

Supervised Classification II

8. Logistic Classification

Idea: Abandon the linear regression model and try a more flexible model of posterior probabilities.

For the so-called logit function we get a logistic model in which log-odds are linear:

Assuming again $g = 2$ (and classes coded as 1 and 2):

$$\log \frac{p(2|\mathbf{x})}{1 - p(2|\mathbf{x})} = \alpha + \beta^T \mathbf{x},$$

with the inverse

$$p(2|\mathbf{x}) = \frac{\exp(\alpha + \beta^T \mathbf{x})}{1 + \exp(\alpha + \beta^T \mathbf{x})}$$

$(\log(v/(1 - v)))$ is called the logit function and is denoted by $\text{logit}(v)$.

Remark. Different transformations of $p(2|x)$ are often used.

The parameters in the logistic model can be estimated (iteratively!) by maximizing the likelihood function

$$\prod_{i=1}^n p(2|\mathbf{x}_i)^{y_i} p(1|\mathbf{x}_i)^{1-y_i},$$

where y_i is the value of the indicator function for class 2 for the i th object.

Classification is done by using the empirical Bayes rule, i.e.:

allocate \mathbf{x} to a class 1 if $\hat{p}(1|\mathbf{x}) \geq \hat{p}(2|\mathbf{x})$

The approach easily generalizes to the case with more than two classes (it is another matter that the greater is g , the more involved the estimation process becomes). We assume

$$\Theta(k|\mathbf{x}) \equiv \log \frac{p(k|\mathbf{x})}{p(1|\mathbf{x})} = \alpha_k + \beta_k^T \mathbf{x},$$

for $k = 2, \dots, g$, with the inverses

$$p(k|\mathbf{x}) = \frac{\exp \Theta(k|\mathbf{x})}{\exp \Theta(1|\mathbf{x}) + \dots + \exp \Theta(g|\mathbf{x})}$$

with $\Theta(1|\mathbf{x}) = 0$.

Classification is done by using the empirical Bayes rule, i.e.:

allocate \mathbf{x} to class k for which the value of $\hat{p}(k|\mathbf{x})$ is maximal.

Remark. Both LDA (but not Fisher's LDA) and logistic classification model imply that

$$\log p(k|\mathbf{x})/p(1|\mathbf{x})$$

is a linear function of the predictors. Thus both methods yield a **linear** classification boundary. The difference between the methods:

- Logistic classification uses conditional likelihood to estimate β and α - no information on distribution of X is used.
- LDA uses the normality assumption: $(X|Y = k) \sim N(\mu_k, \Sigma)$.

Logistic discrimination is more appropriate when distribution of X significantly differs from normal or class covariances are significantly different.

Remark. We skip discussion of diagnostics of the logistic model.

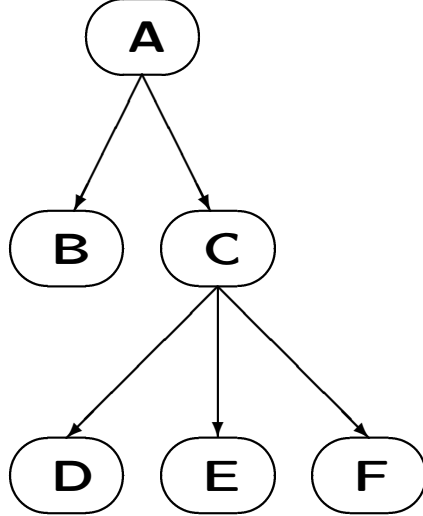
9. Classification or decision trees

Classification trees yield a division of the feature space into hyperrectangles (with edges parallel to the axes) - on each of the hyperrectangles a unique decision is reached.

For quantitative and ordinal attributes, hypercubes are formed by conjunction of conditions of the form $\{x_i \geq b\}$ or $\{x_i \leq b\}$, where x_i is a chosen attribute and b is a chosen threshold. For nominal attributes, the story is different (to be explained).

The process of division of the feature space into rectangles may be represented by a tree.

Tree - undirected, acyclic and connected graph. Classification tree – a directed graph with a single root node (cf. next page), which is a tree if considered as an undirected graph. We shall confine ourselves to binary trees.



Vertices of a tree - its nodes

A - root of the tree

B, C -children of parent A,

B, C, D, E, F -descendants of A,

B, D, E, F - leaves of the tree.

Root - no edge enters this node, a leaf - no edge leaves it. In a binary tree, two edges leave each node except for leaves.

In each node logical condition $\{x_i \geq b\}$ or $\{x_i \leq b\}$ or $\{x_i = b\}$ is located. A node m defines a rectangle $\mathcal{R}_m \subset \mathcal{X}$ pertaining to *all* conditions along the path from the root to this node.

Condition satisfied \implies move to left

Condition **not** satisfied \implies move to right

Condition (i.e. a chosen attribute and threshold b) changes from node to node.

Decision rule based on a constructed tree:

- Move a new element x down the tree. Find a leaf at which this element ends up.
- Classify x to the class to which most of the elements of 'its' leaf belong.

Partition rules: CART (Breiman *et al* (1984))

Quality of a tree depends on conditions employed to split data at each node. Our (first) objective: end up with a tree such that its leaves are as 'pure' as possible. We need:

- measure of node impurity;
- measure of change of impurity from level to level

Measures of node impurity

Consider node m defined by a rectangle $\mathcal{R}_m \subset \mathcal{X}$.

Proportion of class k observations in node m

$$\hat{p}_{m,k} = \frac{1}{n_m} \sum_{\mathbf{x}_i \in \mathcal{R}_m} I(y_i = k) = \frac{n_{m,k}}{n_m},$$

$n_{m,k}$ - number of observations of class k in node m ;

n_m - number of all observations in node m ;

A reasonable measure of impurity should be 0 when $n_{m,k} = n_m$ for some m , and should be maximal when all $n_{m,k}$ are the same.

Two such measures:

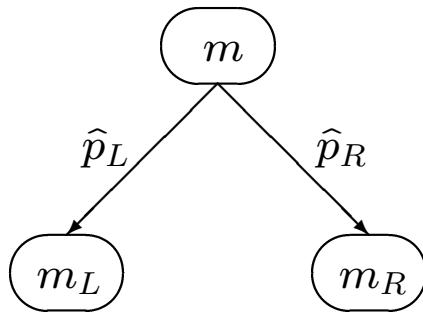
- $Q_m(T) = \sum_{k=1}^g \hat{p}_{m,k} (1 - \hat{p}_{m,k})$ (**Gini index**)
- $Q_m(T) = - \sum_{k=1}^g \hat{p}_{m,k} \log \hat{p}_{m,k}$ (**entropy**)

Measure of change of impurity from level to level

Let \hat{p}_L be the proportion in node m of elements which moved to its left child m_L

$$\hat{p}_L = \frac{n_{m_L}}{n_m},$$

where n_{m_L} is the number of observations in node m_L . Obviously, $\hat{p}_R = 1 - \hat{p}_L$.



Change of impurity from node m to its children m_L and m_R :

$$\Delta Q_{m,m_L,m_R} = Q_m(T) - \hat{p}_L Q_{m_L}(T) - \hat{p}_R Q_{m_R}(T).$$

Choice of a split:

Maximize $\Delta Q_{m,m_L,m_R}$ over all splits of the form $\{x_i \leq b\}$ for ordinal or quantitative variables and $\{x_i = b\}$ (or otherwise) for nominal variables over all attributes x_i and thresholds equal to one of the values of the attribute in the training sample.

Grow a tree using this strategy until all leaves consist of elements of one class only. Call the tree T_0 .

T_0 : usually a very big tree of high complexity – danger of overfitting.

How to make it smaller ?

Pruning a classification tree

$R(T)$ - proportion of misclassified observations in a training sample for tree T . We suspect that the actual error rate $err(T_0)$ may be quite large.

Idea: prune T_0 using cost/complexity criterion:

$$R_\alpha(T) = R(T) + \alpha|T|,$$

for $\alpha > 0$, where $|T|$ is the number of leaves of tree T . For fixed α 's we minimize $R_\alpha(T)$. For $\alpha = 0$ we get T_0 , for large α , tree consisting of the root only.

Fact: Increasing α from 0 to ∞ we get a discrete family $\{T_i\}_{i=1}^K$ of trees $\subset T_0$ such that T_i minimizes $R_\alpha(T)$ for $\alpha \in [\alpha_j, \alpha_{j+1})$, $j = 1, 2, \dots, K$.

We choose a final tree from the family $\{T_i\}_{i=1}^K$ by estimating its $err(T_i)$ using v -fold crossvalidation:

$$\hat{err}(T_i) = R_{CV}(T_i).$$

Then we choose T_{j_0} such that $R_{CV}(T_{j_0}) = \operatorname{argmin}_j R_{CV}(T_j)$. (Actually the rule is slightly more complex, but we omit details.)

Remarks

- If the true model has high level interactions, then trees are usually much more effective than linear methods at finding them. However, trees have difficulty in modelling additive structures.
- Trees handle categorical variables in a natural way.
- Trees handle missing values by ignoring them when building a tree and considering the so-called **surrogate splits** when classifying data
- Relative instability of trees: classification can change significantly when some objects from training set are removed. This is due to **hierarchical structure** of a tree: the effect of an error in a top split is propagated down to all the splits below. Ensembles of trees are often used.

10. Empirical Bayes rules (kernel and nearest neighbor DA)

Bayes rule :

Allocate an observation to the population for which

$$\pi_k p(\mathbf{x}|k)$$

is maximized.

This is a theoretical rule as we neither know π_k nor $p(\mathbf{x}|k)$.

We have to estimate these quantities:

Estimation of π_k : $\hat{\pi}_k = \frac{n_k}{n}$ where $n_k = \#\{i : Y_i = k\}$. **Warning:** Appropriate for a random sample drawn from distribution of (Y, X) , not for stratified sampling when separate samples are drawn from $(X|Y = i)$.

Estimation of $p(\mathbf{x}|k)$ is much more complex: we want to estimate density using i.i.d. sample drawn from this density.

Consider X_1, X_2, \dots, X_n iid sample in R^p pertaining to density f . The case of discrete X (e.g. multinomial) is skipped due to lack of time.

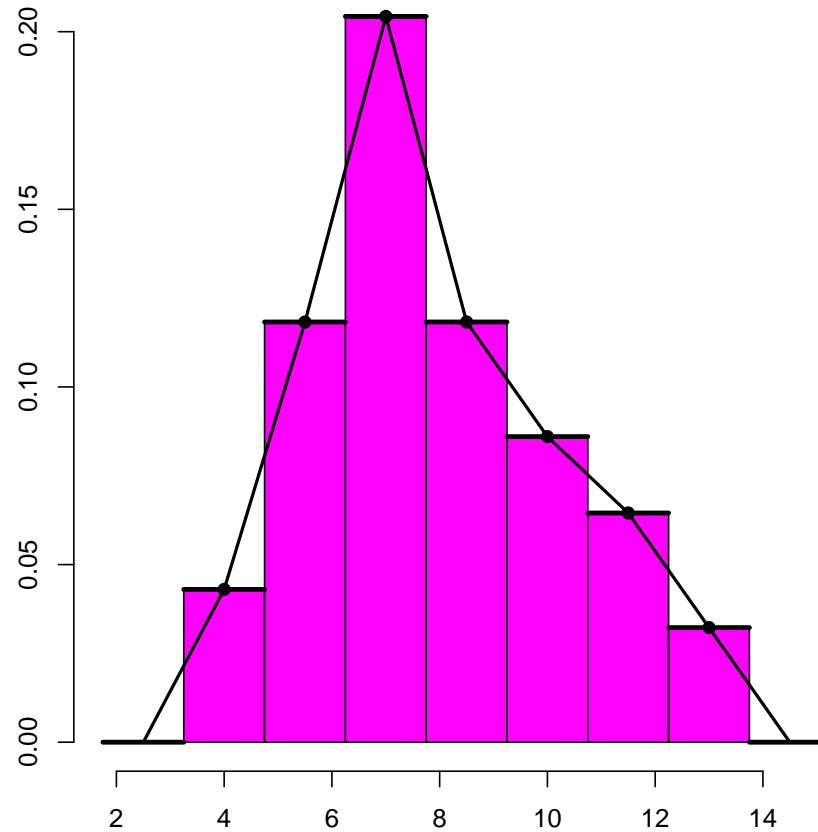
Consider $p = 1$ first. Simple estimate of f - histogram. We consider a sequence of equidistant points (adjacent points differ by h_n)

$$\dots x_{-1}(n) < x_0(n) < x_1(n) < \dots$$

such that $x_{i+1}(n) - x_i(n) = h_n$ and for $x \in (x_i(n), x_{i+1}(n)]$

$$\hat{f}_{hist}(x) = \frac{1}{h_n} (F_n(x_{i+1}(n)) - F_n(x_i(n))),$$

where $F_n(x) = \#\{X_i \leq x\}/n$. The figure shows a typical histogram with its piecewise linear approximation.



It is known that that one gets the best approximation (in probabilistic sense) for midpoints of partition intervals.

Idea: move the histogram together with a point at which we want to estimate a density

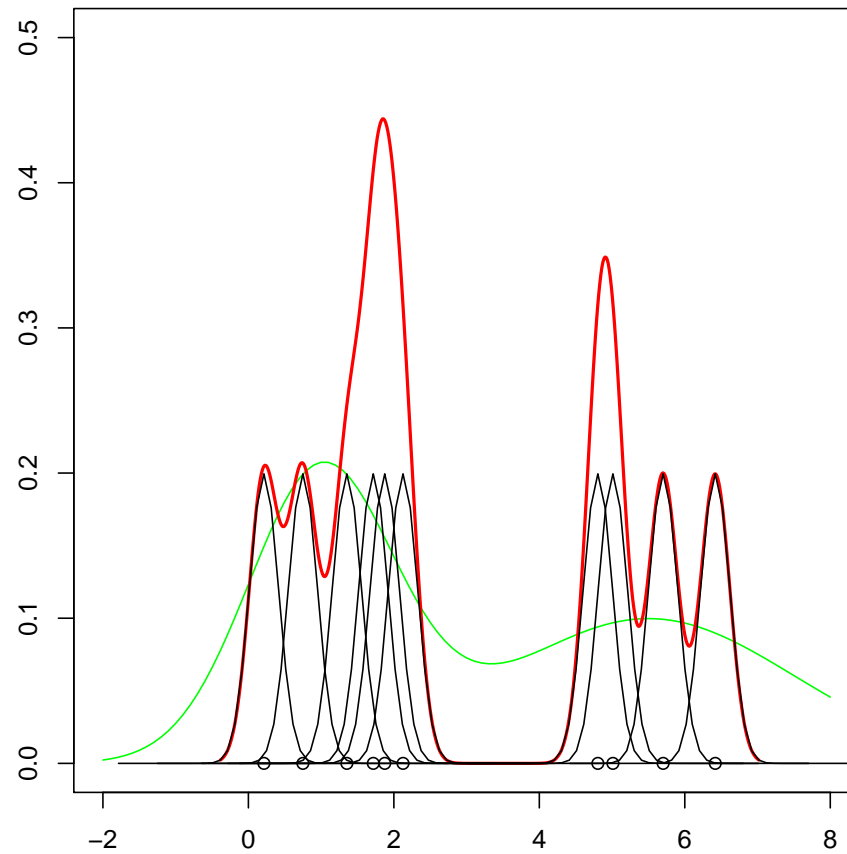
Moving-window histogram

$$\hat{f}_n(x) = \frac{1}{h_n} (F_n(x + h_n/2) - F_n(x - h_n/2)) = \frac{1}{nh_n} \sum_{i=1}^n e\left(\frac{x - X_i}{h_n}\right).$$

where $e(t) = 1$ for $-0.5 \leq t < 0.5$, otherwise 0.

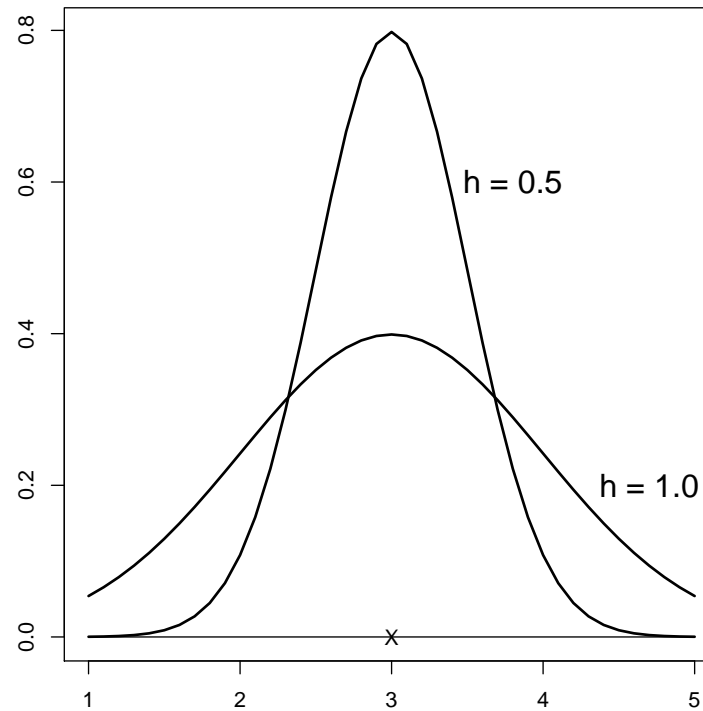
We can choose arbitrary density function K (kernel) instead of e . Rosenblatt-Parzen estimator of f

$$\hat{f}_n(x) = \frac{1}{nh_n} \sum_{i=1}^n K\left(\frac{x - X_i}{h_n}\right).$$



Usually K is taken as the standard normal density or some other smooth symmetric density.

Choice of h_n greatly influences the shape of the estimate obtained.



Usually h_n is chosen as a minimizer of some measure of accuracy of \hat{f} , in particular of $MISE = \int E(\hat{f}_n(x) - f(x))^2 dx$.

For normal f we get in this way

$$h_n = (4/3)^{1/5} \min(s_1, s_2) n^{-1/5},$$

where s_1 = empirical standard deviation of X_i and

$$s_2 = \text{interquantile range}/1.34$$

is another range-based estimate of σ . This works pretty well for unimodal densities.

Another solution: define h_n in such a way that it depends on the data and point x in such a way that $h_n(x, X_1, X_2, \dots, X_n)$ is large for x in regions when the data is sparse and is small in the opposite case. The most obvious candidate: Nearest neighbour (NN) distance. We take $k(n) \uparrow$ sequence of integers and define

$$h_n = R_n(x, X_1, X_2, \dots, X_n)$$

as the distance from x to its $k(n)^{th}$ nearest neighbour.

Kernel estimation of f is readily generalized to p dimensions. The 'only' problem is the curse of dimensionality: for large p we need enormous sample sizes to estimate density in this way. The ways to circumvent it are based on finding interesting low-dimensional projections of f - projection pursuit density estimation.

How does this translate to classification rules?

Empirical Bayes rules are obtained:

Allocate an observation to the population for which $\hat{\pi}_k \hat{p}(x|k)$ is maximized.

For $g = 2$ allocate to population 1 if

$$\frac{n_1}{n} \frac{1}{n_1 h_n} \sum_{i=1}^{n_1} K\left(\frac{x - X_{i1}}{h_n}\right) \geq \frac{n_2}{n} \frac{1}{n_2 h_n} \sum_{i=1}^{n_2} K\left(\frac{x - X_{i2}}{h_n}\right).$$

h_n is chosen usually the same for both samples. For $h_n = R_n$ and uniform kernel we get the celebrated kNN rule:

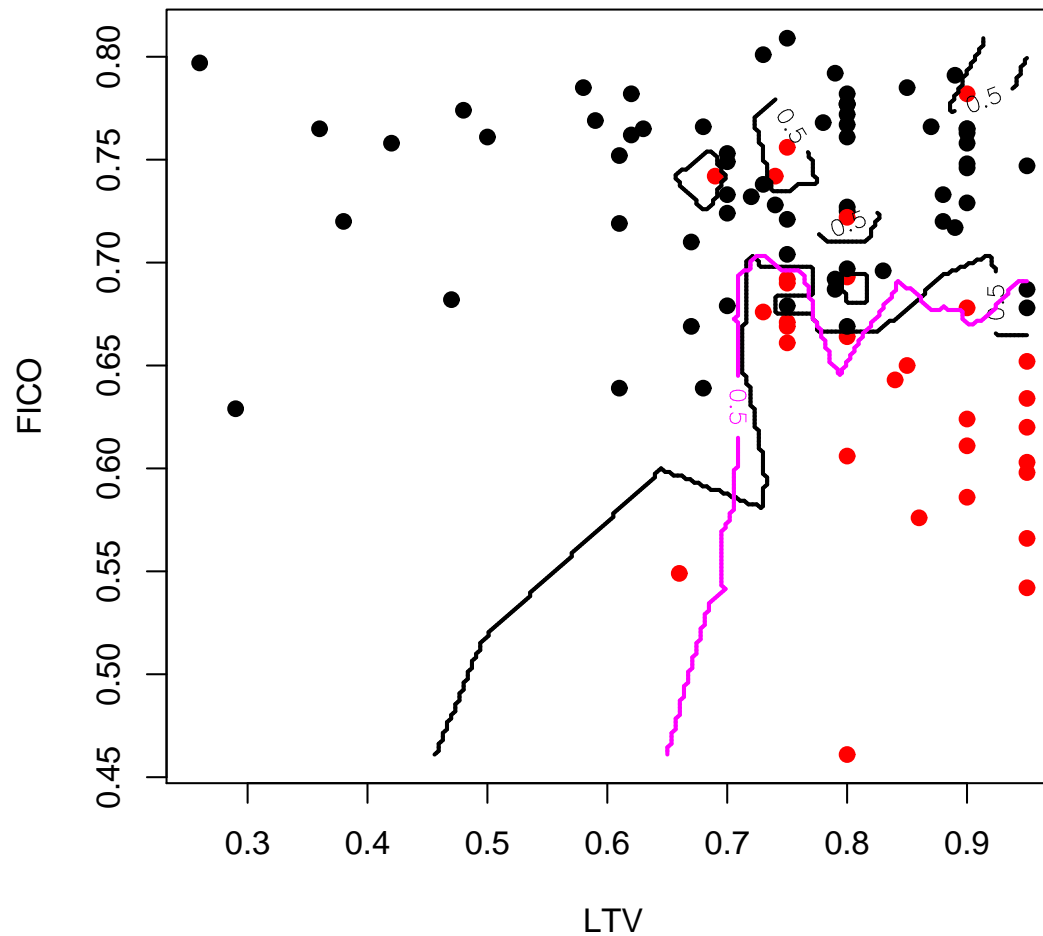
Consider the sphere around x containing exactly $k(n)$ nearest neighbors in the merged sample. Classify x to the class having the most representatives in the neighbourhood. In the case of ties classify arbitrarily to one of the classes with most representatives.

This works usually well even for $k(n) \equiv 1$ when all coordinates of x are quantitative and have comparable variability. Otherwise we have to standardize.

Observe that for 1-NN rule resubstitution estimator of err is always equal to 0, indicating how inadequate this estimator can be.

Example *Mortgage data* concerns default on house credit payments and its two predictors FICO (Fair Isaac scoring) and LTV (ratio of credit to the value of the house).

Mortgage Data: k-nn
k=1 (black), k=5 (magenta)



Small 'islands' containing few points indicate overfitting. kNN with $k = 5$ seems to be much less prone to overfitting in this case.

11. Other methods (a brief synopsis)

- **Mixture Discriminant Analysis (MDA)**
- **Prototype methods**

Training data is represented by a set of points (prototypes) in feature space. Classification of a query point x is made to the class of the closest prototype. 'Closest' is usually defined by Euclidean distance after standardization of each predictor.

Crucial: prototypes should be well positioned to capture the distribution of each class.

Example: Learning Vector Quantization (LVQ1)

1. Choose R initial prototypes for each class: $m_1(j), \dots, m_R(j)$, $j = 1, \dots, g$, e.g. by random sampling.

2. Sample a training point \mathbf{x}_i (with replacement) and let (k, j) be the index of the closest prototype $m_k(j)$ to \mathbf{x}_i . Now,

(a) if $y_i = j$ (i.e., they are in the same class) move the prototype **towards** the training point.

$$m_k(j) \leftarrow m_k(j) + \varepsilon(\mathbf{x}_i - m_k(j)),$$

where ε is positive learning rate;

(b) if $y_i \neq j$ (i.e., they are in different classes) move the prototype **away** the training point.

$$m_k(j) \leftarrow m_k(j) - \varepsilon(\mathbf{x}_i - m_k(j)).$$

3. Repeat step 2 decreasing the learning rate ε with each iteration to 0.

Drawback: LVQ is not based on optimization of some criterion what makes it difficult to understand its properties.

- **Flexible Discriminant Analysis (FDA)**

The method stemming from multivariate linear regression analysis in which nonlinear transformations of features and class indicators are allowed.

- **Support Vector Machines (SVM)**

For two linearly separable classes the method relies on construction of two parallel hyperplanes separating the classes with the widest possible margin between them (to be discussed in some detail).

12. Ensembles of classifiers

Various classifiers may be constructed using different paradigms (LDA, QA, logistic, NN classifiers, classification trees).

Can we enhance their performance ?

Yes, by taking **all** their decisions into account, as in:

Bagging

C bootstrap samples are drawn from the original sample. C classifiers are built pertaining to respective samples (elements sampled several times are considered as different data points).

For a new observation we let it be classified by all individual classifiers and assign it to a class by majority voting.

Usually yields some, but not very significant improvement over an individual well-chosen classifier, in contrast to

Boosting

- Based on the idea of weighing elements of the training set;
- Observations misclassified by i^{th} classifier receive larger weights when fed to a next classifier. Successive classifiers are forced to concentrate on training observations misclassified by previous ones in a sequence.

Weighted trees: simple modification of ordinary trees consisting in replacing class proportions for each node of a tree by weighted proportions. Quality of splits is judged using weighted proportions.

AdaBoost algorithm: $C_i(x)$, $i = 1, \dots, C$ classifiers with values in $\{-1, 1\}$.

1. Initialize observation weights $w_i = 1/n$, $i = 1, 2, \dots, n$.

2. For $m = 1, \dots, C$

(a) Fit a classifier $C_m(x)$ to the training data using weights w_i .

(b) Compute weighted error

$$err_m = \frac{\sum_{i=1}^n w_i I(y_i \neq C_m(x_i))}{\sum_{i=1}^n w_i}.$$

(c) Compute $\alpha_m = \log((1 - err_m)/err_m)$.

(d) $w_i \leftarrow w_i \exp(\alpha_m I(y_i \neq C_m(x_i)))$, $i = 1, \dots, n$.

3. Output $C(x) = \text{sign}(\sum_{m=1}^C \alpha_m C_m(x))$.

Observe that, as long as $err_m < 1/2$, $\alpha_m > 0$ and hence the elements which were misclassified at the m^{th} stage get larger weights in the next round of the algorithm.

Important note: If properly regularized, boosting is asymptotically Bayes optimal.

Random Forests

To be discussed.

Random Forests and ensembles based on boosting seem to be the best off-the-shelf classifiers!