Università di Verona

A.Y. 2021-22

Machine Learning & Artificial Intelligence

Linear Discriminant Functions

Vittorio Murino

Classifiers with linear discriminating functions

- Given a problem with 2 classes, the goal is to create a linear function $g(\mathbf{x})$ that separates the examples belonging to the two classes

  $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$

  where $\mathbf{x}=[x_1,...,x_n]$, $\mathbf{w}=[w_1,...,w_n]$ (weights) and w_0 *bias* (threshold).

- *A sample* $\mathbf{x}_i$ *is classified as belonging to* ω_1

  if $\mathbf{w}^t\mathbf{x}_i + w_0 > 0$, *and to* ω_2 *if* $\mathbf{w}^t\mathbf{x}_i + w_0 < 0$,

- More generally, we consider $\mathbf{w'} = [w_0,w_1,...,w_n]$ and $\mathbf{x'} = [1,x_1,...,x_n]$, therefore the decision rule becomes:

  A sample $\mathbf{x}_i$ *is classified as belonging to* ω_1

  if $\mathbf{w'}^t\mathbf{x}_i' > 0$, *or to* ω_2 *if* $\mathbf{w'}^t\mathbf{x}_i' < 0$

- Suppose we have a set of m samples $\mathrm{x}_1,\ldots,\mathrm{x}_\mathrm{m}$, some tagged ω_1 and others tagged ω_2
- We want to use these samples to determine the weights $\mathbf{w}$ and $\mathbf{w_0}$ of the linear discriminating function
- Suppose also that there is a solution for which the probability of error is very low.
- Thus, a reasonable approach is to look for a weight vector that classifies all of the samples correctly or that the probability of making a mistake is zero.
- If such a weight vector exists, the samples are said to be *linearly separable*.

- The goal is to calculate these weights for which

  $\mathbf{w}^t\mathbf{x}_i > 0$ for each $\mathbf{x}_i$ belonging to ω_1

  $\mathbf{w}^t\mathbf{x}_i < 0$ for each $\mathbf{x}_i$ belonging to ω_2
- In the latter case it can also be said that $\mathbf{x}_i$ is classified correctly if $\mathbf{w}^t(-\mathbf{x}_i) > 0$.
- This suggests a normalization that simplifies the treatment in the case of two different classes, namely, the fact that you can only find the weight vector such that $\mathbf{w}^t\mathbf{x}_i > 0$ for all samples regardless of classes.
- This vector is called a separator vector or a solution vector.

- The weight vector $\mathbf{w}$ can be considered as a point in the weight space.
- Each sample $\mathbf{x}_i$ places a constraint on the possible placement of the solution vector.
- The equation $\mathbf{w}^t\mathbf{x}_i = 0$ defines a *hyperplane* passing through the origin of the weight space having $\mathbf{x}_i$ as a normal vector.
- The solution vector, if it exists, must be in the positive part of each hyperplane and must lie at the intersection of the n half-spaces.
- Thus, each vector in this region is the solution vector and the corresponding region is the solution region.

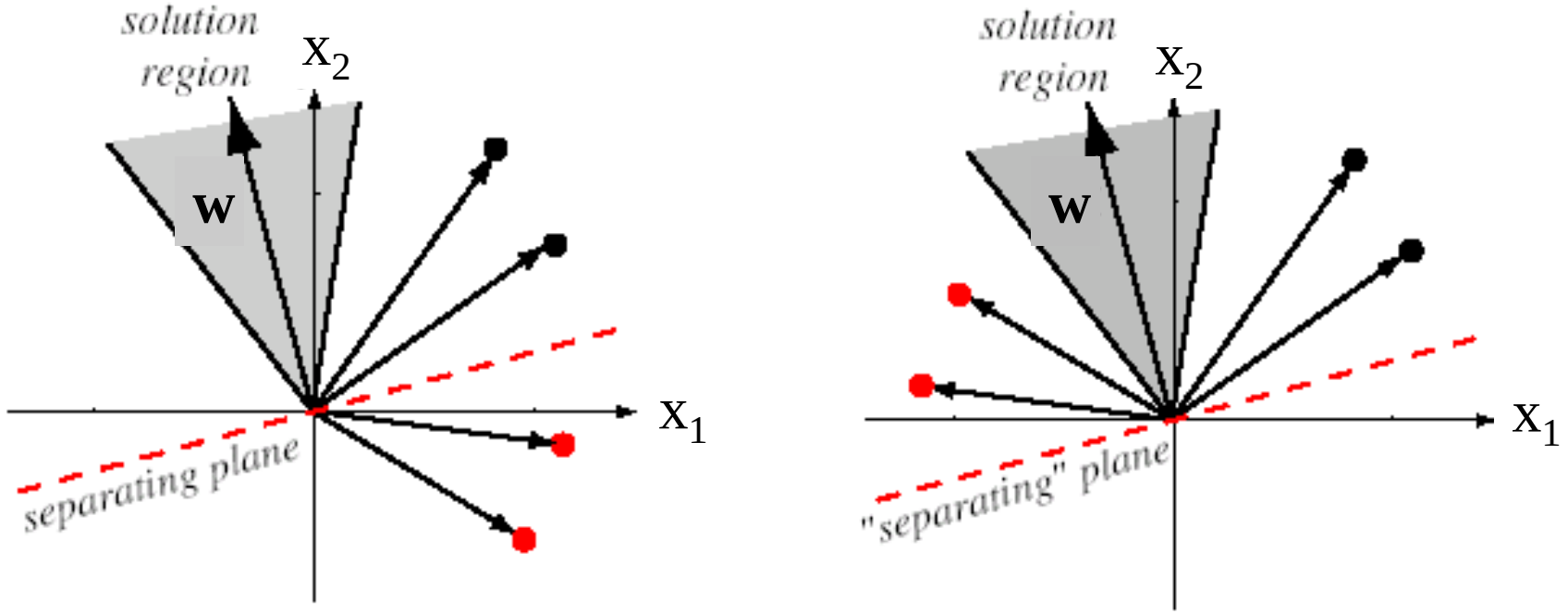


FIGURE 5.8. Four training samples (black for ω_1, red for ω_2) and the solution region in feature space. The figure on the left shows the raw data; the solution vectors leads to a plane that separates the patterns from the two categories. In the figure on the right, the red points have been "normalized"—that is, changed in sign. Now the solution vector leads to a plane that places all "normalized" points on the same side. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- It is clear, therefore, that if the solution vector exists it is not unique.
- There are several ways to impose additional requirements to constrain the solution vector.
- One possibility is to seek a unit-length weight vector that maximizes the minimum distance from the samples to the separating plane.
- Another possibility is to seek the minimum-length weight vector satisfying

$$\mathbf{w}^t\mathbf{x}_i \geq b, \quad \forall i$$

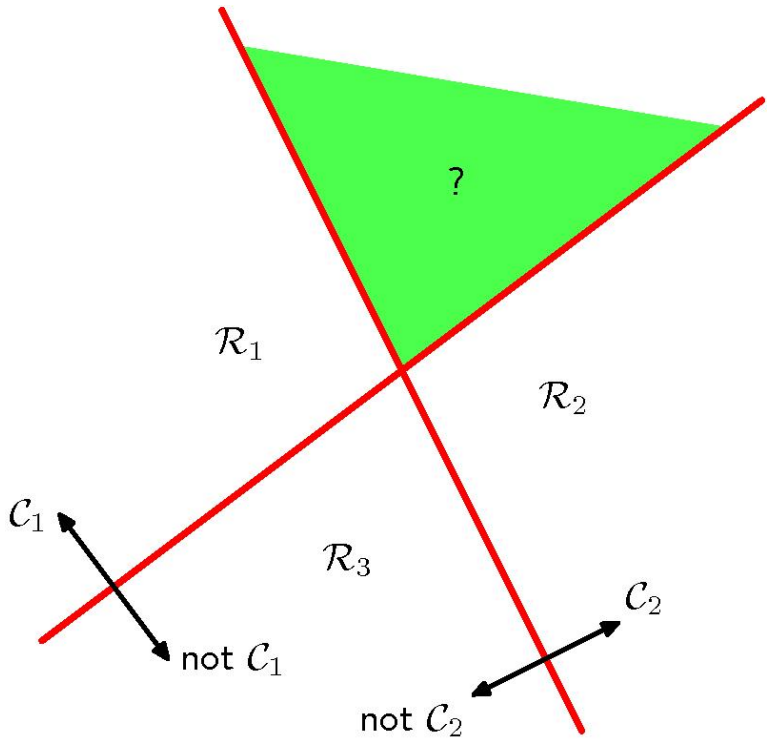
where *b* is a positive constant called *margin*.

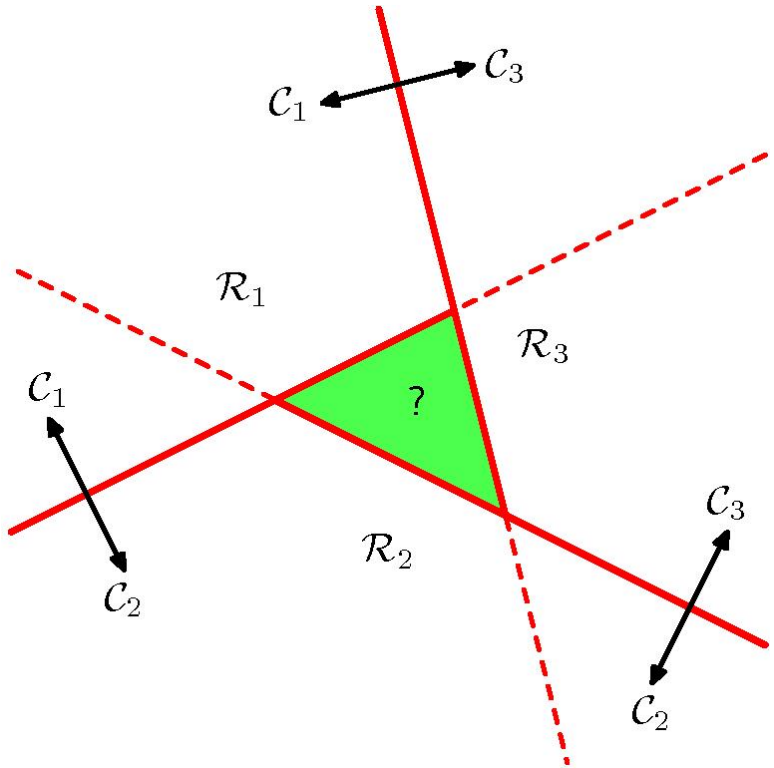
- These tecquiques attempt to find a solution vector closer to the “middle” of the solution region:
  - in this way the resulting solution is more likely to classify new test samples correctly.

- In the case of C-class problems, C-1 classifiers $g_i(\mathbf{x})$ can be constructed, one for each class C_i versus *non*-C_i, called *one-vs-rest* classifiers

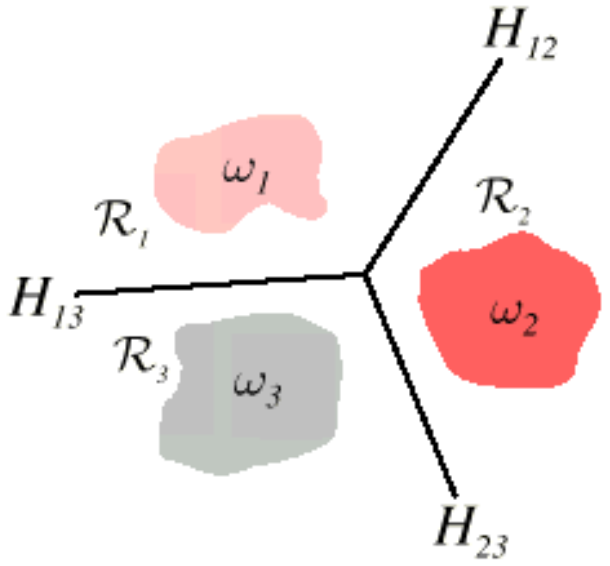
- Or we can build $C(C\text{-}1)/2$ binary classifiers (*one-vs-one*), and then classify a test sample following a majority rule

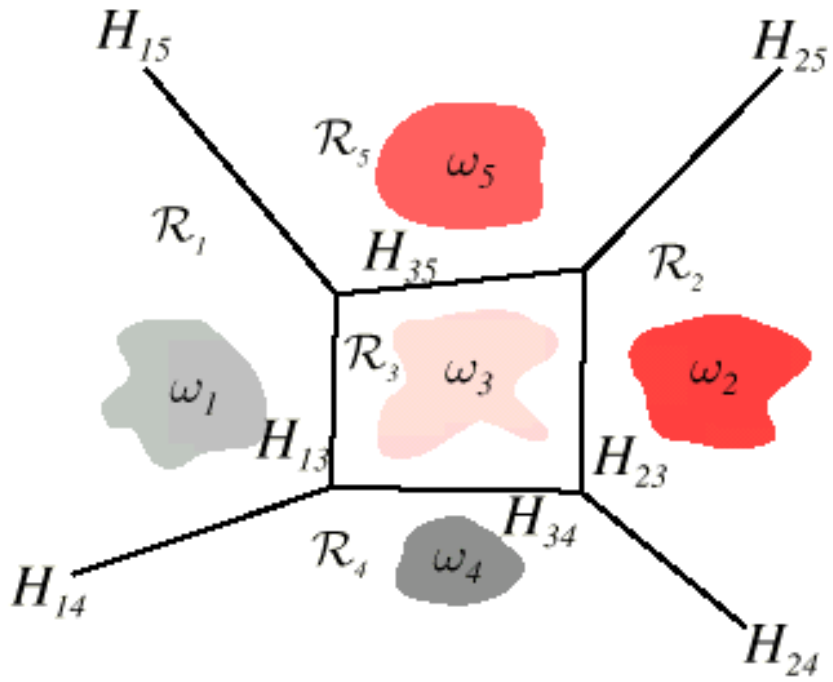
- Both lead to areas of ambiguity.





- It is solved by constructing a single class C classifier that includes C linear functions and then an unknown vector $\mathbf{x}_i$ is assigned to class C_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for each $i \neq j$.





- In general, the *feature* space can be partitioned into sub-regions that coincide with the different recognition classes.
- But the decision surfaces that delimit the sub-regions belonging to different classes can be described by discriminating functions that can be linear, piecewise linear or non-linear.

Linear classifier

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i \cdot x_i$$

where w_i are the elements of the weight vector and w_0 is the threshold weight.

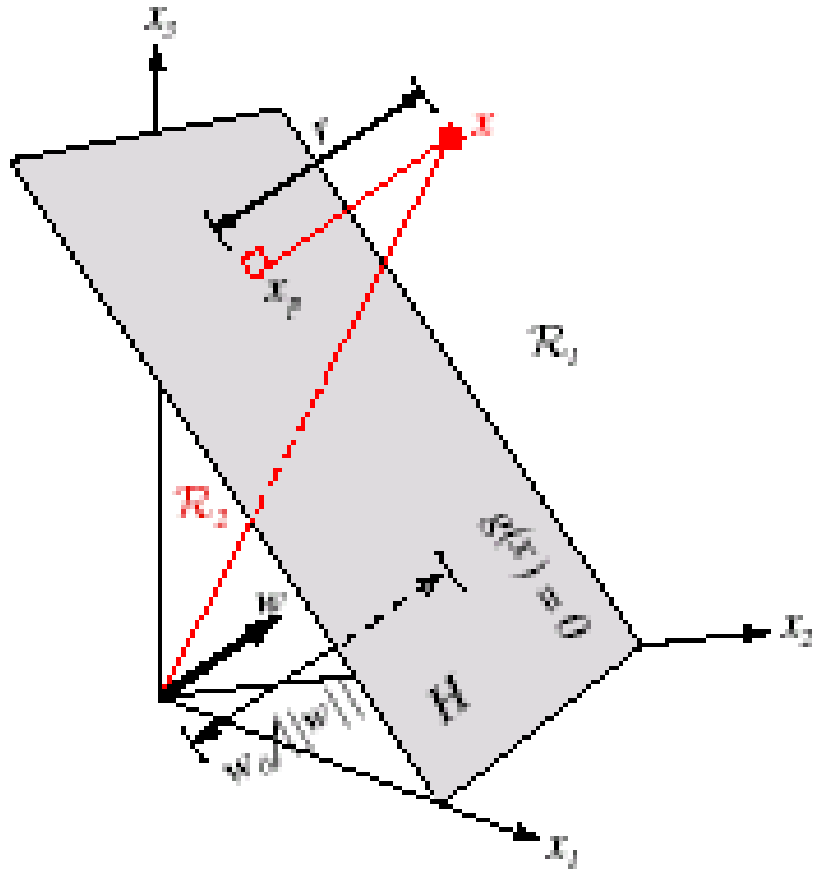
Quadratic classifier

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij} x_i x_j$$

The equation $g(\mathbf{x}) = 0$ determines a decision surface that separates points belonging to different classes.

- In the linear case, a vector $\mathbf{w}$ that defines a hyperplane decision surface should be determined.
- Similarly, in the latter case, other than $\mathbf{w}$, also the matrix $[\mathbf{V}]$ (hyperquadric decision surface) should be determined.
- It can be demonstrated that $\mathbf{w}$ is orthogonal to any vector lying in the hyperplane.
- The discriminant function $g(\mathbf{x})$ gives an algebraic measure if the distance of $\mathbf{x}$ from the hyperplane.
- Thus, a linear discriminant function subdivides the *feature* space by a decision surface represented as a hyperplane.

- Considering the problem from the geometrical point of view, the orientation of this surface is determined by the normal vector $\mathbf{w}$, while the location of the surface is determined by the threshold weight w_0.



- We will mainly deal with the calculation of linear discriminating functions because:
  a) linear functions are the easiest;
  b) every discriminant function can be represented by a piecewise linear function, under a certain approximation (error);
  c) it is possible to carry out non-linear transformations on the features in order to have classification decision surfaces that can be described with linear discriminant functions.

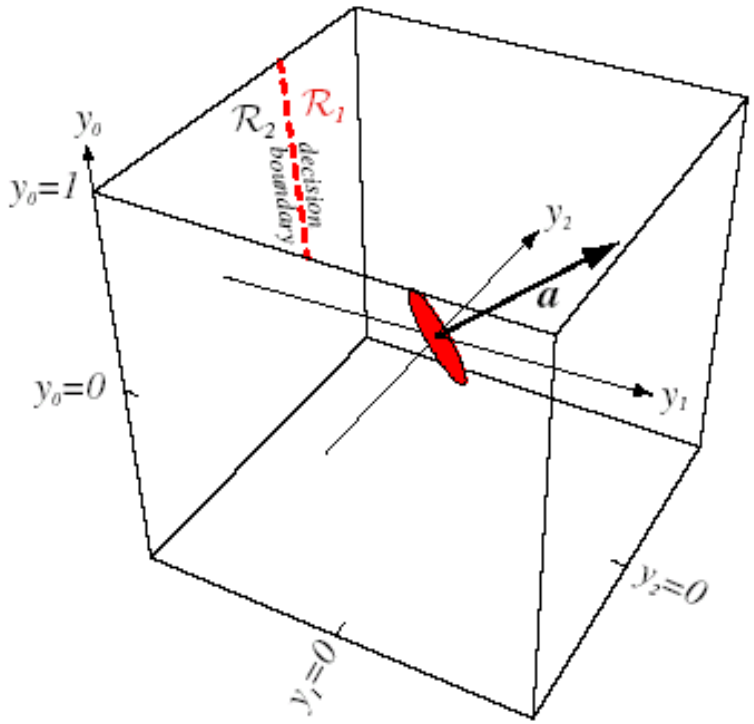
Augmented Vector

- Since in the case of linear classifiers we want an expression of type $\mathbf{w'}^{t}\mathbf{x'} > 0$, $\mathbf{x'}$ must be a vector with an additional component equal to 1 that allows us to determine w_0 along with the other components w_i.

- The problem is then transformed into an homogeneous form:
  - the addition of a component equal for all vectors $\mathbf{x}$ does not change the distance between 2 points, leaving the classification unchanged;
  - the new decision surface (hyperplane) passes through the origin even if the original surface (i.e., the one considering only the original samples without the added component) can be anywhere in the feature space;

- the distance of the new samples from the decision surface is less than or equal to the distance of the original samples from the original decision surface.

- Ultimately, the problem of finding a weights vector **w** and a threshold w_0 is transformed into the problem of finding a single vector of weights **w'**.



x' ←→ y
w' ←→ a

- Thus, the problem of determining discriminant functions given a certain set of training samples, coincides with that of determining $\mathbf{w}$ weights for each class *i*.

- Therefore, if we have two classes ω_1 and ω_2, we expect that it is possible to find a vector $\mathbf{w}$' such that

$$\mathbf{w}'^{t}\,\mathbf{x}_m^1 > 0$$
$$\mathbf{w}'^{t}\,\mathbf{x}_m^2 < 0$$

then, we can associate $\mathbf{x}_m^1$ to the class ω_1 and $\mathbf{x}_m^2$ to the class ω_2.

- Generally, to avoid finding a vector $\mathbf{w}$ with too small components, it is preferred to place a more stringent condition on the vector of the weights associated with the discriminant function of a class, of the type:

$$\mathbf{w}'^{t}\,\mathbf{x}_m^k > b$$

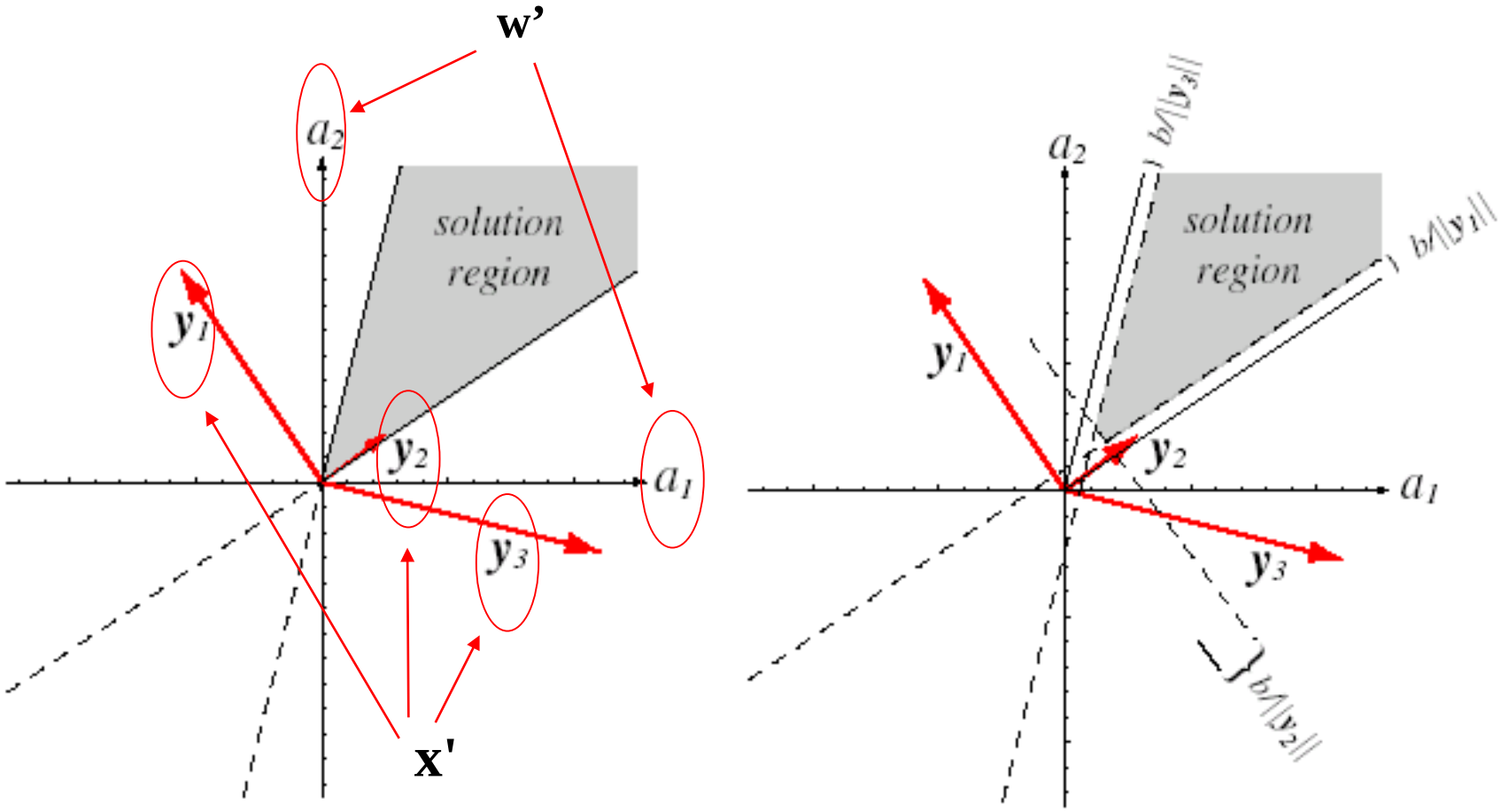
where $b > 0$, b margin (in general there may be a different b for each sample).

- With the introduction of the margin, the solution region lies within the original solution region and is isolated from the old contours of the solution region by the margin.

- Intuitively, the margin should improve the decision region favouring new samples to be classified correctly.

- Usually, $b = 1$ without loss of generality.

- Once we have established the analogy between the two problems, we can see the methods by which the weight vector can be found.



We are in the case of augmented vector

Linear discriminant functions

- Let's consider a function *h* written as:

$$g_i(\mathbf{x}) = h_i(\mathbf{w}, \mathbf{x}') = w_0 + \sum_{k=1}^{n} w_k x_k$$

Gradient Descent

- The gradient descent procedure is one of the is one of the simplest approaches to calculating a discriminant function.
- It is an iterative method of progressive weight adjustment based on the following property:

  The gradient vector in space W points to the direction of maximum deviation of a function to be maximized/minimized.

- The procedure consists of updating the value of the weight vector at the $k+1$ step with a contribution proportional to the gradient module calculated at the previous step and can be formulated as:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \rho_k \nabla J(\mathbf{w})\Big|_{\mathbf{w}=\mathbf{w}(k)} \tag{1}$$

  where $J(\mathbf{w})$ is an evaluation function that must be minimized.

- $J(\mathbf{w})$ is chosen in such a way as to reach the minimum when $\mathbf{w}$ approaches to the optimal solution, that is, should be convex.

- The minimum of $J(\mathbf{w})$ is obtained by moving $\mathbf{w}$ in the direction opposite to the gradient
- ∇ is the gradient operator symbol, given by:

$$\nabla J = \begin{bmatrix} \dfrac{\partial J}{\partial w_1} \\ \vdots \\ \vdots \\ \vdots \\ \dfrac{\partial J}{\partial w_n} \end{bmatrix}$$

- ρ_k is an appropriate scalar value that varies with the iteration k, fixing the "magnitude" of the correction.

- The procedure at the k-th step can be summarized as:
  1) Arbitrarily choose a weights vector $\mathbf{w}(1)$ and calculate the gradient vector $J(\mathbf{w}(1))$;
  2) Obtain a vector $\mathbf{w}(2)$ at a certain distance (also fixed by means of ρ_k) in the direction of maximum descent (*steepest descent*).

- We can investigate how to determine the optimal choice of ρ_k .

- To this end, we should make some more assumptions about the form of $J(\mathbf{w})$.

- Suppose that J can be well approximated by a series of Taylor truncated to the second order, that is:

$$J\left[\mathbf{w}(k+1)\right] \cong J\left[\mathbf{w}(k)\right] + \nabla J\left[\mathbf{w}(k)\right]\left(\mathbf{w}(k+1) - \mathbf{w}(k)\right) + \frac{1}{2}\left(\mathbf{w}(k+1) - \mathbf{w}(k)\right)^t \mathbf{D}\left(\mathbf{w}(k+1) - \mathbf{w}(k)\right) \quad (2)$$

- $\mathbf{D}$ is the Hessian matrix of J (of size $(n+1)\times(n+1)$) that is:

$$\mathbf{D} \Rightarrow D_{ij} = \left.\frac{\partial^2 J}{\partial w_i \partial w_j}\right|_{\mathbf{w}=\mathbf{w}(k)}$$

- Substituting (1) into (2) we have:

$$J\left[\mathbf{w}(k+1)\right] \cong J\left[\mathbf{w}(k)\right] - \rho_k \left\|\nabla J\left[\mathbf{w}(k)\right]\right\|^2 + \frac{1}{2}\rho_k^2 \nabla J^t \mathbf{D} \nabla J \quad (3)$$

- Let's say that at this point we want to find that value of ρ_k for which we have a minimum of the function J at the step k+1.
- This means imposing the following constraint:

$$\frac{\partial J\left[w(k+1)\right]}{\partial \rho_k} = 0 \tag{4}$$

- The condition (4) applied to (3) can be translated into the constraint:

$$-\left|\nabla J\left[\mathbf{w}(k)\right]\right|^2 + \rho_k \nabla J^t \mathbf{D} \nabla J = 0$$

which gives rise to an optimum ρ_k value given by: $$\rho_k = \left.\frac{|\nabla J|^2}{\nabla J^t \mathbf{D} \nabla J}\right|_{\mathbf{w}=\mathbf{w}(k)}$$

- Another possible way to get the solution of our problem – to reach the minimum $J(\mathbf{w})$ - is to look for a minimum of $\mathbf{w}(k+1)$, that is, asking:

$$\frac{\partial J}{\partial \mathbf{w}(k+1)} = 0$$

- Applying this constraint to (2) results in:

$$\nabla J\left[\mathbf{w}(k)\right] \cdot \mathbf{1} + \frac{1}{2}\mathbf{1}^t \mathbf{D}\left(\mathbf{w}(k+1) - \mathbf{w}(k)\right) + \frac{1}{2}\left(\mathbf{w}(k+1) - \mathbf{w}(k)\right)^t \mathbf{D} \cdot \mathbf{1} = 0$$

- Remembering that $D^t = D$ for the simmetry of the Hessian, and the property of the matrices that $A^t D = D^t A$, the following relation is obtained:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mathbf{D}^{-1} \nabla J\Big|_{\mathbf{w}=\mathbf{w}(k)}$$

- In practice, in this case we have $\rho_k = \mathrm{D}^{-1}$ that is the Newton-Raphson algorithm.
- There may be some problems inherent in these ρ_k choices, as, for example, D^{-1} may not exist.
- In addition, matrix inversion is computationally expensive, and must be repeated at every step of the algorithm.
- In addition, the implicit assumpion of second-order surfaces may not be sufficiently precise.
- Therefore, one often prefers to choose ρ_k once for all and thus keeps it constant.
- A different choice is represented by $\rho_k = 1/\mathrm{k}$. In this way:
  a) the convergence rate is increased;
  b) oscillations are avoided.

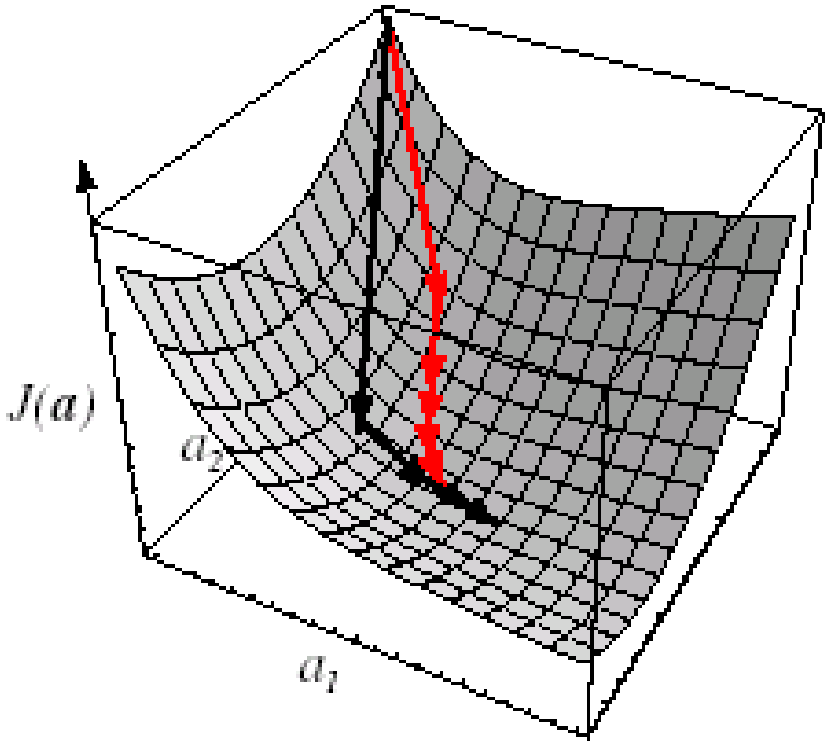


FIGURE 5.10. The sequence of weight vectors given by a simple gradient descent method (red) and by Newton's (second order) algorithm (black). Newton's method typically leads to greater improvement per step, even when using optimal learning rates for both methods. However the added computational burden of inverting the Hessian matrix used in Newton's method is not always justified, and simple gradient descent may suffice. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

The *Perceptron* Method

- Consider now the problem of constructing a criterion function for solving the linear inequalities $\mathbf{w}^t\mathbf{x}_i > 0$.
- The most obvious choice is to let $J(\mathbf{w};\mathbf{x}_1,\ldots,\mathbf{x}_n)$ be the number of samples misclassified by $\mathbf{w}$.
- However, because this function is piecewise constant, the gradient descent procedure is obviously a poor candidate for solving this problem
- A better choice for J can be:

$$J(\mathbf{w}) = -\sum_{i \in X} \mathbf{w}^t \mathbf{x}_i \tag{5}$$

where X is a set of samples *misclassified* by $\mathbf{w}$.

- Geometrically, $J(\mathbf{w})$ is proportional to the sum of the distances of misclassified samples from the decision boundary.
- It converges only if there is a linear separation between the 2 classes: this hypothesis is very strong.
- Since the i-th component of the gradient of $J(\mathbf{w})$ is equal to $\partial J/\partial \mathrm{w_i}$, we can observe from the equation (5) that:

$$\nabla J = -\sum_{i \in X} \mathbf{x}_i$$

- and then the gradient descent algorithm is

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \rho_k \cdot \sum_{i \in X} \mathbf{x}_i, \quad \text{con} \quad \rho_k = \frac{1}{k}$$

- Note that $\sum_{i \in X} \mathbf{x}_i$ does not depend on $\mathbf{w}$,
  and it is a very abrupt corrective factor, and so before we get to the convergence there may be strong fluctuations.

- This method has been used to simulate both *hardware* and *software* the first neural network and then it has evolved to more complex versions such as multilayer *perceptron* (see lecture on Neural Networks)

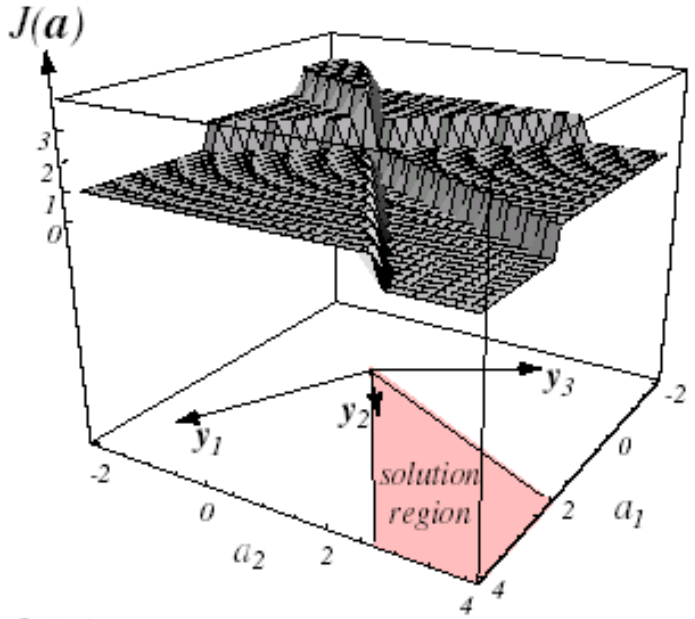
The *Relaxation* Method

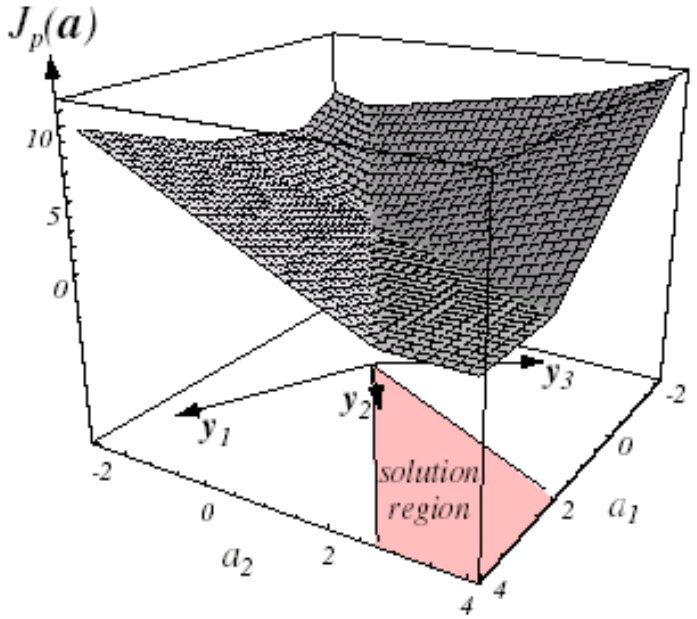
- The function $J(\mathbf{w})$ takes the form

$$J_q(\mathbf{w}) = \sum_{i \in X} \left(\mathbf{w}^t \mathbf{x}_i\right)^2$$

- Also this function takes into account the misclassified samples and it's minimized when $\mathbf{w}$ is the solution vector.
- Its chief difference is that its gradient is now continuous, whereas the gradient of the perceptron function J is not.
- So, such a function is suitable for finding smooth surfaces, but this may lead to problems.
- In fact, in this way the points very far from the origin weigh a lot.
- Another problem is that, experimentally, it is often found that the solution becomes the obvious one, $\mathbf{w} = 0$.

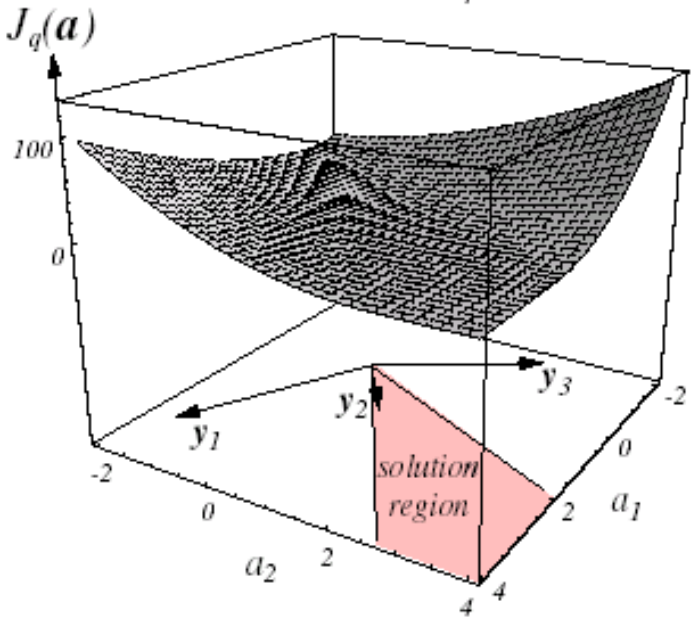
samples wrongly classified

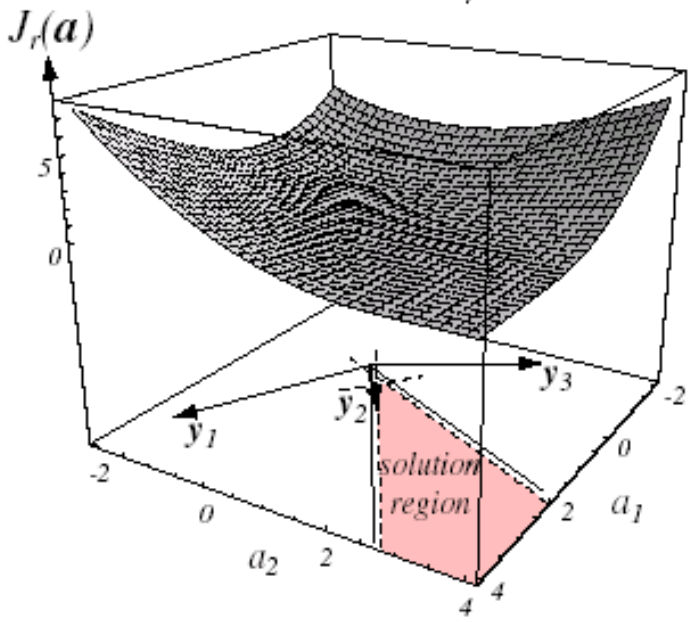




Perceptron

Relaxation





Relaxation with margin

Relaxation with margin

- So, another option for $J(\mathbf{w})$ can be:

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i \in X} \left\{ \frac{\left(\mathbf{w}^t \mathbf{x}_i - b\right)^2}{\|\mathbf{x}_i\|^2} \right\}$$

  where $b \geq 0$ margin (p.es., $b = 1$) and $\mathbf{w}^t\ \mathbf{x}_i \leq b$.

- All the above 3 methods assume linear separability.
- In the latter case then:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \rho_k \sum_{i \in X} \left\{ \frac{\left(\mathbf{w}^t \mathbf{x}_i - b\right)\mathbf{x}_i}{\|\mathbf{x}_i\|^2} \right\}$$

  with $\mathbf{x}_i$ that verify $\mathbf{w}^t\ \mathbf{x}_i \leq b$ and $\mathbf{w}$ is calculated with the usual rule.

- $J(\mathbf{w}) > 0$ always, and $J(\mathbf{w}) = 0$ if $\mathbf{w}^t\ \mathbf{x}_i \geq b \quad \forall \mathbf{x}_i$

The *MSE* Method (*Minimum Square Error*)

- The criterion functions considered so far have focused their attention on the misclassified samples.
- We shall now consider a criterion function that involves *all* of the samples.

- Our goal is to verify $\mathbf{w}^t \mathbf{x}_i = b_i$, where the b_i are some arbitrarily specified positive constants.

- Thus, we have replaced the problem of finding the solution to a set of linear inequalities with the more stringent, but better understood problem of finding the solution to a set of linear equations.

- Suppose to have a certain training set, that is:
  - 1º sample: $x_{1,1}w_1 + x_{1,2}w_2 + \ldots + x_{1,n}w_n + x_{1,n+1}w_{n+1} = b_1, \quad b_1 \geq 0$
  - 2º sample: $x_{2,1}w_1 + x_{2,2}w_2 + \ldots + x_{2,n}w_n + x_{2,n+1}w_{n+1} = b_2, \quad b_2 \geq 0$
  - …….
  - N-th sample: $x_{N,1}w_1 + x_{N,2}w_2 + \ldots + x_{N,n}w_n + x_{N,n+1}w_{n+1} = b_N, \quad b_N \geq 0$

where w_{n+1} is equal to w_0 seen previously and the augmented vector is (change of sign for ω_2):

$x_{i,n+1} = 1$ for ω_1 and $x_{i,n+1} = -1$ for ω_2

- We then obtain N equations (equal to the training samples, N_1 for ω_1 and N_2 for ω_2), with $N = N_1+N_2$.

- So, in matrix notation, we will have:

$$\mathbf{X} \cdot \mathbf{w} = \mathbf{b}$$

where

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} & \pm 1 \\ x_{2,1} & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & \vdots & \vdots \\ x_{N,1} & \cdots & \cdots & x_{N,n} & \pm 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix}$$

- Generally $N >> n+1$, so there are more equations than unknowns. In this case, it is typically not possible to derive an exact solution ($\mathbf{w} = \mathbf{X}^{-1}\mathbf{b}$ has no meaning).
- Let's call now $\mathbf{e}$ the error vector so defined

$$\mathbf{e} = \mathbf{X}\,\mathbf{w} - \mathbf{b}$$

- Let's also write (the quadratic error)

$$J_M(\mathbf{w}) = (\mathbf{X}\,\mathbf{w} - \mathbf{b})^t\,(\mathbf{X}\,\mathbf{w} - \mathbf{b})$$

- I could apply now one of the methods seen previously to calculate w_{k+1}.
- The one based on the calculation of the Hessian is too expensive, so we can apply the first one (eq. (1)) even if it converges more slowly.

- We basically want J_M to have a minimum wrt $\mathbf{w}$, so I calculate the gradient:

$$\nabla_w J_M(\mathbf{w}) = 2\,\mathbf{X}^t\,(\mathbf{X}\,\mathbf{w} - \mathbf{b}) = 2\,(\mathbf{X}\,\mathbf{w} - \mathbf{b})^t\,\mathbf{X} = 0$$

$$\Rightarrow \mathbf{X}^t\,\mathbf{X}\,\mathbf{w} = \mathbf{X}^t\,\mathbf{b}$$

- If we multiply by $[\mathbf{X}^t\,\mathbf{X}]^{-1}$, we get

$$\mathbf{w} = [\mathbf{X}^t\,\mathbf{X}]^{-1}\,\mathbf{X}^t\,\mathbf{b} = \mathbf{X}^{\#}\,\mathbf{b}$$

  where $\mathbf{X}^{\#} = [\mathbf{X}^t\,\mathbf{X}]^{-1}\,\mathbf{X}^t$ is the *pseudoinverse* of $\mathbf{X}$.

- Then $\mathbf{w} = \mathbf{X}^{\#}\,\mathbf{b}$ is the optimal solution according to the MSE method.
- Indeed, $\mathbf{X}^{\#}$ is difficult to obtain: I have to make the inverse of n+1×n+1 matrix.
- On the other hand, it's not an iterative process, that is, I get the solution $\mathbf{w}$ analytically.

- The MSE solution depends on the margin vector $\mathbf{b}$ since different choices of $\mathbf{b}$ lead to different properties of the solution.

- If $\mathbf{b}$ is arbitrarily fixed, there is no evidence that the MSE solution gives a separator vector in the case of linear separability.

- However, it is reasonable to think that, by minimizing the quadratic function, a useful discriminating function can be obtained both in the case of linear and non-linear separability.

- Note that with a careful choice of $\mathbf{b}$, the MSE discriminating function is related to Fisher's linear discriminator

The *Least MSE* (*LMSE*) or *Widrow-Hoff* Method

- As specified above

  $J_M(\mathbf{w}) = (\mathbf{X}\,\mathbf{w} - \mathbf{b})^t\,(\mathbf{X}\,\mathbf{w} - \mathbf{b}) = \|\,\mathbf{X}\,\mathbf{w} - \mathbf{b}\,\|^2$

  can be minimized by a gradient descent procedure.

- This approach has two advantages over the pseudoinverse method:
  1) avoids the problems that arise when $\mathbf{X}^t\mathbf{X}$ is singular,
  2) avoid treating large matrices.

- In addition, the required calculation is in practice an iterative scheme with *feedback* that must automatically cope with computational problems due to rounding and truncation.

- Since $\nabla J_M(\mathbf{w}) = 2\,\mathbf{X}^t\,(\mathbf{X}\,\mathbf{w} - \mathbf{b})$, the gradient descent algorithm is

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \rho_k\,\mathbf{X}^t\,(\mathbf{X}\mathbf{w}_k - \mathbf{b}) \quad \text{with } \mathbf{w}_1 \text{ arbitrary,}$$

- It is easy to show that if $\rho_k = \rho_1/k$ (where ρ_1 is an arbitrary positive constant), then the above equation generates a sequence of weight vectors that converges to a vector given by

$$\mathbf{X}^t\,(\mathbf{X}\,\mathbf{w} - \mathbf{b}) = 0$$

- Thus, the gradient descent algorithm leads to the solution regardless of whether $\mathbf{X}^t\mathbf{X}$ is singular or not.

- So, using this approach and considering one sample at a time, we get

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \rho_k (b_k - \mathbf{w}_k^t \mathbf{x}_k) \mathbf{x}_k$$

called *LMSE* (a.k.a. *Widrow-Hoff*) method.

- b_k is the value for the k-th sample of the margin $\mathbf{b}$ ($b_k \geq 0$), i.e., it depends on the sample $\mathbf{x}_k$.

- This descent algorithm appears similar to that of relaxation: the primary difference between the two is the error correction rule, such that $\mathbf{w}_k^t \mathbf{x}_k$ is always less than b_k, while the Widrow-Hoff rule "corrects" $\mathbf{w}_k$ whenever $\mathbf{w}_k^t \mathbf{x}_k \neq b_k$.

- In many cases, it is impossible to satisfy all the equations $\mathbf{w}_k^t \mathbf{x}_k = b_k$ and therefore the correction never ceases.

- Thus, ρ_k must decrease with k to achieve the convergence of the method.

- This is one of the most widely used techniques for linear classifiers.

- The disadvantage is that we have to iterate at least 3-4 times on all samples namely $K = 3\text{-}4N$.

- For each iteration, we take a different sample, since k is the index of the iteration but also of the samples (i.e., $\mathbf{x}_k$).

Example (Widrow-Hoff): LMSE deterministic

- Given the following two classes :
  ω_1 : (0,0,0,1) (1,0,0,1) (1,0,1,1) (1,1,0,1)
  ω_2 : (0,0,1,1) (0,1,0,1) (0,1,1,1) (1,1,1,1)

and the recursive formula:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \rho_k \left(\mathbf{z}_k - \mathbf{w}_k^t \mathbf{x}_k \right) \mathbf{x}_k$$

- We are already with the augmented vector case, but we does not change the sign to ω_2 (but I wrote $\mathbf{z}_k$ and not $\mathbf{b}_k$).

- At the start, we'll have:
  $\mathbf{w}_1 = \mathbf{0}$, $\rho_k = 1/k$, $\mathbf{z}_k = \pm \mathbf{1}$

- Then:

$$\mathbf{w}_2 = \mathbf{w}_1 + \rho_1\left(1 - \mathbf{w}_1^t\mathbf{x}_1\right)\mathbf{x}_1 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

$$\mathbf{w}_3 = \mathbf{w}_2 + \rho_2\left(1 - \mathbf{w}_2^t\mathbf{x}_2\right)\mathbf{x}_2 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} + \frac{1}{2}\mathbf{x}_2(1-1) = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

$$\vdots$$

- The fact that $\mathbf{w}_3 = \mathbf{w}_2$ is purely casual, and one must still continue the calculations, "cycling" several times over all the samples, it continues to revise $\mathbf{w}$ until $(\mathbf{z}_k - \mathbf{w}_k{}^t\,\mathbf{x}_k) = 0$ for each sample, damping at each iteration by ρ_k.

Other bibliography

- [Gonzalez, 1974]

  J.T.Tou, R.C.Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publ. Comp., Massachusetts, 1974.

- [Fukunaga, 1990]

  K.Fukunaga, *Introduction to Statistical Pattern Recognition*, Second Edition, Academic Press, Inc., Boston, 1990.

- [Duda, 1973]

  R.O.Duda, P.E.Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.