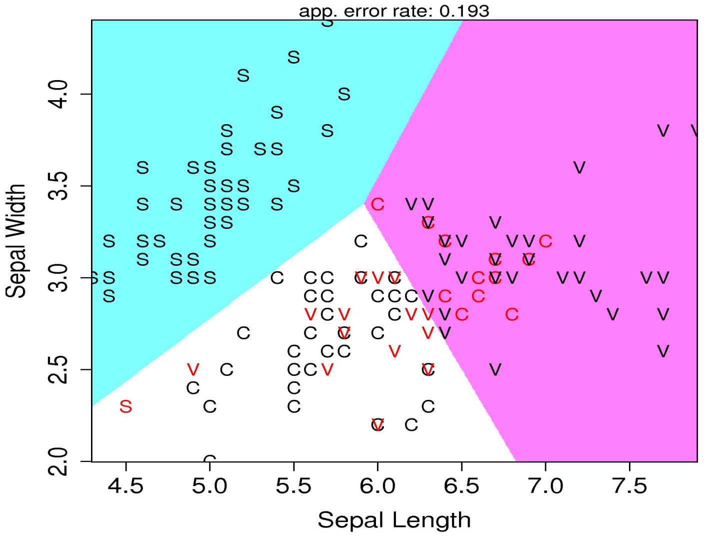
Iris data: LDA



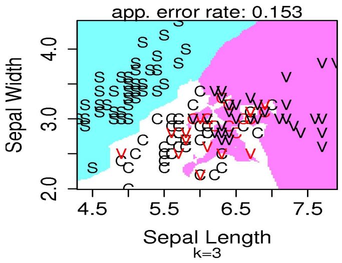
Iris data: QDA

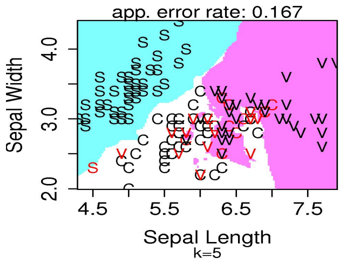


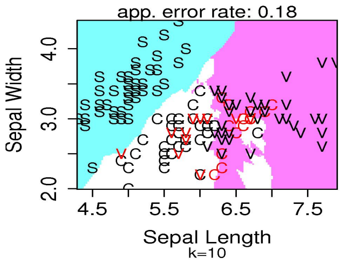
Iris data: DDA

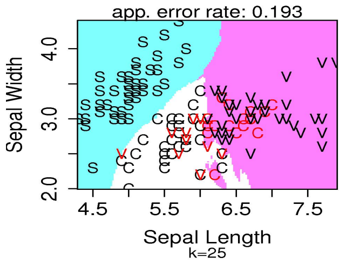


Iris data: knn

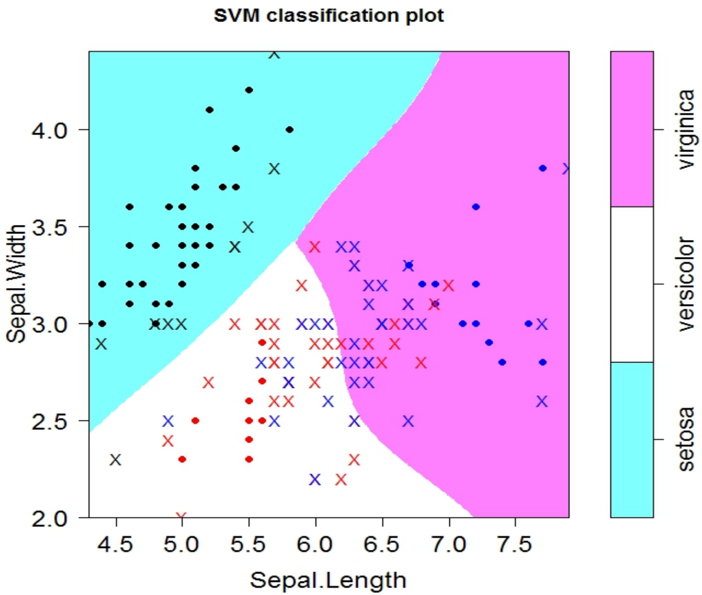




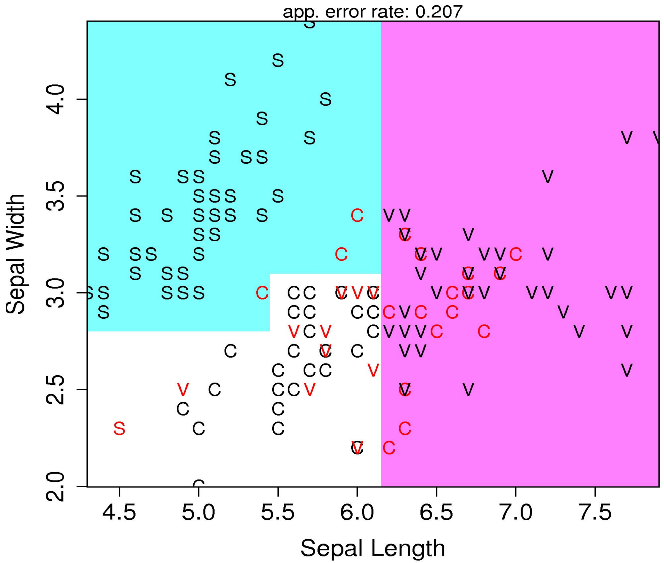




Iris data: SVM



Iris data: CART



BREAK

Three Aspects of Tree Construction

- Split selection rule
  - *Binary* splits – look only *one step ahead*
  - *Impurity measure* (Gini index or entropy) to optimize split
- Split-stopping rule
  - *Issue:* A very large tree will tend to *overfit* the data; too small a tree might not capture important structure
  - *Usual solution:* grow *large* tree then do *pruning*
- Assignment of predicted values
  - (weighted) Voting among observations in the node

Feature selection

- Feature selection is an extremely important issue in classification
- Particularly relevant for microarray data containing thousands of features – most of which will not be useful for classification
- Feature selection is automatic with trees
- For DA, NN need preliminary selection
- SVM tends to perform better with preliminary selection
- Need to account for feature selection when assessing performance to avoid *bias*
- Missing data
  - Automatic imputation with trees
  - Otherwise, impute (or ignore)

Polytomous classification

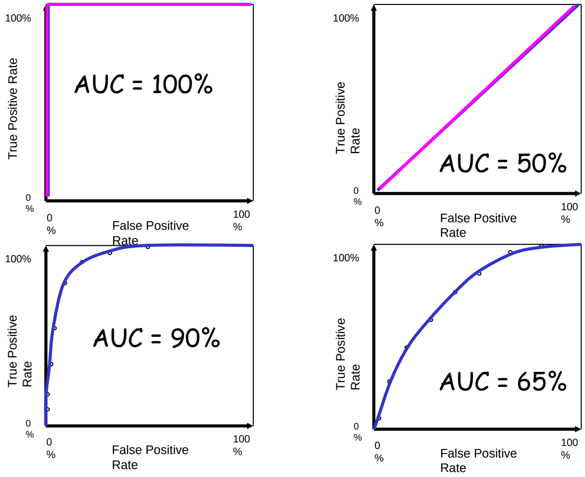
- Many classifiers only work for *binary* (two-class) problems
- It might be otherwise advantageous to convert a K-class problem into a series of binary problems – *e.g.* large number of unequally represented classes in the learning set
- Two possibilities:
  - *All pairwise binary classification:* final predicted class is the class predicted most often in the $\binom{K}{2}$ individual classifications
  - *One-versus-all binary classification:* binary classification K times; final predicted class is the class with the highest estimated posterior probability

Performance assessment I

Often used in assessing classifiers in biomedical studies:

- *Brier score:* Measures accuracy of a set of probability assessments
  Problem: Cannot be used for all classifier types
- *Area under the (ROC) curve (AUC):* Receiver operating characteristic (ROC) curve plots *sensitivity* (true positive rate) vs. false positive rate for a *binary classifier* as the threshold varies; *area under the ROC curve (AUC)* provides an overall measure of classifier performance
  Useful for *comparing* classifier performance

AUC for ROC curves



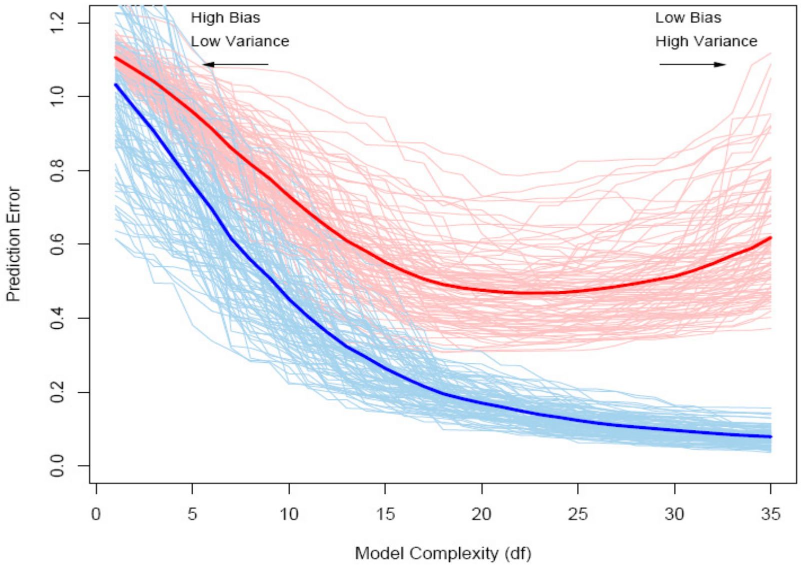
Performance assessment II

- *Resubstitution estimation:* Error rate on the learning set
  Problem: downward bias
- *Test set estimation:* Divide cases in learning set into two sets, L_1 and L_2; classifier built using L_1, error rate computed for L_2 (L_1 and L_2 must be iid)
  Problem: reduced effective sample size

Overfitting

- As a classifier becomes more and more *complex*
  - it can adapt to more complicated underlying structures (decreased bias)
  - resubstitution error rate increases (increased variance)
- More parameters ⇒ More complex
- Want to find an *optimal complexity* with minimum test error
- Training (or learning) error *decreases* with complexity, and can even drop to 0 for sufficiently high complexity
- Highly complex classifiers can be modeling not just the signal in the data, but also the *noise*
- If the error in the learning set is very low, the model is probably *overfit*
- *Overfit classifiers will typically generalize poorly*, so you are unlikely to get accurate predictions for new observations

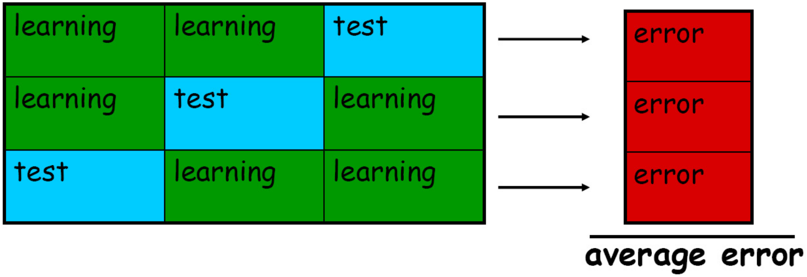
Overfitting



Performance assessment III

- *V-fold cross-validation (CV) estimation:* Cases in learning set randomly divided into V subsets of (nearly) equal size. Build classifiers leaving one set out; test set error rates computed on left out set and averaged.
  Bias-variance tradeoff: smaller V can give larger bias but smaller variance
- *Leave-one-out cross-validation (LOOCV):* special case of CV with $V = n$
  - low bias but high variance error rate estimates
  - provides good estimates for *stable* (low variance) classifiers (*e.g.* k-nn)
  - computationally intensive for large n
- *Out-of-bag (oob) estimation:* covered below
- 0.632 *and* 0.632^+ *estimation:* covered below

Cross-validation



How **NOT** to estimate error

**** DON'T DO THIS ** DON'T DO THIS ****

- Use the whole data set to choose which variables (features) to use in the classifier
- Divide the data into (10, say) subsets for CV
- Leave out a subset and build a classifier with features chosen from the whole data set
- Use the classifier to predict the left out subset
- Average over left out subsets to estimate error

**** DON'T DO THIS ** DON'T DO THIS ****

Aggregating classifiers

- Breiman (1996, 1998) found that gains in accuracy could be obtained by *aggregating predictors* built from "perturbed versions" of the learning set
- The multiple versions of the predictor are aggregated by voting
- Let $C(\cdot, L_b)$ denote the *classifier* built from the b^{th} perturbed learning set L_b, and let w_b denote the *weight* given to predictions made by this classifier
- The predicted class for an observation $\boldsymbol{x}$ is given by

$$argmax_k \sum_b w_b I(C(\boldsymbol{x}, L_b) = k)$$

Bagging

- Bagging = Bootstrap aggregating
- *Nonparametric Bootstrap (standard bagging):* perturbed learning sets drawn at random with replacement from the learning set; predictors built for each perturbed dataset and aggregated by plurality voting ($w_b = 1$)
- *Parametric Bootstrap:* perturbed learning sets specified by parametric model (*e.g.* multivariate normal)
- Convex pseudo-data (Breiman 1996)

Out-of-bag (oob) error rate estimation

- Out-of-bag error rate estimate: unbiased
- Use the *left out cases* from each bootstrap sample as a test set
- Classify these test set cases, and compare to the class labels of the learning set to get the *out-of-bag estimate* of the error rate

0.632 and 0.632$^+$ estimation

- Proposal to reduce the upward bias of the bootstrap
- Method uses a weighted combination of bootstrap error estimate (weight 0.632) and resubstitution estimate (weight 0.368)
- Further refinement produces the 0.632$^+$ estimate, which increases the weight of the bootstrap estimate when the resubstitution error is small; seems particularly appropriate in the case of overfitting

Boosting

- Freund and Schapire (1997), Breiman (1998)
- Data resampled *adaptively* so that the weights in the resampling are increased for those cases most often misclassified
- Predictor aggregation done by weighted voting

Random Forests

- *Random Forests* is a classification method based on classification trees
- In Random Forests, *many* classification trees are grown (without pruning) based on randomly selected variables using bootstrapped samples from the original data
- To classify a new object from an input vector, put the input vector down *each of the trees* in the "forest"
- Each tree gives a classification (the tree "votes" for a class)
- The Random Forest classifier chooses the classification having the most votes (over all the trees in the forest)

Classification of samples using microarray data

- In the microarray context, confronted with "small n, large p" problem: the number of features/variables p is high, in the tens of thousands, but the number of samples n is small, usually not more than a few hundred
- Three approaches to deal with the $n \ll p$ setting:
  1. *variable selection*, for example using univariate statistical tests
  2. *regularization or shrinkage methods*, such as the Support Vector Machine, penalized regression methods or boosting (some also perform variable selection)
  3. *dimension reduction* or feature extraction, often using partial least squares

Books on classification

- Maindonald and Braun. *Data Analysis and Graphics Using R*
- Hastie, Tibshirani, Friedman. *The Elements of Statistical Learning*
- Venables and Ripley. *Modern Applied Statistics with S-Plus (MASS)*

R packages

http://cran.r-project.org/web/views/MachineLearning.html

- MASS: lda, qda
- sda: LDA, DDA
- class: knn
- rpart: classification and regression trees (recursive partitioning)
- ipred: bagging
- e1071: SVM
- LogitBoost, mboost: boosting
- randomForest: trees with bagging
- MLinterfaces, CMA: wrapper for multiple methods

In conclusion: A philosophical note

From Leo Breiman and Adele Cutler, creators of Random Forests (RF):

> *RF is an example of a tool that is useful in doing analyses of scientific data.*
>
> *But the cleverest algorithms are no substitute for human intelligence and knowledge of the data in the problem.*
>
> *Take the output of random forests not as absolute truth, but as smart computer generated guesses that may be helpful in leading to a deeper understanding of the problem.*

This statement could apply to any classifier.