Machine learning methods for bioinformatics

INFO-F-528

Gianluca Bontempi

Département d'Informatique
Boulevard de Triomphe - CP 212
http://mlg.ulb.ac.be

Discriminant functions

- In zero-one loss function the optimal classifier is achieved by assigning an input pattern to a class for which the posterior probability (or the associated discriminant function $g_k(x) = \text{Prob}\{\mathbf{y} = k|x\}$) is maximum (*maximum a posteriori*)

$$c^*(x) = \arg \max_{c_k \in \{c_1,\ldots,c_K\}} \text{Prob}\{\mathbf{y} = k|x\}$$

- We can multiply all the discriminant functions by the same positive constant or shift them by the same additive constant without influencing the decision.
- More generally, if we replace every $g_k(x)$ by $f(g_k(x))$, where $f(\cdot)$ is a monotonically increasing function, the resulting classification is unchanged.

$$c^*(x) = \arg \max_{c_k \in \{c_1,\ldots,c_K\}} g_k(x) = \arg \max_{c_k \in \{c_1,\ldots,c_K\}} f(g_k(x))$$

Discriminant functions

- Any of the following choices gives identical classification result:

$$g_k(x) = \text{Prob}\{\mathbf{y} = k|x\} = \frac{p(x|\mathbf{y} = k)P(\mathbf{y} = k)}{\sum_{k=1}^{K} p(x|\mathbf{y} = k)P(\mathbf{y} = k)}$$

$$g_k(x) = p(x|\mathbf{y} = k)P(\mathbf{y} = k)$$

$$g_k(x) = \ln p(x|\mathbf{y} = k) + \ln P(\mathbf{y} = k)$$

and returns a minimum-error-rate classification.

Discriminant functions in the gaussian case

- Let us consider the case where the densities are multivariate normal, i.e. $p(x|\mathbf{y}=k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ where $x \in \mathbb{R}^n$, μ_k is a $[n,1]$ vector and Σ_k is a $[n,n]$ covariance matrix.
- Since

$$p(x|\mathbf{y}=k) = \frac{1}{(\sqrt{2\pi})^n \sqrt{\det(\Sigma_k)}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right\}$$

the discriminant function is then

$$g_k(x) = \ln p(x|\mathbf{y}=k) + \ln P(\mathbf{y}=k) =$$
$$= -\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) - \frac{n}{2}\ln 2\pi - \frac{1}{2}\ln\det(\Sigma_k) + \ln P(\mathbf{y}=k)$$

This function is known as the *normal-based quadratic discriminant function*

The discriminant algorithm

- We obtain the following discriminant rule: assign the input x to the class c_k if $g_k(x) > g_i(x)$ for all $k \neq i$.
- In practical tasks, given a training set D_N, the quantities μ_k, Σ_k and $P(\mathbf{y} = k)$ are replaced by their sampled estimators

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} x_{ik},$$

$$\hat{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} (x_{ik} - \hat{\mu}_k)(x_{ik} - \hat{\mu}_k)^T,$$

$$\hat{P}(\mathbf{y} = k) = \frac{N_k}{N}$$

  where $\{x_{ik}\}, i = 1, \ldots, N_k$ is the set of training input patterns attributed to the class c_k and $\sum_{k=1}^{K} N_k = N$.
- In the following we will consider the simplest case: $\Sigma_k = \sigma^2 I$ where I is the $[n, n]$ identity matrix.

Gaussian case: $\Sigma_k = \sigma^2 I$

- This means that all the classes have a Gaussian distribution of $\mathbf{x}$ where the covariance matrix is identical and diagonal.
- For each class, the x samples fall in equal-size spherical clusters which are parallel to the axes.
- We have that the two terms

$$\det(\Sigma_k) = \sigma^{2n}, \qquad \Sigma_k^{-1} = (1/\sigma^2) I$$

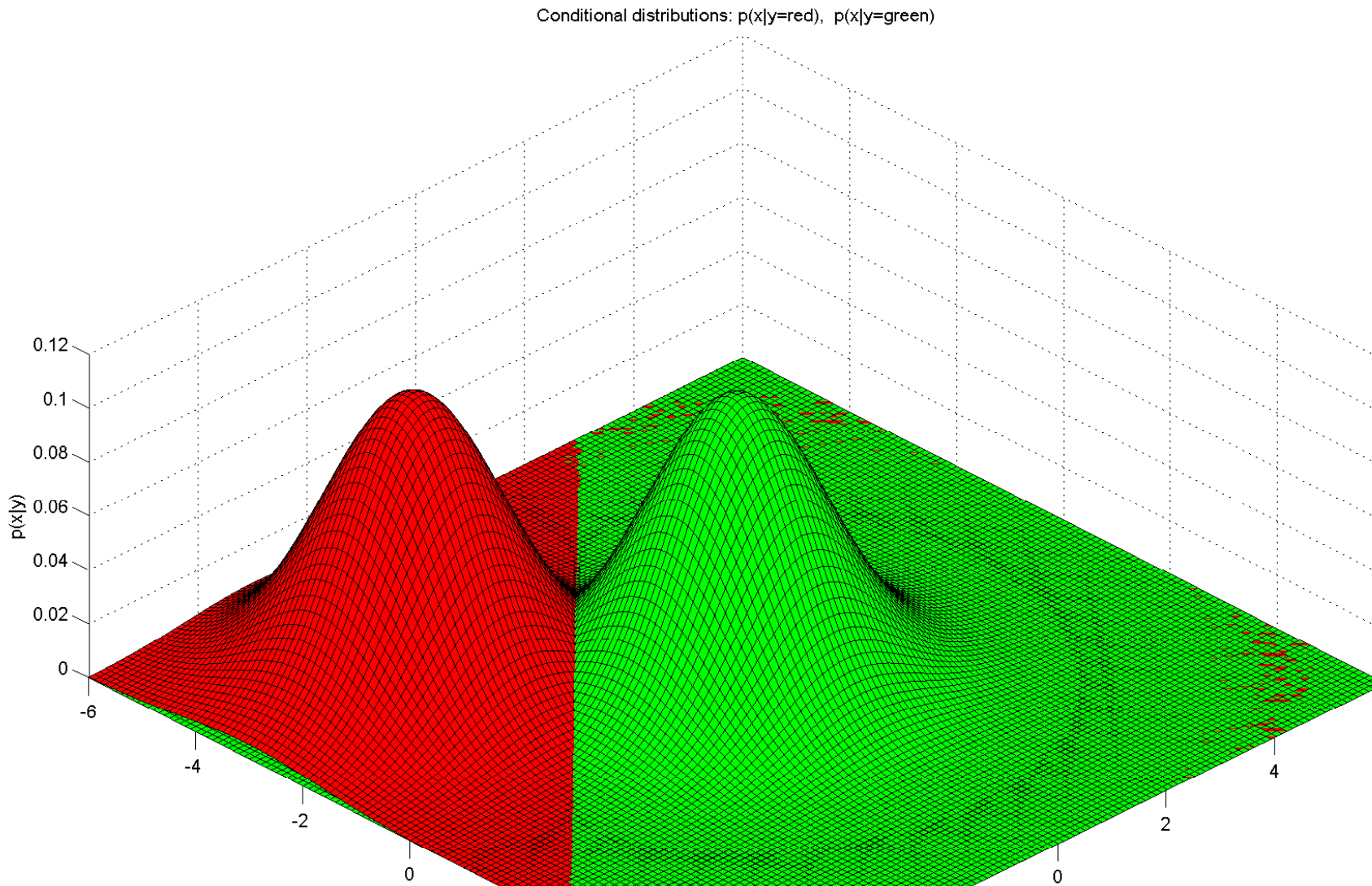
  are independent of k, then they are unimportant additive constants that can be ignored.
- Thus we obtain the simple discriminant function

$$g_k(x) = -\frac{\|x - \mu_k\|^2}{2\sigma^2} + \ln P(\mathbf{y} = k) = -\frac{(x - \mu_k)^T (x - \mu_k)}{2\sigma^2} + \ln P(\mathbf{y} = k)$$

$$= -\frac{1}{2\sigma^2}[x^T x - 2\mu_k^T x + \mu_k^T \mu_k] + \ln P(\mathbf{y} = k)$$

  where $\|\cdot\|$ denotes the *Euclidean norm*.

2 Gaussians with same covariance matrix



- Since the quadratic term $x^T x$ is the same for all k, making it an ignorable additive constant, this is equivalent to a linear discriminant function

$$g_k(x) = w_k^T x + w_{k0}$$

where w_k is a $[n, 1]$ vector

$$w_k = \frac{1}{\sigma^2}\mu_k$$

and the term w_{k0}

$$w_{k0} = -\frac{1}{2\sigma^2}\mu_k^T \mu_k + \ln P(\mathbf{y} = k)$$

is called the *bias* or threshold.

Decision boundary

In the two-classes problem, the decision boundary (i.e. the set of points where $g_1(x) = g_2(x)$) is given by the hyperplane having equation

$$w^T(x - x_0) = 0$$

where

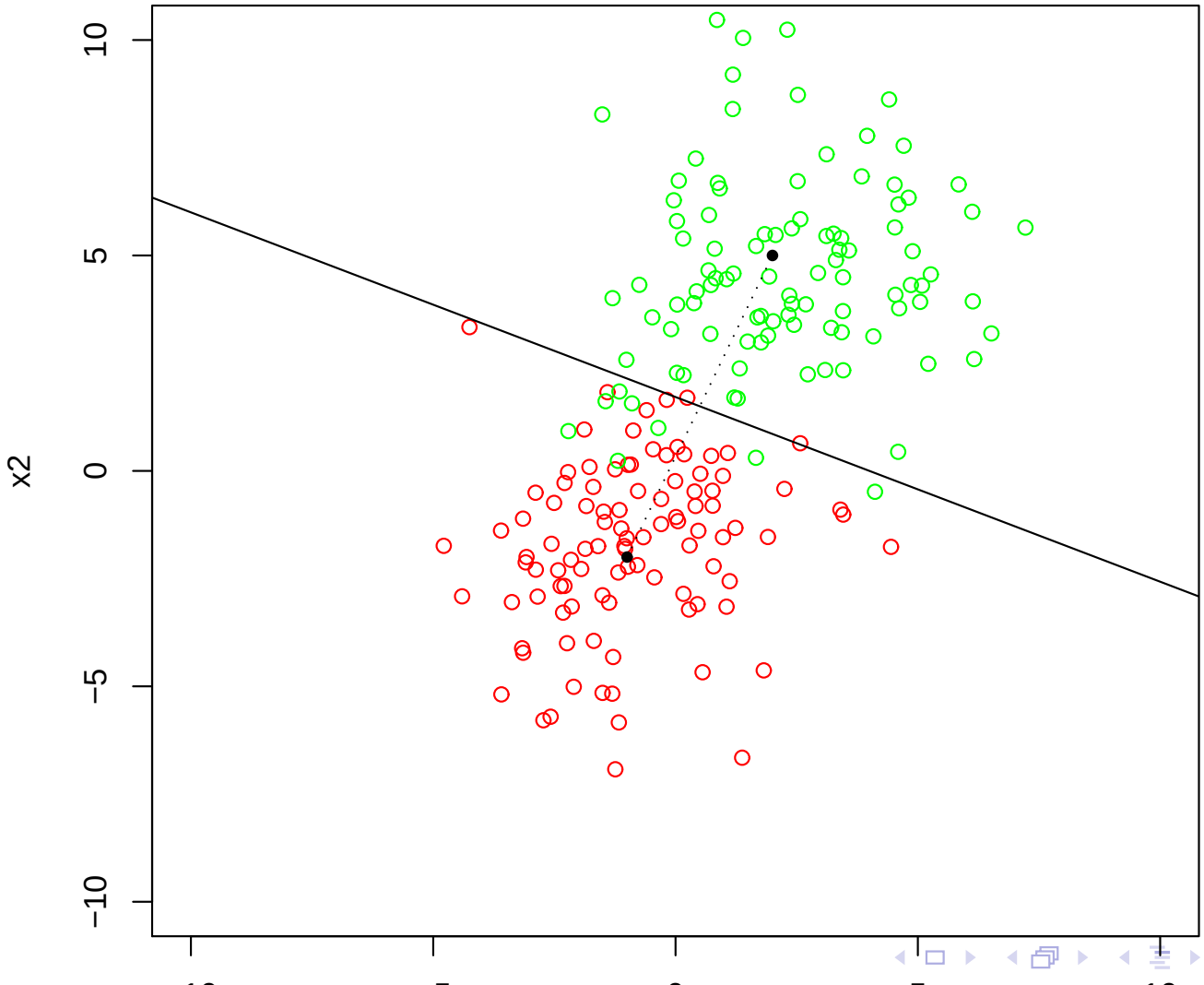
$$w = \mu_1 - \mu_2$$

and

$$x_0 = \frac{1}{2}(\mu_1 + \mu_2) - \frac{\sigma^2}{\|\mu_1 - \mu_2\|^2} \ln \frac{\text{Prob}\{\mathbf{y} = 1\}}{\text{Prob}\{\mathbf{y} = 2\}}(\mu_1 - \mu_2)$$

This equation defines a hyperplane through the point x_0 and orthogonal to the vector w.
Note that if the variance σ^2 is small with respect to the squared distance of the two means, the decision boundary is not very sensitive to the exact values of the a priori probabilities.
See R script `discri.R`.

R script `discri.R`



Uniform prior case

- If the prior probabilities $P(\mathbf{y} = k)$ are the same for all the K classes, then the term $\ln P(\mathbf{y} = k)$ is an unimportant additive constant that can be ignored.
- In this case, it can be shown that the optimum decision rule is a *minimum distance classfier*.
- This means that in order to classify an input x, it measures the Euclidean distance $\|x - \mu_k\|^2$ from x to each of the K mean vectors, and assign x to the category of the nearest mean

$$\hat{c}(x) = \arg\min_k \|x - \mu_k\|^2$$

- It can be shown that for the more generic case $\Sigma_k = \Sigma$, the discriminant rule is based on minimizing the Mahalanobis distance

$$\hat{c}(x) = \arg\min_k (x - \mu_k)^T \Sigma^{-1} (x - \mu_k)$$

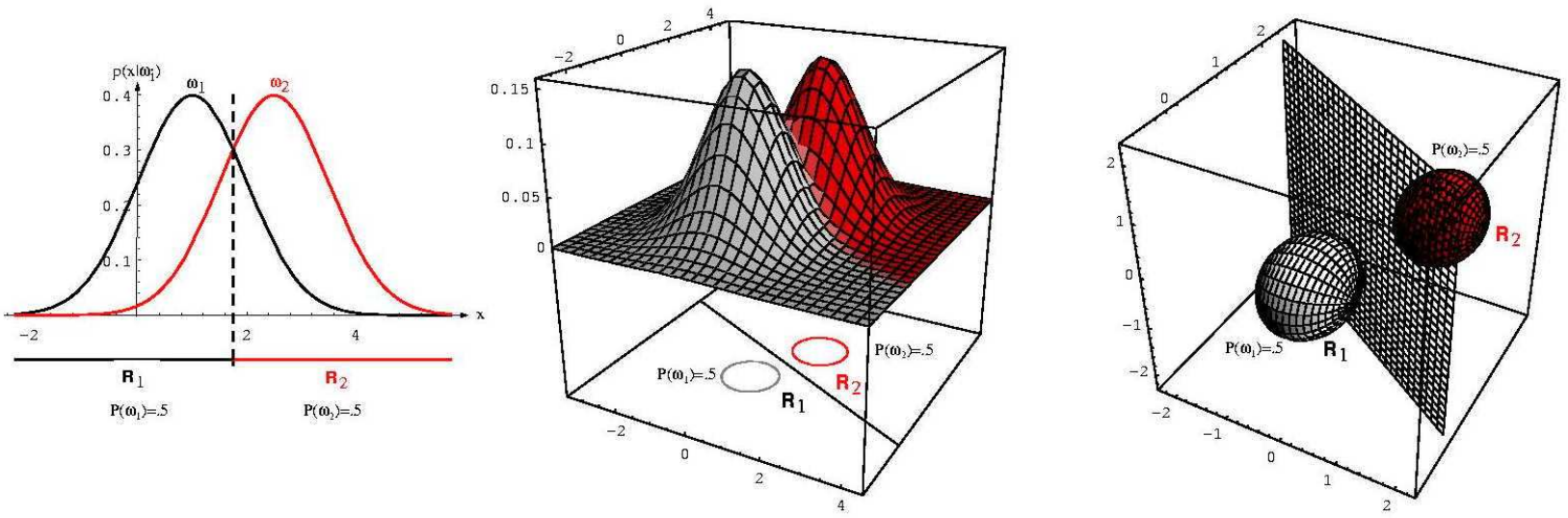
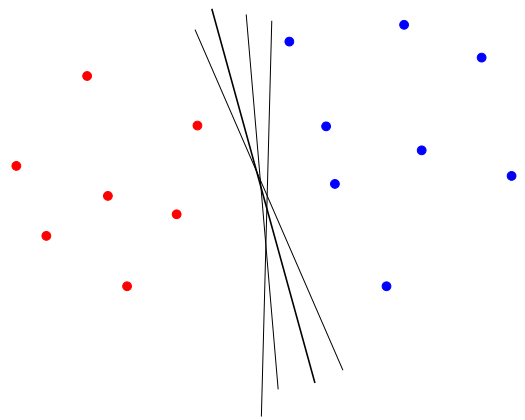


Figure 2.10: If the covariances of two distributions are equal and proportional to the identity matrix, then the distributions are spherical in d dimensions, and the boundary is a generalized hyperplane of $d-1$ dimensions, perpendicular to the line separating the means. In these 1-, 2-, and 3-dimensional examples, we indicate $p(\mathbf{x}|\omega_i)$ and the boundaries for the case $P(\omega_1) = P(\omega_2)$. In the 3-dimensional case, the grid plane separates $\mathcal{R}_1$ from $\mathcal{R}_2$.

Separating hyperplanes

- We showed that in a binary classification task where the two classes are Gaussian distributed with the same covariance, the optimal boundary is an hyperplane (and for $n = 2$ a line).
- In a generic binary classification task, the problem of sinding a separating hyperplane is ill-posed and there are infinitely many possible *separating hyperplanes* of equation

$$\beta_0 + x^T \beta = 0, \qquad x \in \mathbb{R}^n \tag{1}$$



About hyperplanes

In a generic case ($x \in \mathbb{R}^n$) some properties hold for all hyperplanes

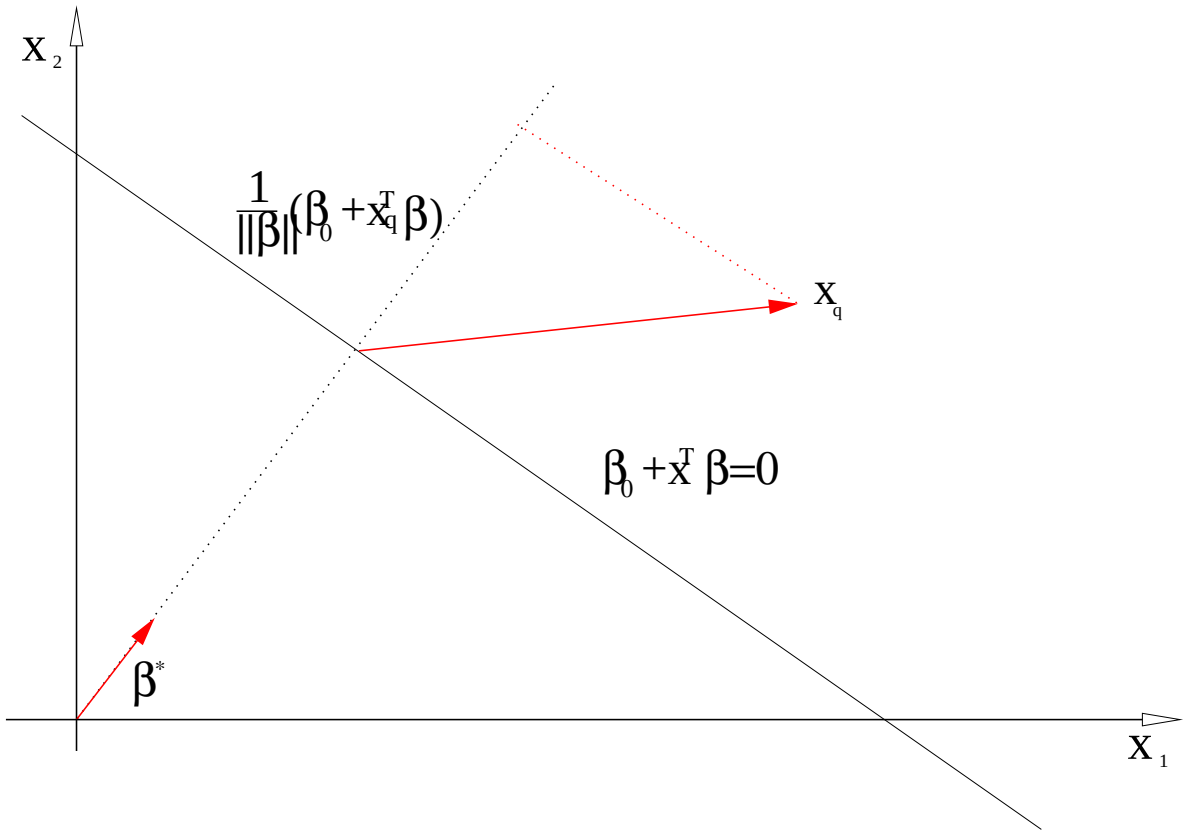
- Since for any two points x_1 and x_2 lying on the hyperplane we have $(x_1 - x_2)^T \beta = 0$, the vector normal to the hyperplane is given by

$$\beta^* = \frac{\beta}{\|\beta\|}$$

- The signed distance of a point x to the hyperplane is called the *geometric margin* and is given by

$$\beta^{*T}(x - x_0) = \frac{x^T\beta - \beta x_0^T}{\|\beta\|} = \frac{1}{\|\beta\|}(x^T\beta + \beta_0)$$

Hyperplane in a bidimensional space



Bidimensional space ($n = 2$): vector β^* normal to the hyperplane and distance of a point from an hyperplane

Perceptrons

- An hyperplane $\beta_0 + \beta^T x = 0$ partitions the input space into two regions where the quantity $\beta_0 + \beta^T x$ takes opposite signs. Let us consider a binary problems where $\mathbf{y} \in \{-1, 1\}$.
- Classifiers that use the sign of the linear combination $\beta_0 + \beta^T x$ to perform classification were called *perceptrons* in the engineering literature in the late 1950.
- The class returned by a perceptron for a given input x_q is

$$\begin{cases} 1 & \text{if } \beta_0 + x_q^T \beta = \beta_0 + \sum_{j=1}^n x_{qj} \beta_q > 0 \\ -1 & \text{if } \beta_0 + x_q^T \beta = \beta_0 + \sum_{j=1}^n x_{qj} \beta_q < 0 \end{cases}$$

- In other terms the decision rule is given by

$$\hat{c}(x) = \text{sgn}(\beta_0 + x^T \beta) \tag{2}$$

where $\hat{c}(x) \in \{-1, 1\}$ is the class returned by the perceptron.

Training of perceptrons

- Let us consider a training set $D_N = \{(x_i, y_i\}, i = 1, \ldots, N\}$
- A point of the training set is well classified points iff the following relation holds

$$\gamma_i = y_i(x_i^T \beta + \beta_0) > 0$$

  where the quantity γ_i is called the *functional margin* of the pair $\langle x_i, y_i \rangle$ with respect to the hyperplane.
- Misclassifications in the training set occur when

$$\begin{cases} y_i = 1 & \text{but } \beta_0 + \beta^T x_i < 0 \\ y_i = -1 & \text{but } \beta_0 + \beta^T x_i > 0 \end{cases} \Leftrightarrow y_i(\beta_0 + \beta^T x_i) < 0$$

Parametric identification in perceptrons

- The parametric identification step of a perceptron learning procedure aims at finding the values $\{\beta, \beta_0\}$ that minimize the misclassification in the training set

$$\widehat{\text{MISE}}_{\text{emp}}(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0)$$

  where $\mathcal{M}$ is the subset of misclassified points in training set.
- Note that this quantity is non-negative and proportional to the distance of the misclassified points to the hyperplane. Since the gradients are

$$\frac{\partial \widehat{\text{MISE}}_{\text{emp}}(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i, \qquad \frac{\partial \widehat{\text{MISE}}_{\text{emp}}(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

  a gradient descent minimization procedure can be adopted.
- This procedure is guaranteed to converge provided there exists a hyperplane that correctly classifies the data: this configuration is called *linearly separable*.

Perceptrons limits

Although the perceptron set the foundations for much of the following research in machine learning, a number of problems with this algorithm have to be mentioned:

- When the data are separable, there are many possible solutions, and which one is found depends on the initialization of the gradient method.
- When the data are not separable, the algorithm will not converge.
- Also for a separable problem the convergence of the gradient minimization can be very slow.

A possible solution to the separating hyperplane problem has been proposed by the SVM technique.

Support vector machines

- Let the *geometric margin of a hyperplane with respect to a training dataset* be the minimum of the geometric margin of the training points.
- Let us define the *margin of a training set* as the maximum geometric margin over all hyperplanes and the hyperplane realizing this maximum as the *maximal margin hyperplane*.
- The SVM approach computes the maximal margin hyperplane for a training set. In other words, the SVM optimal separating hyperplane is the one which separates the two classes by maximizing the distance to the closest point from either class.
- This approach provides a unique solution to the separating hyperplane problem and was shown to lead to good classification performance on real data.

SVM parametric identification

- The search for the optimal hyperplane is modeled as the optimization problem

$$\max_{\beta,\beta_0} C$$

$$\text{subject to } \frac{1}{\|\beta\|} y_i(x_i^T \beta + \beta_0) \geq C \quad \text{for } i = 1, \ldots, N$$

  where the constraint ensures that all the points are at least a distance C from the decision boundary defined by β and β_0.
- The SVM parametric identification step seeks the largest C that satisfies the constraints and the associated parameters.

SVM parametric identification

It can be shown that the original problem (dimensionality $n+1$) can be put in a dual form of dimensionality N

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k = \tag{3}$$

$$= \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \langle x_i, x_k \rangle \tag{4}$$

$$\text{subject to } 0 = \sum_{i=1}^{N} \alpha_i y_i, \tag{5}$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, N \tag{6}$$

where

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{7}$$

and $\langle x_i, x_k \rangle$ is the inner product of x_i and x_k.

SVM parametric identification

It can be shown that the solution α_i are in either of these two situations

1. $\alpha_i > 0$: the point (called *support vector*) is on the boundary of the margin
2. $\alpha_i = 0$: the point is not on the boundary of the margin

Now, the decision function can be written as

$$h(x, \beta, \beta_0) = \text{sign}[x^T \beta + \beta_0]$$

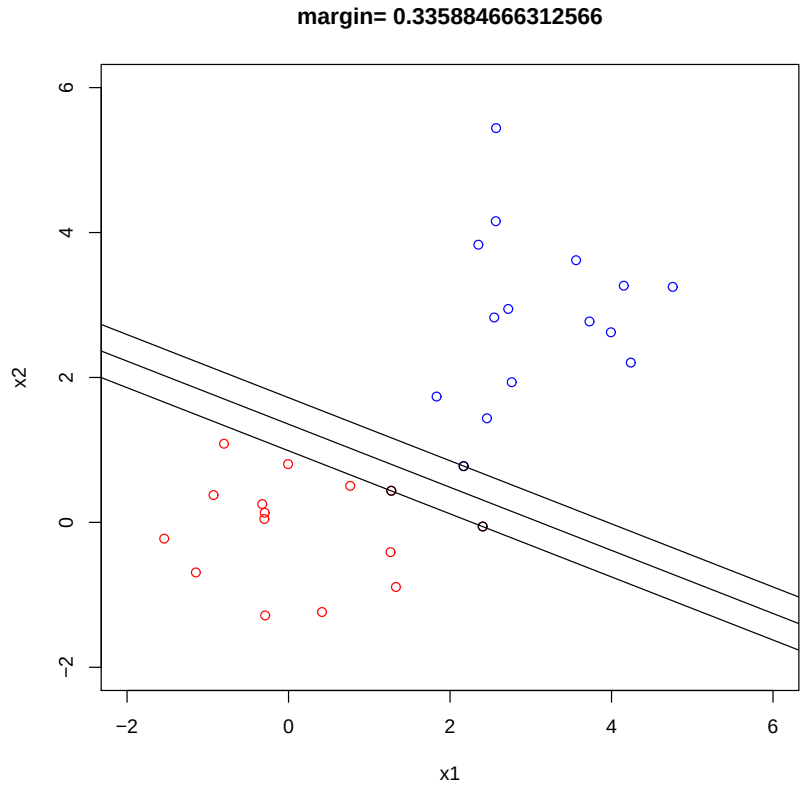
or equivalently

$$h(x, \beta, \beta_0) = \text{sign}[\sum_{\text{support vectors}} y_i \alpha_i \langle x_i x \rangle + \beta_0] \quad (8)$$

SVM parametric identification

- The classifier can be expressed as a function of a limited number of points of the training set, the so called *support vectors* which are on the boundaries.
- This means that in SVM all the points far from the class boundary do not play a major role, unlike the linear discriminant rule where the mean and the variance of the class distributions determine the separating hyperplane.

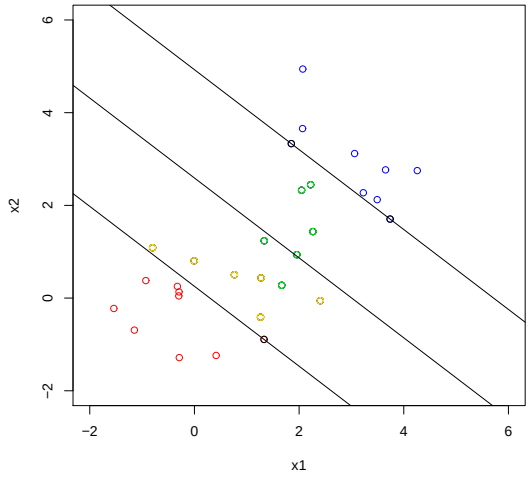
Support vector machines: R script

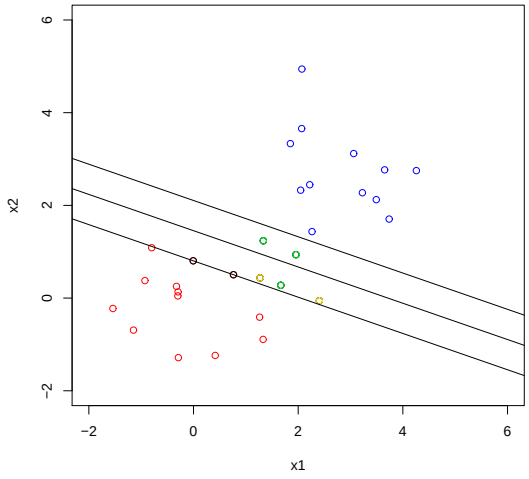


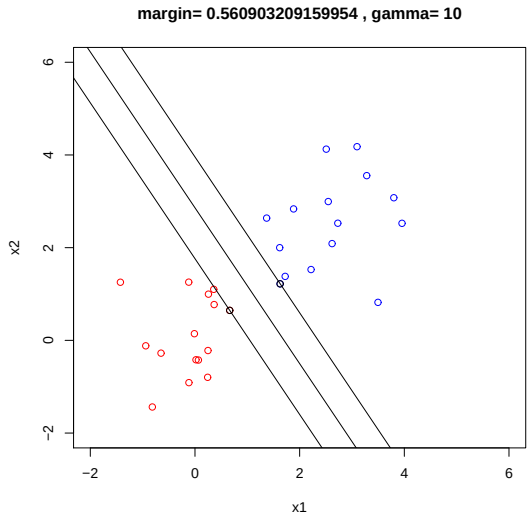
Maximal margin hyperplane for a binary classification task with the support vectors in black.

Non separable cases

- So far we considered classification tasks where we aim to find an hyperplane which features no errors in the training set.
- A modification of the SVM formulation occurs when we suppose that the classes are nonlinearly separable.
- In this case the dual problem is unbounded.
- The idea is still to maximize the margin but by allowing some points to be misclassified.
- The amount of relaxation is controlled by a parameter γ which represents the maximum number of allowed misclassifications in the training set.
- The value γ plays the role of complexity parameter which bounds the total proportional amount by which classifications fall on the wrong side of the margin. In practice, the choice of this parameter requires a structural identification loop where the parameter γ is varied through a wide range of values and assessed through a validation strategy.





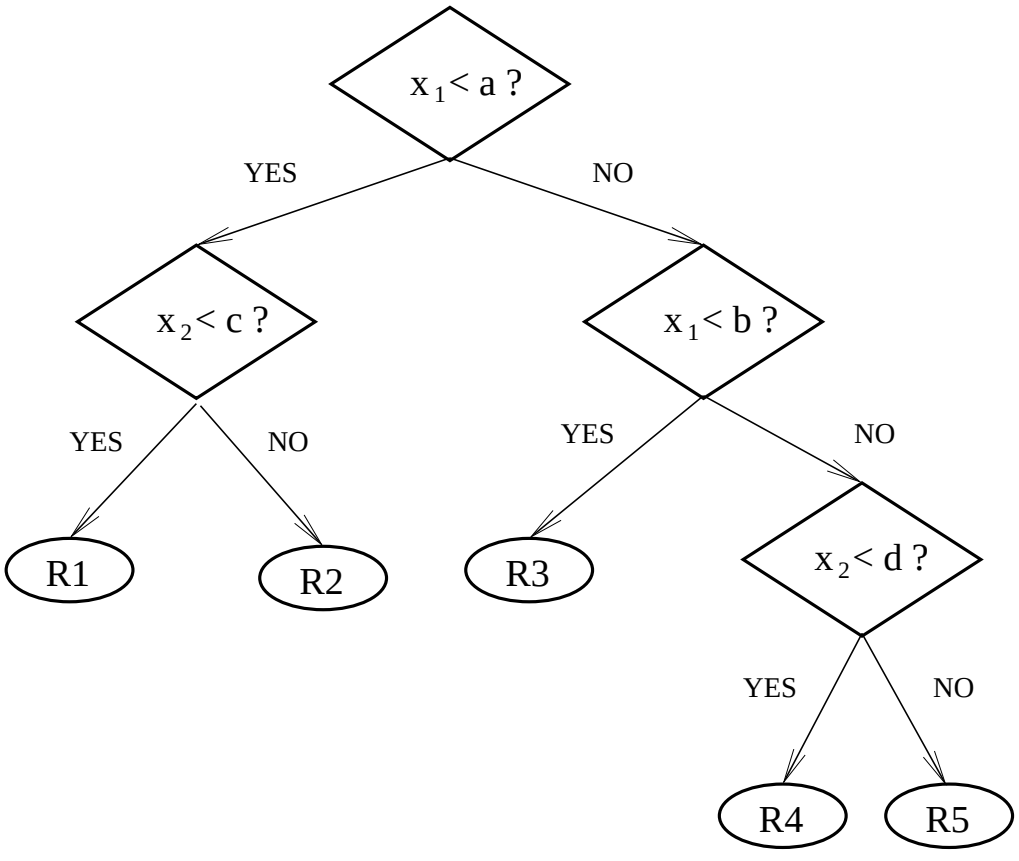


Support vectors are in black, the slack points of the red class are in yellow and the slack points of the blue class are in green

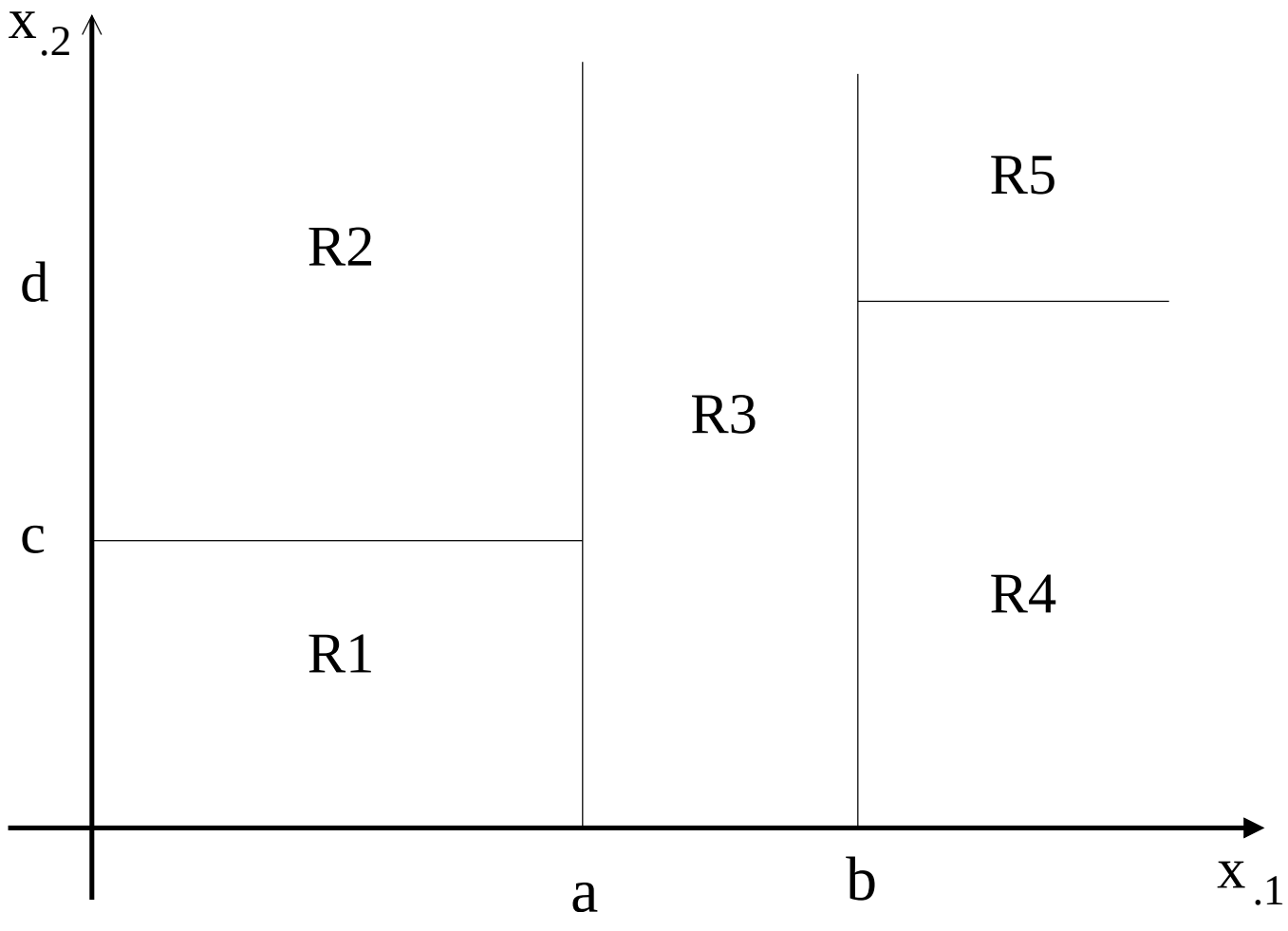
Decision Trees

- The use of tree-based learners dates back to 1963.
- A decision tree partitions the input space into mutually exclusive regions, each of which is assigned a specific model.
- The nodes of a decision tree can be classified in internal nodes and terminal nodes. An *internal node* is a decision-making unit that evaluates a decision function to determine which child node to visit next.
- A *terminal node* or *leaf* has no child nodes and is associated with one of the partitions of the input space. Each terminal node has a unique path that leads from the root to itself.
- In *classification trees* each terminal node contains a label that indicates the class for the associated input region. In *regression trees* the terminal node contains a model that specifies the input/output mapping for the corresponding input partition.
- Divide-and-conquer approach.
- Ease of interpretability.

A binary decision tree.



Input space partitioning



Classification tree predictor

- Let m be the number of leaves and $h_j(\cdot, \alpha_j)$, $j = 1, \ldots, m$, the input/output model associated with the j^{th} leaf.
- Once a prediction in a query point q is required, the query is presented to the root node of the decision tree;
- Depending on the result of the associated decision function, the tree will branch to one of the root's children.
- The procedure is iterated recursively until a leaf $\bar{j}$ is reached and a input/output model is selected. The returned output will be the value $h_{\bar{j}}(q, \alpha_{\bar{j}})$.
- Example let $q = (x_1, x_2)$ with $x_1 < a$ and $x_2 > b$. The predicted output will be $y_q = h_2(q, \alpha_2)$ where α_2 is the vector of parameters of the model localized in region $R2$.

Classification tree learning

The learning procedure has two steps: tree growing and tree pruning:

- During tree growing the algorithm makes a succession of splits that partition the training data into disjoint subsets. Starting from the root node that contains the whole dataset, an exhaustive search is performed to find the split that best reduces a certain cost function.
- Let us consider a certain node t and let $D(t)$ be the corresponding subset of the original D_N. Consider a measure of the variability (or lack of coherence) of the $N(t)$ data contained in the node t, for instance the variance :

$$V(t) = \hat{p}_1(1 - \hat{p}_1) \tag{9}$$

where $\hat{p}_1 = \frac{N_1(t)}{N(t)}$ and $N_1(t)$ is the number of samples with class 1 in the t node.

Classification tree construction

- For any possible split s of node t into the two children t_r and t_l, we define the quantity

  $$\Delta V(s,t) = V(t) - (V(t_l) + V(t_r)) \text{ with } N(t_r) + N(t_l) = N(t)$$

  that represents the decrease of the variability due to a further partition of the dataset.
- The best split is the one that maximizes the decrease ΔV

  $$s^* = \arg\max_s \Delta V(s,t) \tag{10}$$

- Once the best split is attained, the dataset is partitioned into the two disjoint subsets of length $N(t_r)$ and $N(t_l)$, respectively. The same method is recursively applied to all the leaves.
- The procedure terminates either when the error measure associated with a node falls below a certain tolerance level, or when the error reduction ΔV resulting from further splitting does not exceed a threshold value.

Tree pruning

- The tree that the growing procedure yields is typically too large and presents a serious risk of overfitting the dataset. For that reason a pruning procedure is often adopted.
- Pruning uses a complexity based measure of the tree performance

$$V_\lambda(T) = V(T) + \lambda|T|$$

where λ is a parameter that accounts for the tree's complexity and $|T|$ is the number of terminal nodes of the tree T. For a fixed λ we define with $T(\lambda)$ the tree structure which minimizes the quantity $V_\lambda(T)$.

- The parameter λ is gradually increased in order to generate a sequence of tree configurations with decreasing complexity

Tree pruning

- For a generic subtree $T_t \subset T$

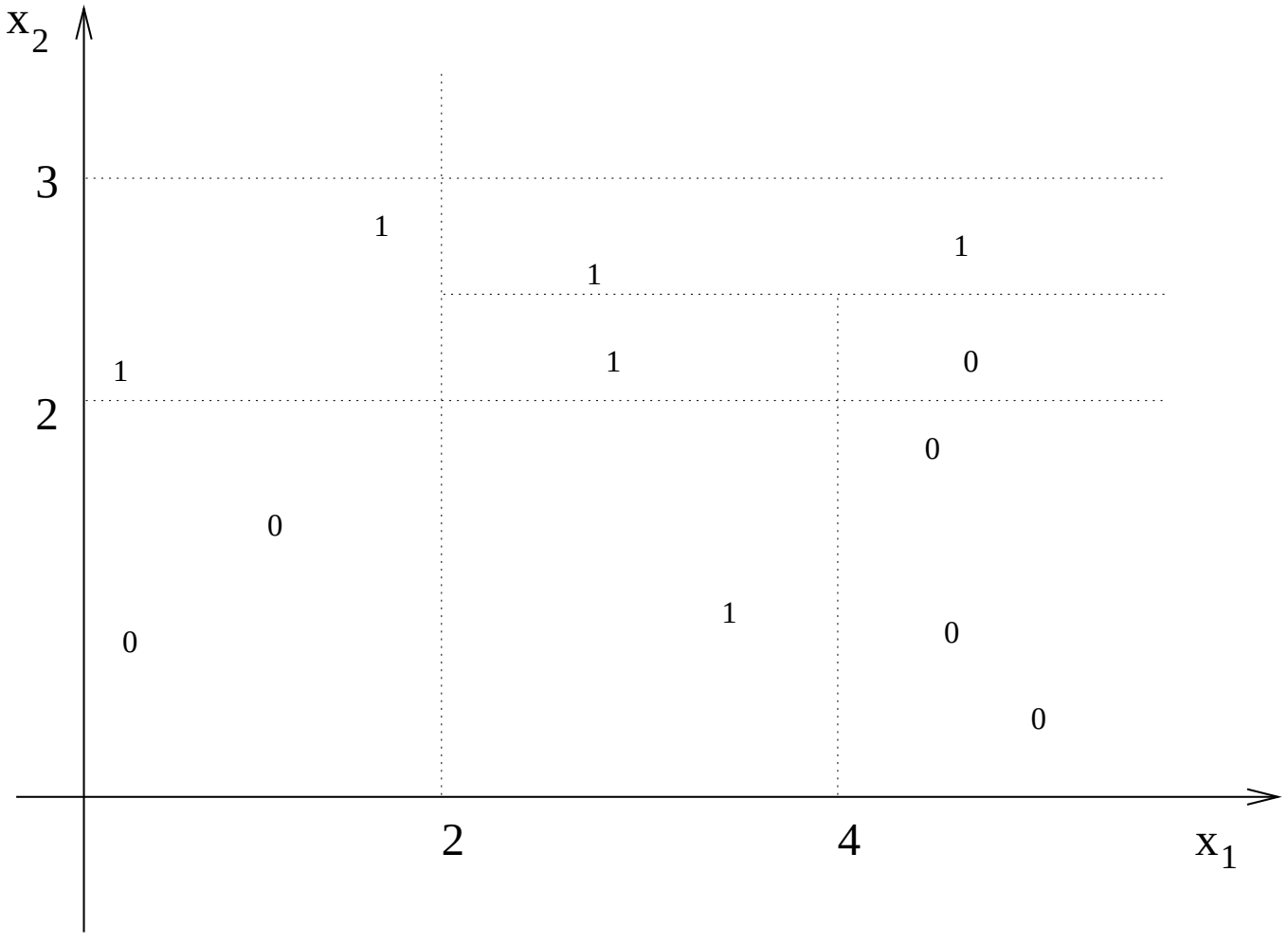
$$V(T_t) + \lambda_t |T_t| \leq V(T) + \lambda_t |T| \Rightarrow \lambda_t \geq \frac{V(T_t) - V(T)}{|T| - |T_t|}$$

  Therefore we choose among all the admissible subtrees T_t the one with the smallest term

$$\frac{V(T_t) - V(T)}{|T| - |T_t|}$$

- At the end of the shrinking process we have a sequence of candidate trees which have to be properly assessed in order to perform the structural selection. As far as validation is concerned, either a procedure of cross-validation or of independent testing can be used.

Dataset



Example