Chapter 5

Linear Discriminant Functions

Introduction

- We shall assume:
  - We know the proper forms for the *discriminant functions* (cf. *probability densities*).
  - We use the samples to estimate the values of parameters of the classifier.
  - None of various procedures for determining discriminant functions requires knowledge of the forms of underlying probability distributions – nonparametric.
- Linear discriminant functions:
  - Linear in the components of $\mathbf{x}$, or
  - Linear in some given set of functions of $\mathbf{x}$.
  - Have pleasant analytical properties
    - Can be optimal if the underlying distributions are cooperative (e.g., Gaussians having equal covariance)
    - We might be willing to sacrifice some performance in order to gain the advantage of the simplicity.
    - Easy to compute.

Introduction

- Problem of finding a linear discriminant function
  - Problem of minimizing a criterion function – training error
  - It is difficult to derive the minimum-risk linear discriminant
    - Several related criterion functions that are analytically more tractable will be investigated.
- Much of our attention will be devoted to studying the convergence properties and computational complexities of various gradient descent procedures for minimizing criterion functions.

Linear discriminant functions and decision surfaces

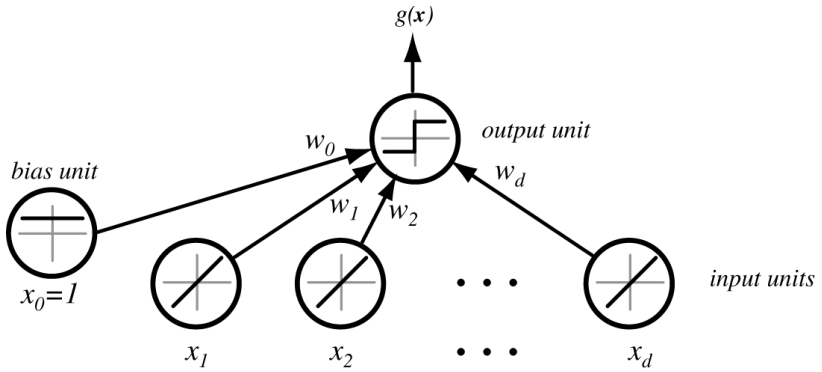
$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + \omega_0$$

weight vector

bias (or threshold weight)

For *c*-category problem, there will be *c* such discriminant functions.

- The Two-Category Case
  - Decide ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) < 0$.
  - $\mathbf{x}$ is assigned to ω_1 if the inner product exceeds the $-\omega_0$ and to ω_2 otherwise.
  - If $g(\mathbf{x}) = 0, \mathbf{x}$ can be ordinarily be assigned to either class.



Linear discriminant functions and decision surfaces

The Two-Category Case (cont.)

- $g(\mathbf{x}) = 0$ defines the decision surfaces.

$$\mathbf{w}^t\mathbf{x}_1 + \omega_0 = \mathbf{w}^t\mathbf{x}_2 + \omega_0 \Rightarrow \mathbf{w}^t(\mathbf{x}_1 - \mathbf{x}_2) = 0$$

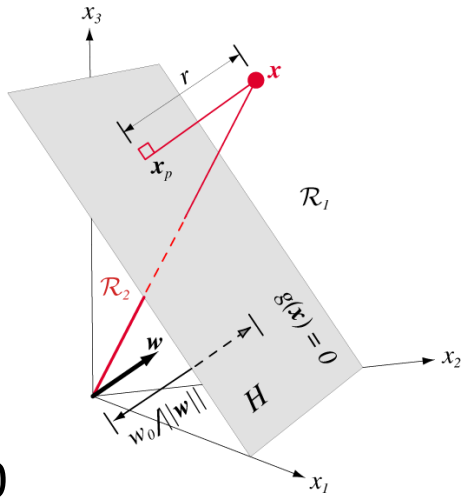
  - $\mathbf{w}$ is normal to any vector lying in the hyperplane (H).
  - H divides the feature space into two regions: R_1 and R_2

- $g(\mathbf{x})$ gives an algebraic measure of the distance from $\mathbf{x}$ to the hyperplane

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

normal projection of $\mathbf{x}$ onto H

the desired algebraic distance



$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + \omega_0 = r\|\mathbf{w}\| \Rightarrow r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|} \quad \text{because } g(\mathbf{x}_p) = 0$$

Linear discriminant functions and decision surfaces

The Two-Category Case (cont.)

$$g(\mathbf{x}_p) = \mathbf{w}^t\mathbf{x}_p + \omega_0 = 0$$

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

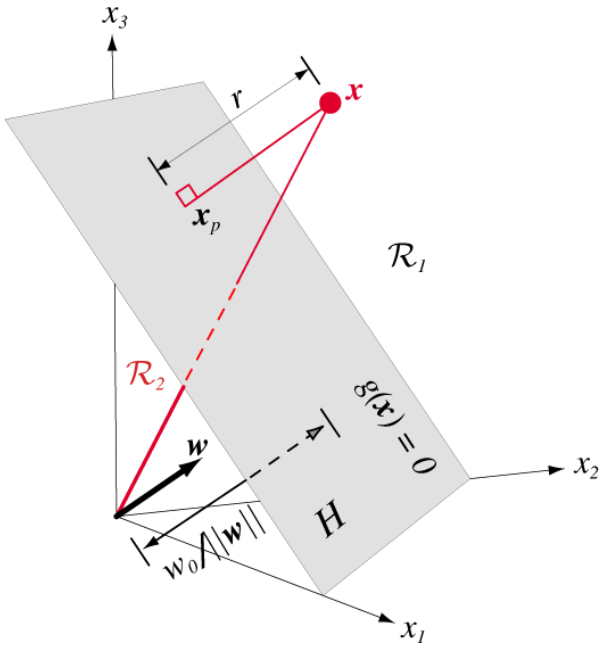
$$\mathbf{x}_p = \mathbf{x} - r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$\begin{aligned} g(\mathbf{x}_p) &= \mathbf{w}^t\left(\mathbf{x} - r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + \omega_0 \\ &= \mathbf{w}^t\mathbf{x} - r\|\mathbf{w}\| + \omega_0 = 0 \end{aligned}$$

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$

The distance from the origin ($\mathbf{x} = \mathbf{0}$) to H

$$r = \frac{g(0)}{\|\mathbf{w}\|} = \frac{\omega_0}{\|\mathbf{w}\|}$$

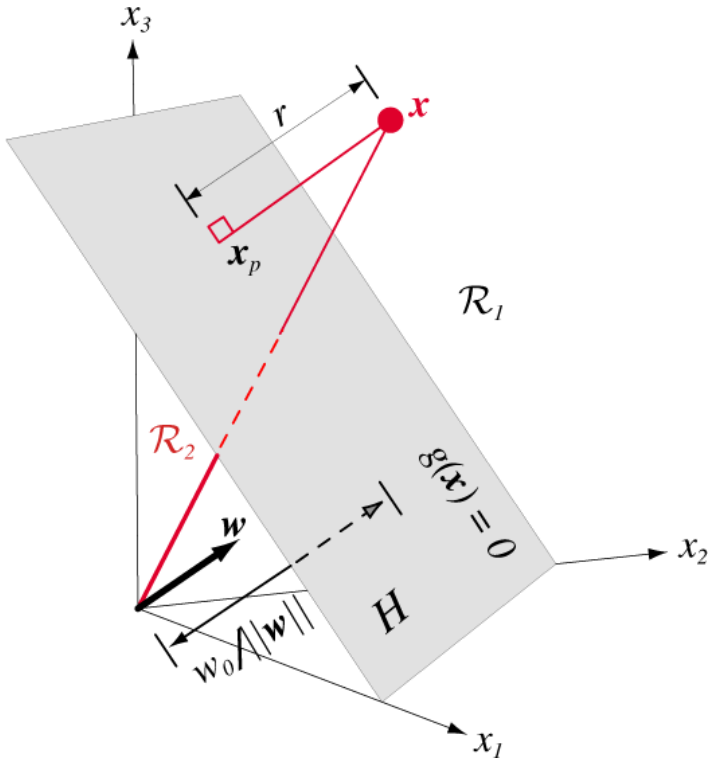


Linear discriminant functions and decision surfaces

The Two-Category Case (cont.)

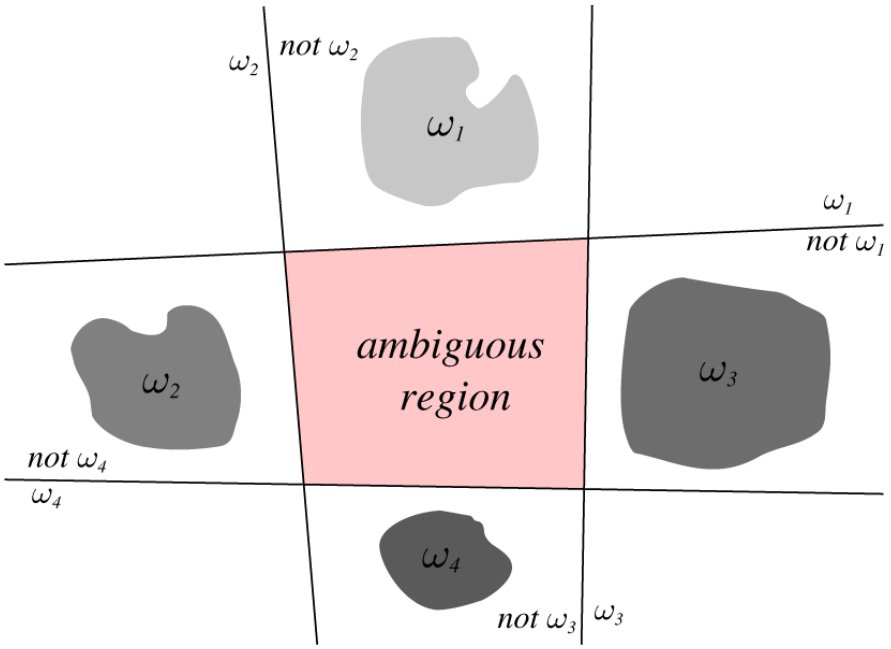
- The distance from the origin to H: $\frac{\omega_0}{\|\mathbf{w}\|}$
  - If $\omega_0 > 0$ the origin is on the positive side of H, otherwise, on the negative side.
  - If $\omega_0 = 0$, H passes through the origin.

- Summary
  - A linear discriminant function divides the feature space by a hyperplane decision surface.
  - The orientation of H is determined by the normal vector $\mathbf{w}$.
  - The location of the H is determined by the bias ω_0.



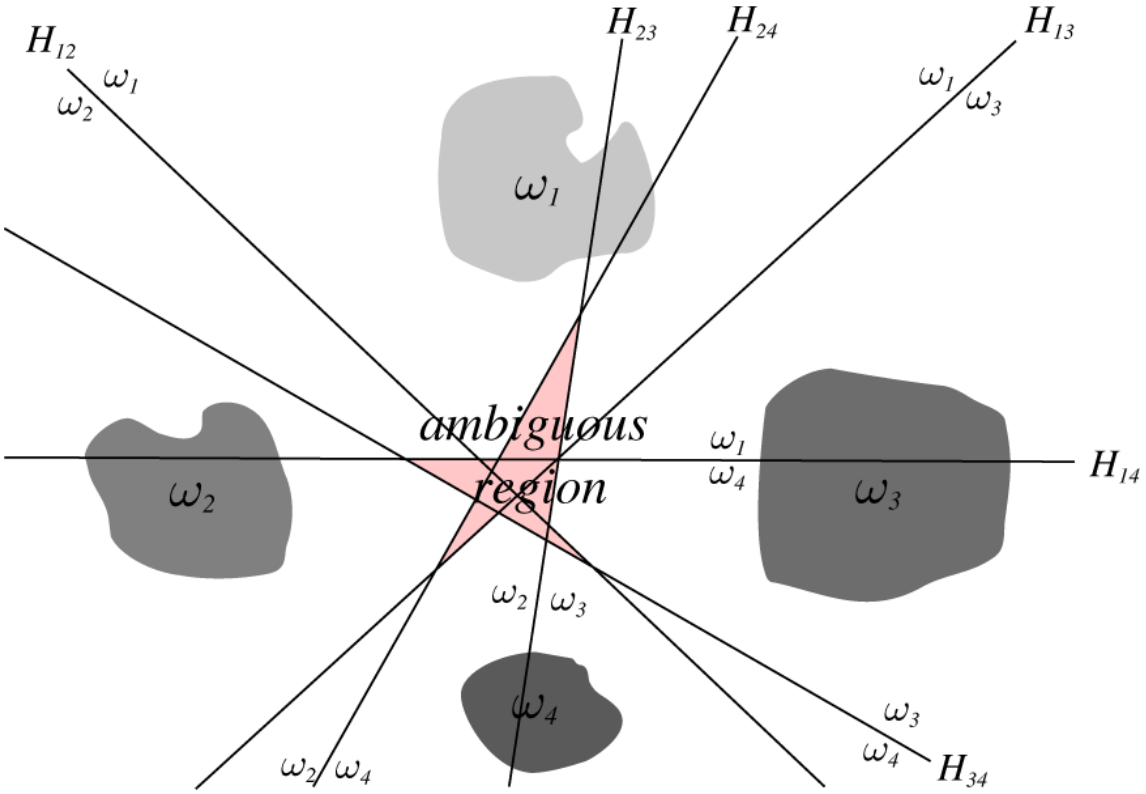
Linear discriminant functions and decision surfaces

- The Multi-Category Case
  - c two-class problems



Linear discriminant functions and decision surfaces

- The Multi-Category Case (cont.)
  - $c(c-1)/2$ linear discriminants (one for every pair of classes)



Linear discriminant functions and decision surfaces

- The Multi-Category Case (cont.)
  - Linear machine

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + \omega_{i0} \quad \text{for } i = 1, \ldots, c$$

    - $\mathbf{x}$ is assigned to ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$.
  - If R_i and R_j are contiguous, the boundary between them is a portion of the hyperplane H_{ij} defined by

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \text{ or } \left(\mathbf{w}_i - \mathbf{w}_j\right)^t \mathbf{x} + \left(\omega_{i0} - \omega_{j0}\right) = 0$$

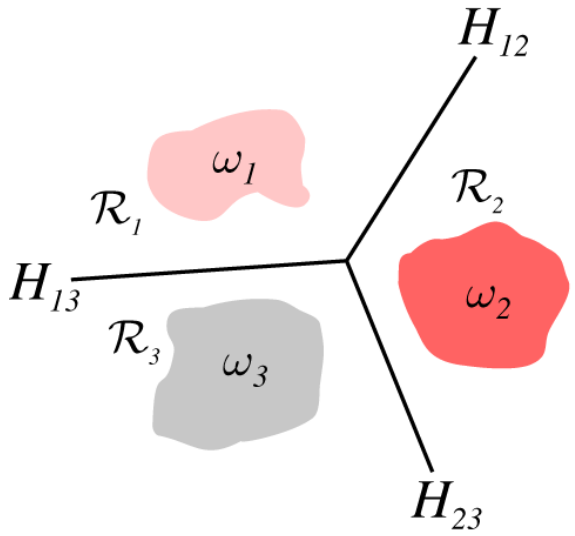
    - $\mathbf{w}_i - \mathbf{w}_j$ is normal to H_{ij}, and the signed distance from $\mathbf{x}$ to H_{ij} is given by

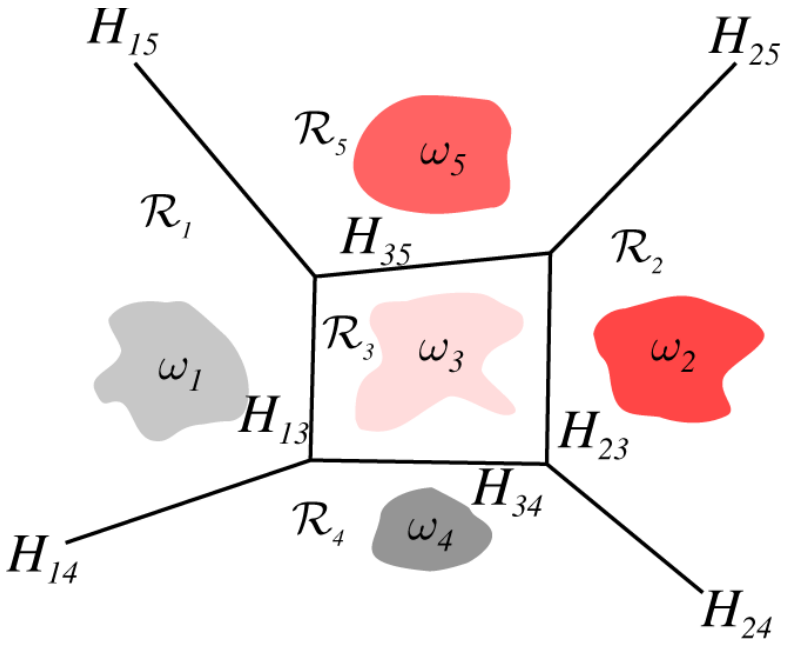
$$\frac{g_i(\mathbf{x}) - g_j(\mathbf{x})}{\left\|\mathbf{w}_i - \mathbf{w}_j\right\|}$$

    - It is not the weight vector themselves but *their differences* that are important.
    - The total number of hyperplane segments in the decision surfaces is often fewer than $c(c-1)/2$.

Linear discriminant functions and decision surfaces

- The Multi-Category Case (cont.)





Generalized linear discriminant functions

- the linear discriminant function

$$g(\mathbf{x}) = \omega_0 + \sum_{i=1}^{d} \omega_i x_i$$

- the quadratic discriminant function

$$g(\mathbf{x}) = \omega_0 + \sum_{i=1}^{d} \omega_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} \omega_{ij} x_i x_j$$

$$\omega_{ij} = \omega_{ji} \text{ because } x_i x_j = x_j x_i$$

  - an additional $d(d+1)/2$ coefficients at its disposal with which to produce more complicated separating surfaces.
- the class of polynomial discriminant functions
  - by continuing to add terms such as $\omega_{ijk} x_i x_j x_k$

Generalized linear discriminant functions

- generalized linear discriminant function

$$g(\mathbf{x}) = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x})$$

$$g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$

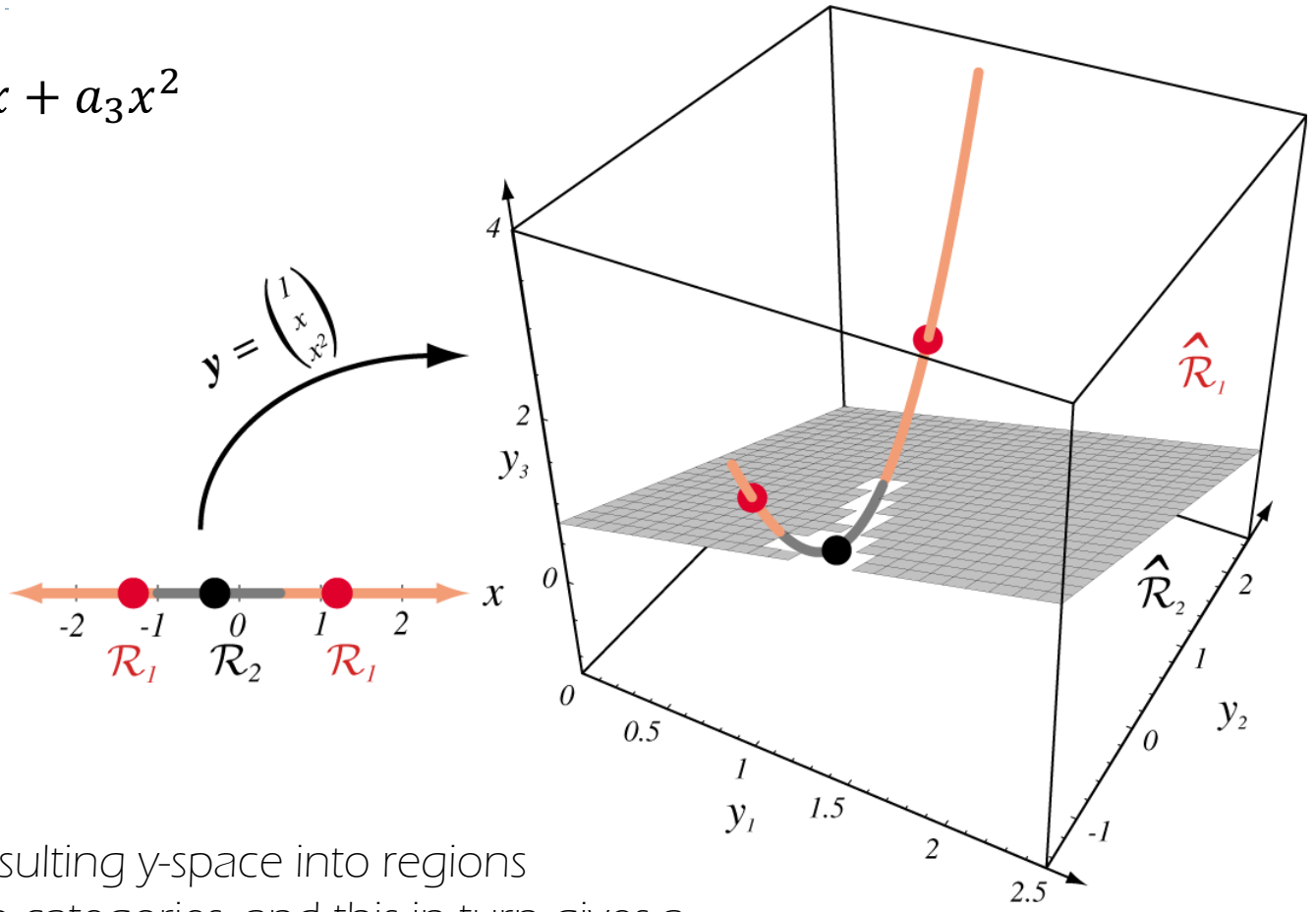
  - $\mathbf{a}$ is now $\hat{d}$-dim weight vector and $\hat{d}$ functions $y_i(\mathbf{x})$ can be arbitrary functions of $\mathbf{x}$.

- the resulting discriminant function is not linear in $\mathbf{x}$, but it is linear in $\mathbf{y}$.
- $\hat{d}$ functions $y_i(\mathbf{x})$ merely map points in d-dim $\mathbf{x}$-space to points in $\hat{d}$-dim $\mathbf{y}$-space.
- the mapping from $\mathbf{x}$ to $\mathbf{y}$ reduces the problem to one of finding a homogeneous linear discriminant function.

Generalized linear discriminant functions

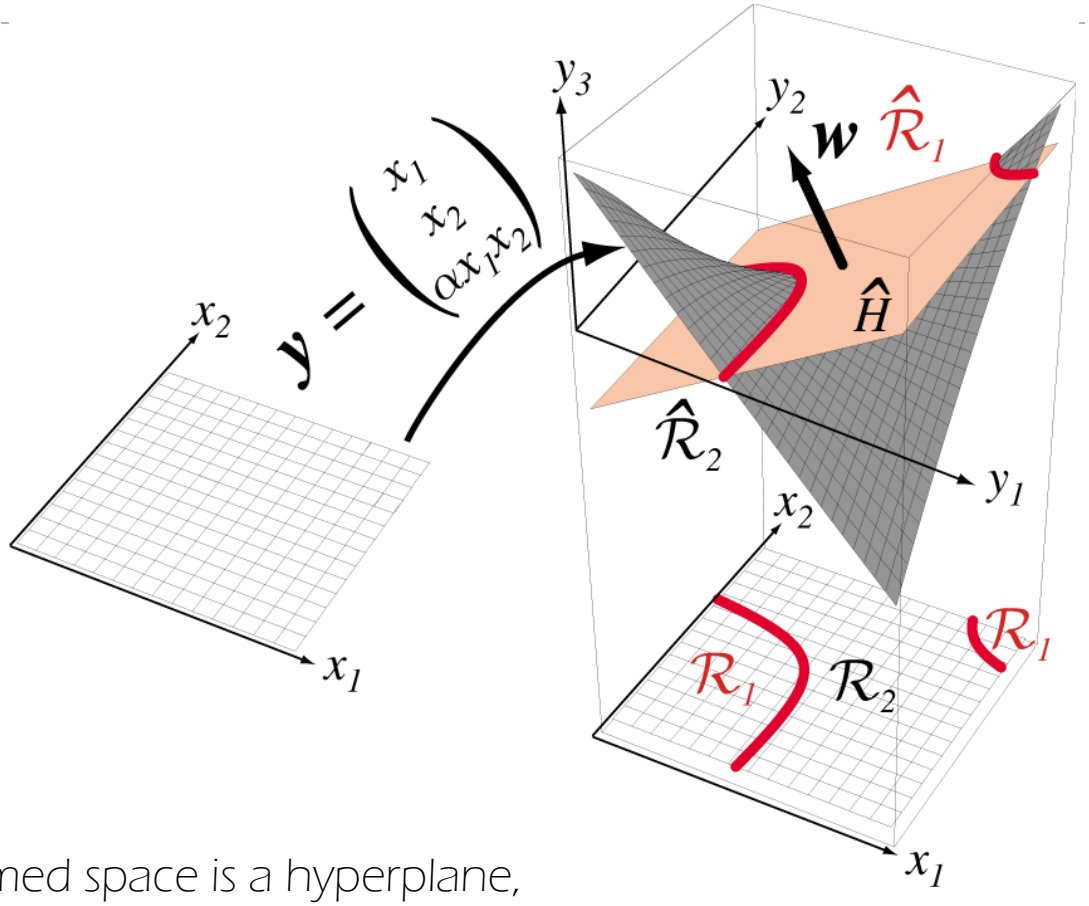
$$g(x) = a_1 + a_2 x + a_3 x^2$$

$$\mathbf{y} = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$



the plane splits the resulting y-space into regions corresponding to two categories, and this in turn gives a nonsimply connected decision region in the one-dimensional x-space.

Generalized linear discriminant functions

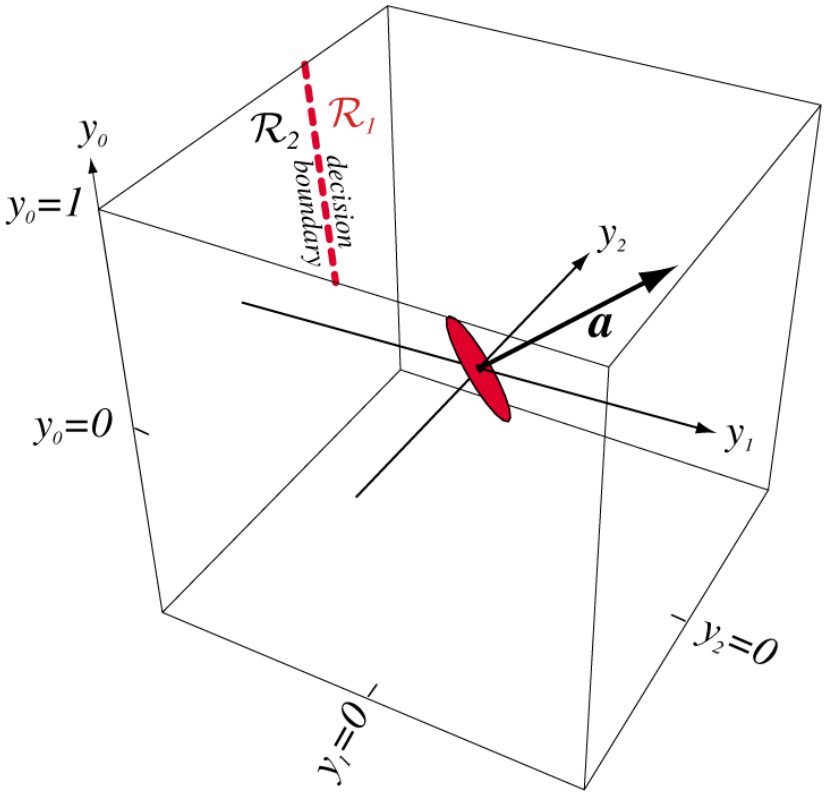


linear discriminant in this transformed space is a hyperplane, which cuts the surface. Points to the positive side of the hyperplane correspond to category w_1, and those beneath it correspond to w_2. Here, in terms of the x space, R_1 is not simply connected.

Generalized linear discriminant functions

$$g(\mathbf{x}) = \omega_0 + \sum_{i=1}^{d} \omega_i x_i = \sum_{i=0}^{d} \omega_i x_i \quad \text{with } x_0 = 1$$

$$g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$



- thus we can write

$$\mathbf{y} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}, \mathbf{a} = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_d \end{bmatrix} = \begin{bmatrix} \omega_0 \\ \mathbf{w} \end{bmatrix}$$

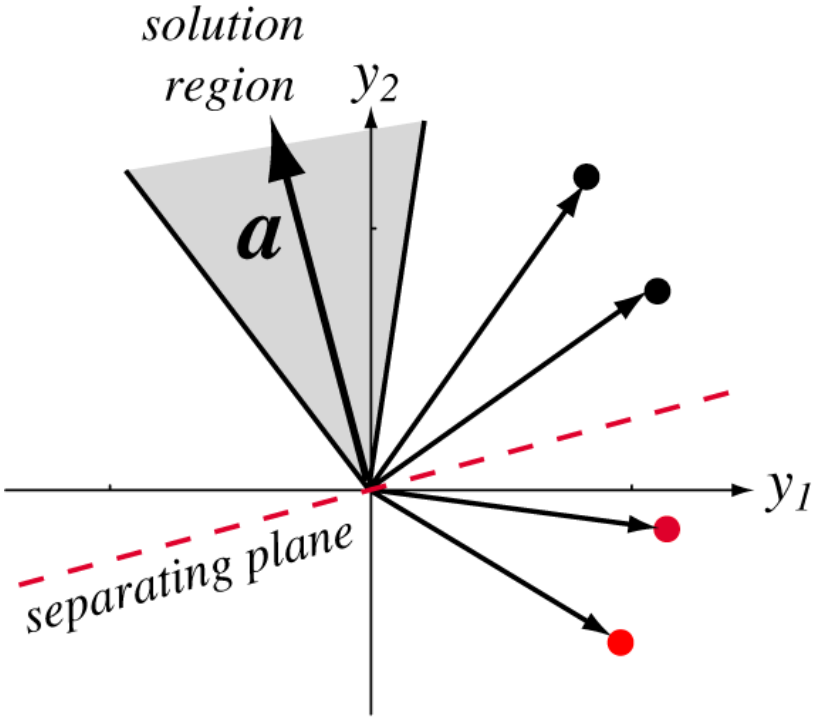
This mapping from d-dimensional x-space to (d+1)-dimensional y-space is mathematically trivial but nonetheless quite convenient.

The two-category linearly separable case

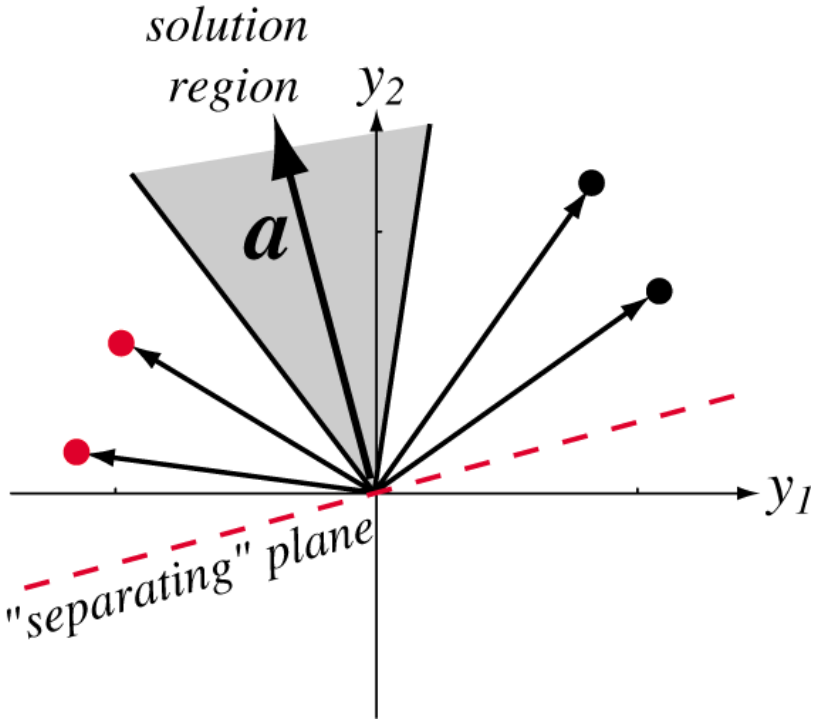
- Linearly Separable
  - Suppose we have reason to believe that there exists a solution for which the probability of error is very low.
    - To look for a weight vector that classifies all of the samples correctly: *if such a weight vector exists the samples are said to be* ***linearly separable.***

- Separating vector
  - A weight vector $\mathbf{a}$ such that $\mathbf{a}^t\mathbf{y}_i > 0$ for all of the samples.

The two-category linearly separable case



The raw data

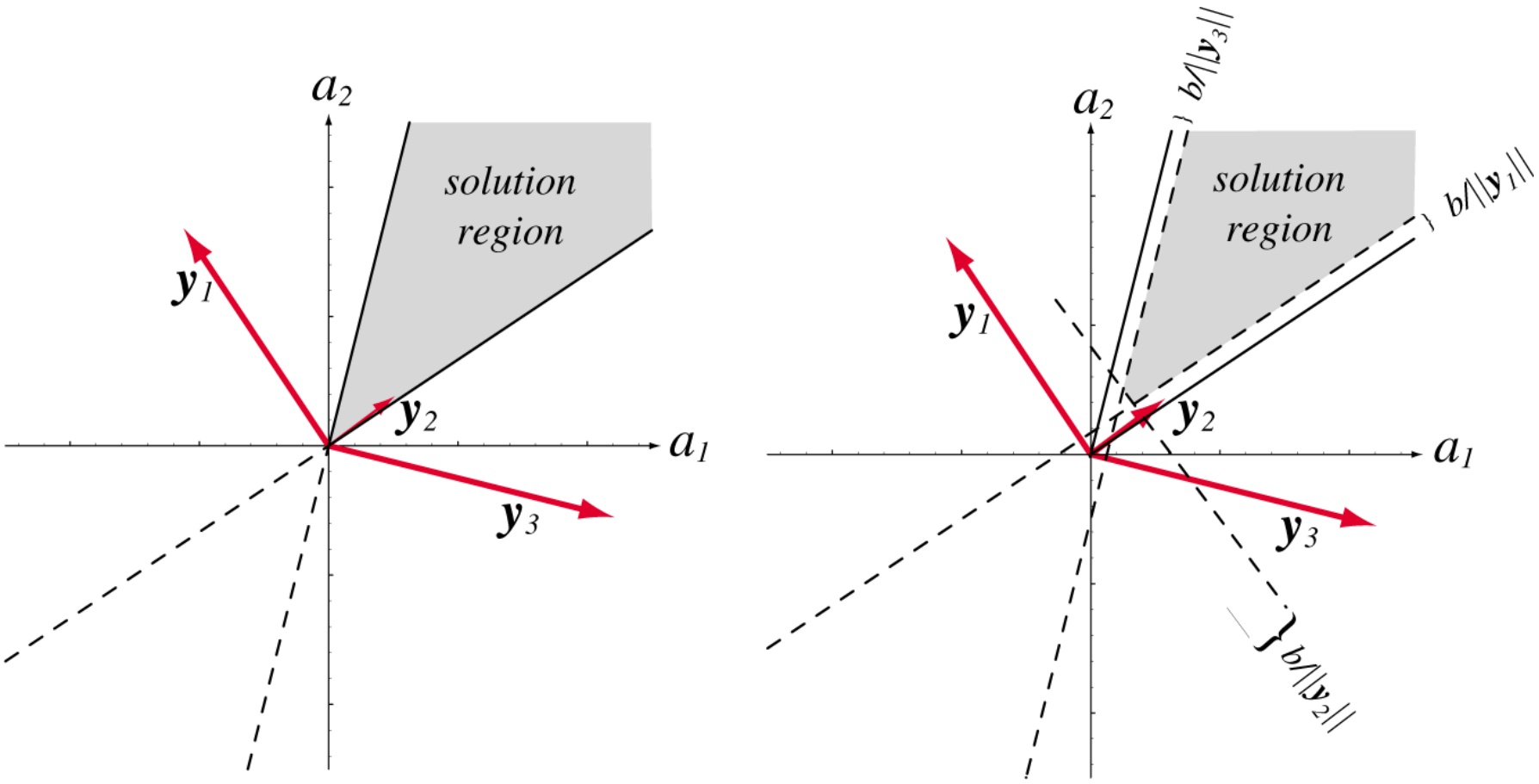


normalized

The two-category linearly separable case

- Weight space
  - The weight vector $\mathbf{a}$ can be thought of as specifying a point in weight space.
    - Each sample $\mathbf{y}_i$ places a constraint on the possible location of a solution vector.
    - The solution vector must be on the positive side of every hyperplane.
- The solution vector is not unique.
  - Several ways to impose additional requirements to constrain the solution vector.
    - to seek a unit-length weight vector that maximizes the minimum distance from the samples to the separating plane.
    - to introduce the margin : $\mathbf{a}^t\mathbf{y}_i \geq b > 0$
  - The motivation to find a solution vector closer to the "middle" of the solution region is the natural belief that the resulting solution is more likely to classify new test samples correctly.

The two-category linearly separable case



The two-category linearly separable case

- Gradient Descent Procedures
  - The approach we shall take to finding a solution to the set of linear inequalities $\mathbf{a}^t\mathbf{y}_i > 0$ will be to define a criterion function $J(\mathbf{a})$ that is minimized if $\mathbf{a}$ is a solution vector.

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k)) \qquad (12)$$

Learning rate

  - We hope that such a sequence of weight vectors will converge to a solution minimizing the criterion function.
  - Algorithm 1 (Basic Gradient Descent) p225
    - Problems associated with the BGD procedures
      - Constructing functions we want to minimize
      - The choice of the learning rate (too small/ too large)

Taylor Series Expansion

We will assume that the performance index is an analytic function, so that all of its derivative exist.

$$F(x) = F(x^*) + \frac{d}{dx}F(x)\Big|_{x=x^*}(x - x^*) + \frac{1}{2}\frac{d^2}{dx^2}F(x)\Big|_{x=x^*}(x - x^*)^2 + \cdots \frac{1}{n!}\frac{d^n}{dx^n}F(x)\Big|_{x=x^*}(x - x^*)^n + \cdots$$

We will use the Taylor series expansion to approximate the performance index, by limiting the expansion to a finite number of terms.

Taylor Series Expansion

Example:

$$F(x) = e^{-x}$$

Taylor series of $F(x)$ about $x^* = 0$:

$$F(x) = e^{-x} = e^{-0} - e^{-0}(x-0) + \frac{1}{2}e^{-0}(x-0)^2 - \frac{1}{6}e^{-0}(x-0)^3 + \cdots$$

$$F(x) = 1 - x + \frac{1}{2}x^2 - \frac{1}{6}x^3 + \cdots$$

Taylor series approximations:

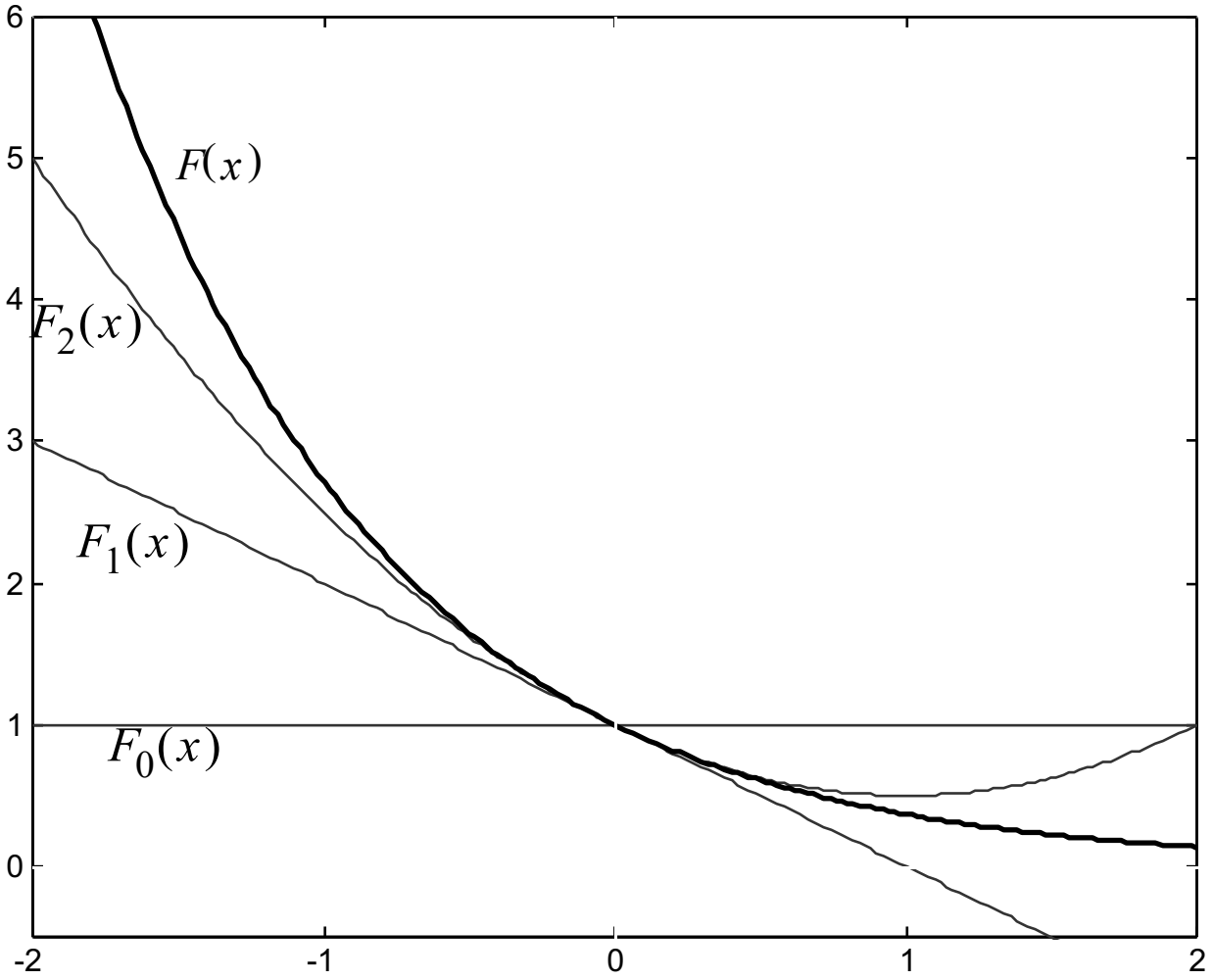
$$F(x) \approx F_0(x) = 1 \qquad 0^{th}\text{-order}$$

$$F(x) \approx F_1(x) = 1 - x \qquad 1^{st}\text{-order}$$

$$F(x) \approx F_2(x) = 1 - x + \frac{1}{2}x^2 \qquad 2^{nd}\text{-order}$$

Taylor Series Expansion

Example (cont.):



Taylor Series Expansion

Vector case:

$$F(\mathbf{x}) = F(x_1, x_2, \ldots, x_n)$$

$$\begin{aligned} F(\mathbf{x}) = F(\mathbf{x}^*) & \\ & + \frac{\partial}{\partial x_1} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*) + \frac{\partial}{\partial x_2} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_2 - x_2^*) + \cdots \\ & + \frac{\partial}{\partial x_n} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_n - x_n^*) + \frac{1}{2}\frac{\partial^2}{\partial x_1^2} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*)^2 \\ & + \frac{1}{2}\frac{\partial^2}{\partial x_1 \partial x_2} F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}^*} (x_1 - x_1^*)(x_2 - x_2^*) + \cdots \end{aligned}$$

Taylor Series Expansion

In matrix form:

$$F(\mathbf{x}) = F(\mathbf{x}^*) + \nabla F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\nabla^2 F(\mathbf{x})^T\Big|_{\mathbf{x}=\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) + \cdots$$

Gradient

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}F(\mathbf{x}) \\ \frac{\partial}{\partial x_2}F(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n}F(\mathbf{x}) \end{bmatrix}$$

Hessian

$$\nabla^2 F(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}F(\mathbf{x}) & \frac{\partial^2}{\partial x_1 \partial x_2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n}F(\mathbf{x}) \\ \frac{\partial^2}{\partial x_2 \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_2^2}F(\mathbf{x}) & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n}F(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1}F(\mathbf{x}) & \frac{\partial^2}{\partial x_n \partial x_2}F(\mathbf{x}) & \vdots & \frac{\partial^2}{\partial x_n^2}F(\mathbf{x}) \end{bmatrix}$$

The two-category linearly separable case

- Gradient Descent Procedures(cont.)
  - A principled method for setting the learning rate
    - 2nd-order expansion around a value $\mathbf{a}(k)$

Hessian matrix

$$J(\mathbf{a}) \approx J(\mathbf{a}(k)) + \nabla J^t(\mathbf{a}-\mathbf{a}(k)) + \frac{1}{2}(\mathbf{a}-\mathbf{a}(k))^t \mathbf{H}(\mathbf{a}-\mathbf{a}(k)) \quad (13)$$

the update rule is

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k))$$

$$J(\mathbf{a}(k+1)) \approx$$

$$J(\mathbf{a}(k)) + \nabla J^t(\mathbf{a}(k))[\eta(k)\nabla J(\mathbf{a}(k))] + \frac{1}{2}[\eta(k)\nabla J^t(\mathbf{a}(k))]\mathbf{H}(\mathbf{a}(k)[\eta(k)\nabla J(\mathbf{a}(k))]$$

$$J(\mathbf{a}(k+1)) \approx J(\mathbf{a}(k)) - \eta(k)\|\nabla J\|^2 + \frac{1}{2}\eta^2(k)\nabla J^t \mathbf{H}\nabla J$$

minimize this with respect to $\eta(k)$

$$0 = \nabla J^t(\mathbf{a}(k))\nabla J(\mathbf{a}(k)) + \eta(k)\nabla J^t(\mathbf{a}(k))\mathbf{H}((\mathbf{a}(k))\nabla J(\mathbf{a}(k))$$

$$\eta(k) = \frac{\|\nabla J\|^2}{\nabla J^t \mathbf{H}\nabla J} \quad (14)$$

The two-category linearly separable case

- Gradient Descent Procedures(cont.)

Newton's algorithm $\mathbf{a}(k+1) = \mathbf{a}(k) - \mathbf{H}^{-1}\nabla J$

Newton's method is based on the second-order Taylor series:

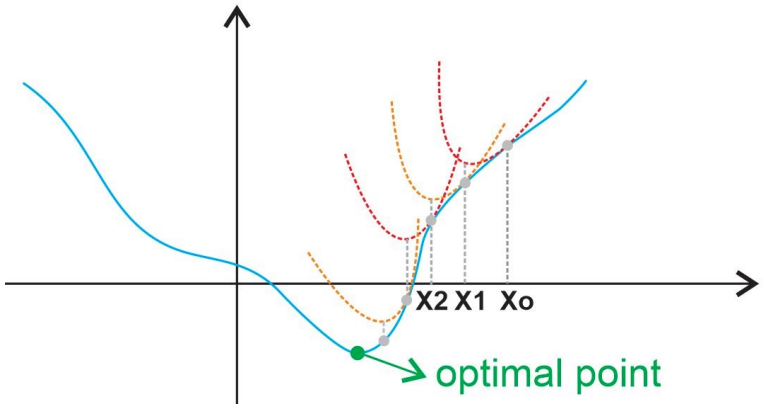
$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx F(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x}_k + \frac{1}{2}\Delta\mathbf{x}_k^T \mathbf{A}_k \Delta\mathbf{x}_k$$

Take the gradient of this second-order approximation and set it equal to zero to find the stationary point:

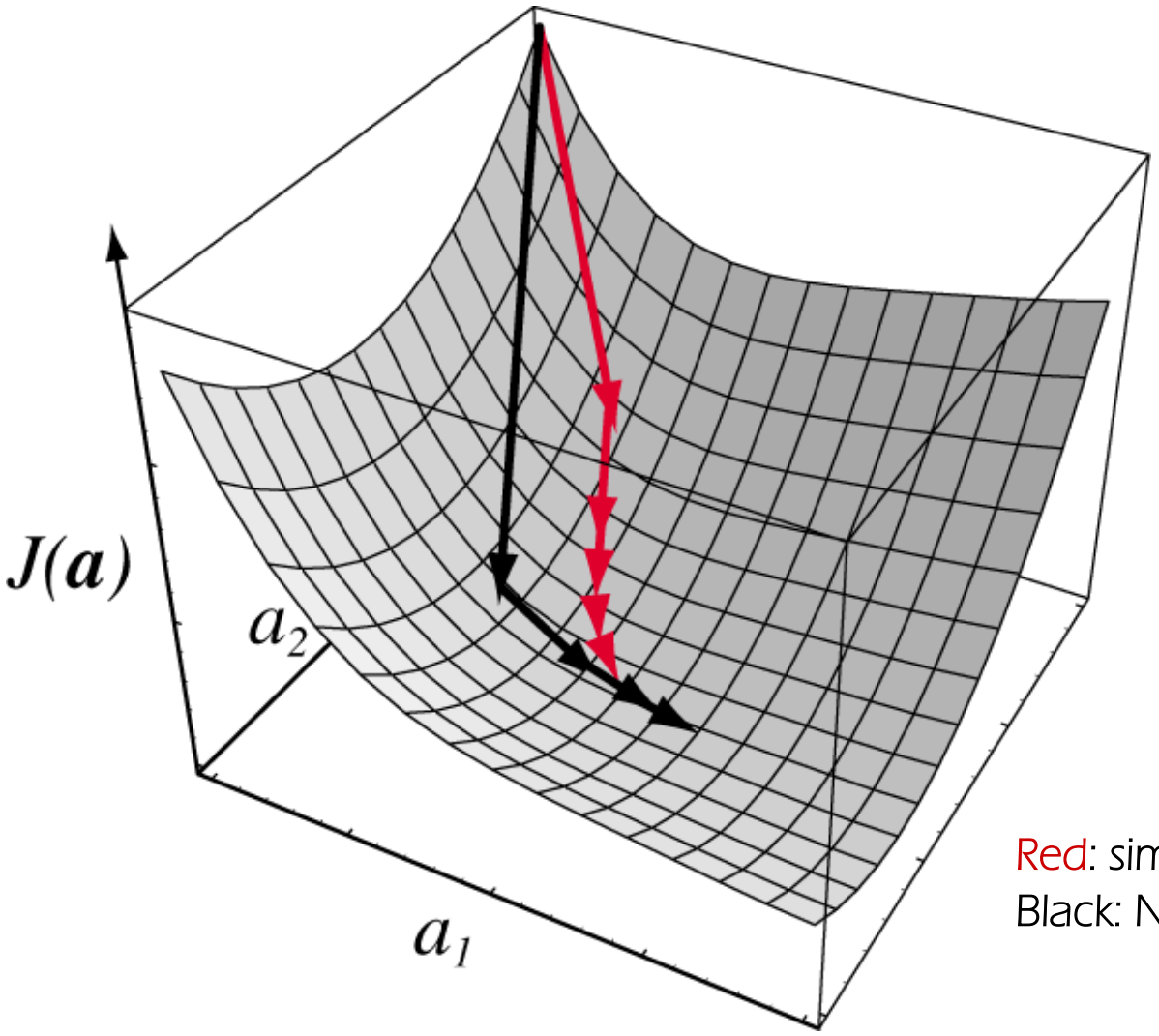
$$\mathbf{g}_k + \mathbf{A}_k \Delta\mathbf{x}_k = \mathbf{0}$$

$$\Delta\mathbf{x}_k = -\mathbf{A}_k^{-1}\mathbf{g}_k$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}_k$$



The two-category linearly separable case

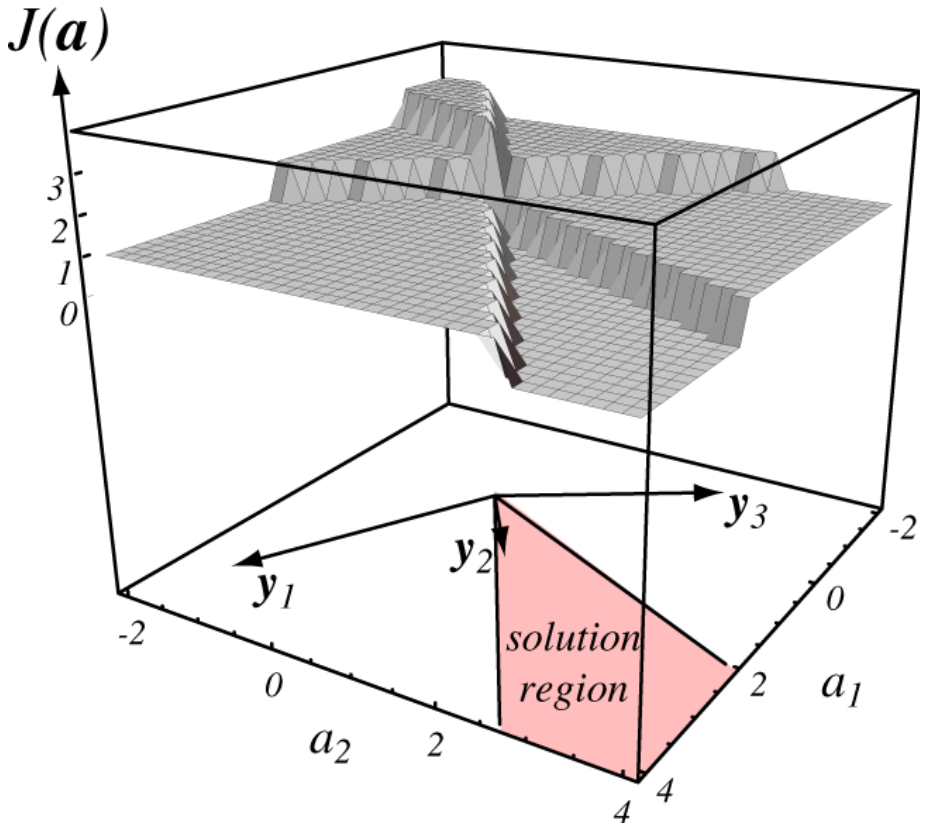


Red: simple gradient descent method
Black: Newton's algorithm

Minimizing the perceptron criterion function

- Problem of constructing a criterion function

  $J(\mathbf{a};\mathbf{y}_1,\cdots,\mathbf{y}_n)$ the number of samples misclassified by $\mathbf{a}$



The function is piecewise constant – a poor candidate for a gradient search.

Minimizing the perceptron criterion function

- Problem of constructing a criterion function(cont.)

$$J_p(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{a}^t\mathbf{y})$$ Perceptron criterion function

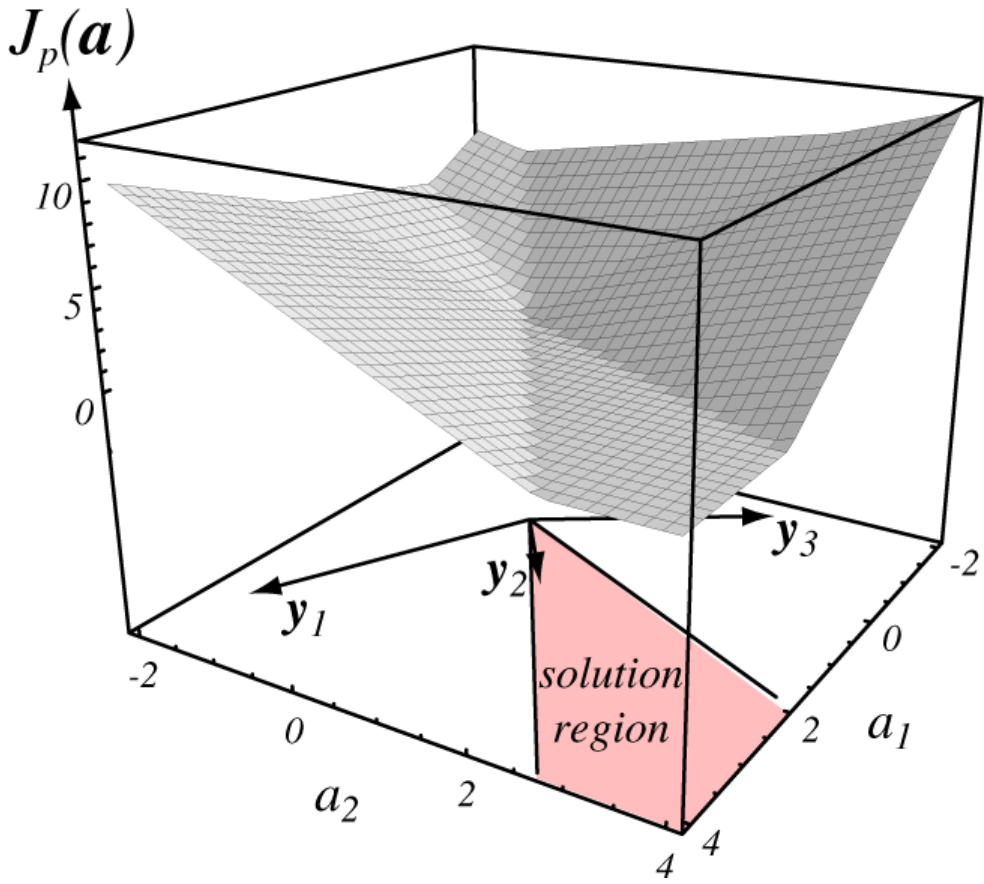
The set of samples misclassified by **a**

Proportional to the sum of the distances from the misclassified samples to the decision boundary

$$\nabla J_p(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{y}) \qquad (17)$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{y}) \quad (18)$$

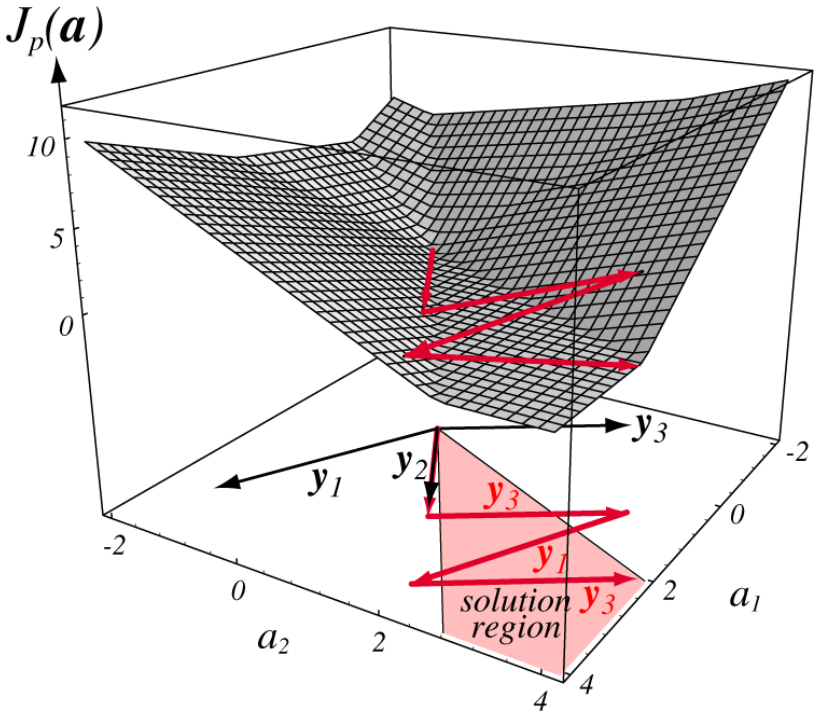
Batch perceptron (Algorithm 3)



Minimizing the perceptron criterion function

- Batch Training
  - The batch perceptron algorithm:
    - The next weight vector is obtained by adding some multiple of the sum of the misclassified samples to the present weight vector.

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k) \sum_{\mathbf{y} \in Y} (-\mathbf{y})$$



The perceptron criterion, $J_p(\mathbf{a})$, is plotted as a function of the weights a_1 and a_2 for a three-pattern problem.
($\mathbf{a}(1)=0$ and $\eta(k)=1$)

Minimizing the perceptron criterion function

- Single-sample correction
  - We shall consider the samples in a sequence and shall modify the weight vector whenever it misclassifies a *single* sample.

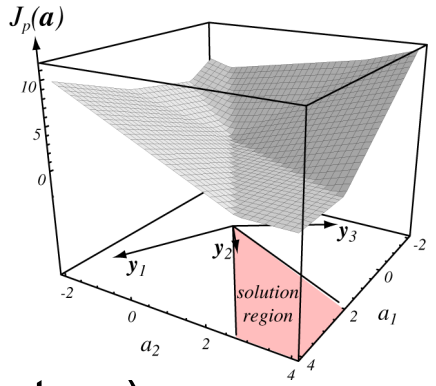
  - Fixed increment rule

$$\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_1, \mathbf{y}_2, \cdots \qquad \mathbf{y}^1, \mathbf{y}^2, \mathbf{y}^3, \mathbf{y}^4, \mathbf{y}^5, \cdots \Rightarrow \mathbf{y}_1, \mathbf{y}_3, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_2, \cdots$$

$$\mathbf{a}(1)$$

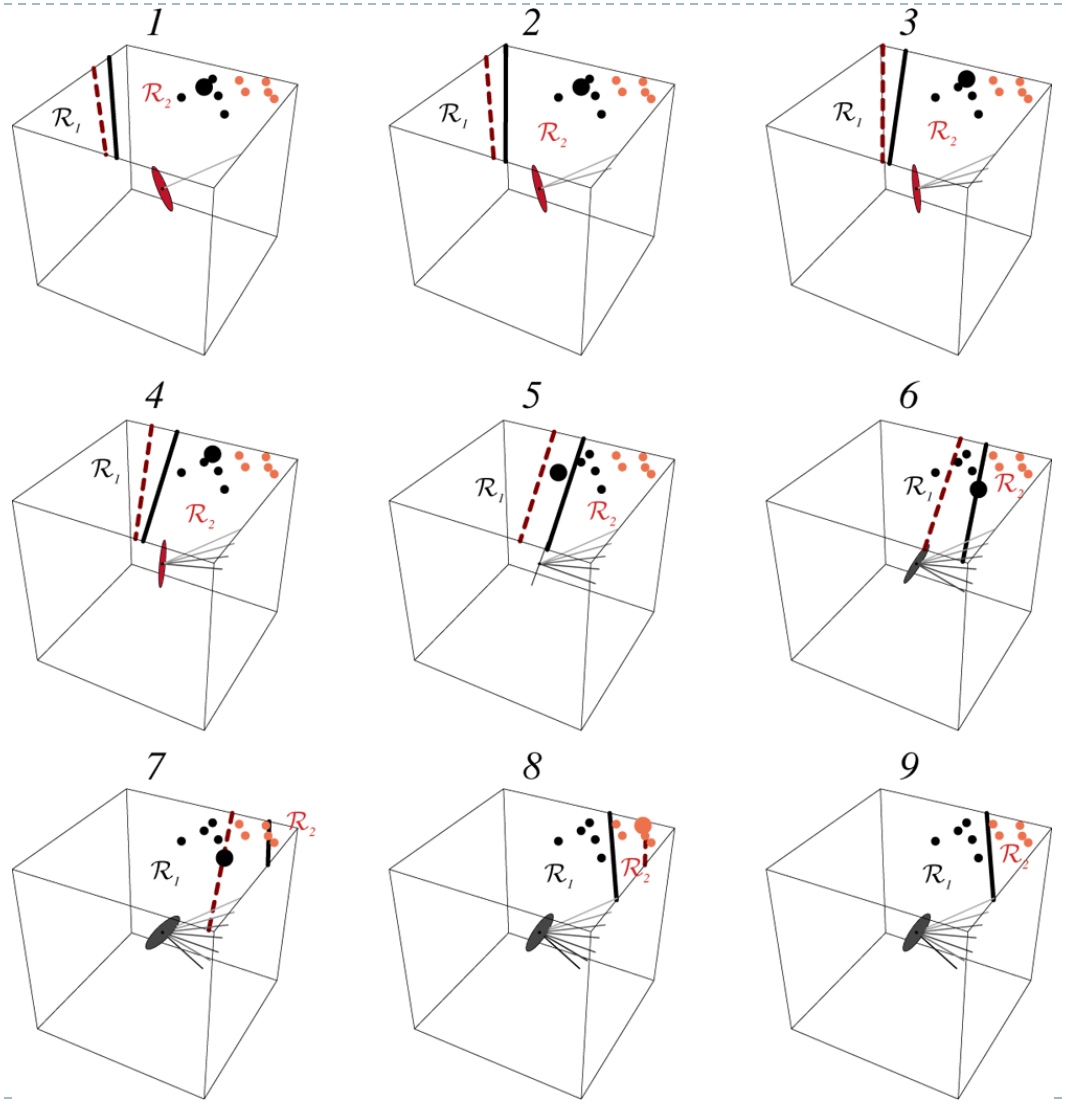
$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k \qquad (20)$$

$$\text{where } \mathbf{a}^t(k)\mathbf{y}^k \le 0 \text{ for all } k$$



Algorithm 4: (Fixed-increment single-sample perceptron)

Minimizing the perceptron criterion function



Minimizing the perceptron criterion function

- Some Direct Generalizations
  - Algorithm 5 (Variable-increment perceptron with margin)
    - introduction of a variable increment $\eta(k)$ and a margin b
    - $\eta(k)$: positive constant or decreases as $1/k$

$$\mathbf{a}(1)$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k$$

$$\text{where } \mathbf{a}^t(k)\mathbf{y}^k \le b \text{ for all } k$$

Minimizing the perceptron criterion function

- Some Direct Generalizations (cont.)
  - Algorithm 6: (Batch Variable increment perceptron)
    - The trajectory of the weight vector is smoothed.

$$\mathbf{a}(1)$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\sum_{y \in Y_k} \mathbf{y}$$

  - Generally speaking, one would prefer to have $\eta(k)$ become smaller as time goes:
    - This is particularly true if there is reason to believe that the set of samples is not linearly separable, because it reduces the disruptive effect of a few "bad" samples.

Relaxation procedures

- To include a broader class of criterion functions and methods for minimizing them.

- The Descent Algorithm

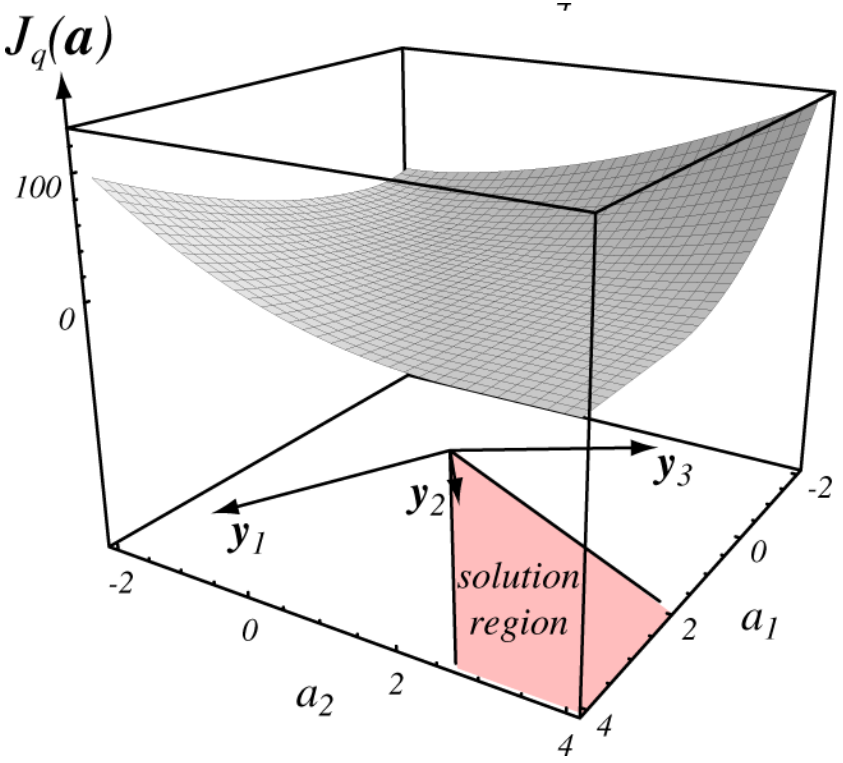
$$J_q(\mathbf{a}) = \sum_{\mathbf{y}\in Y} (\mathbf{a}^t\mathbf{y})^2$$

difference: the gradient is continuous

Problem:

1. **a=0**
2. The value can be dominated by the longest sample vectors.

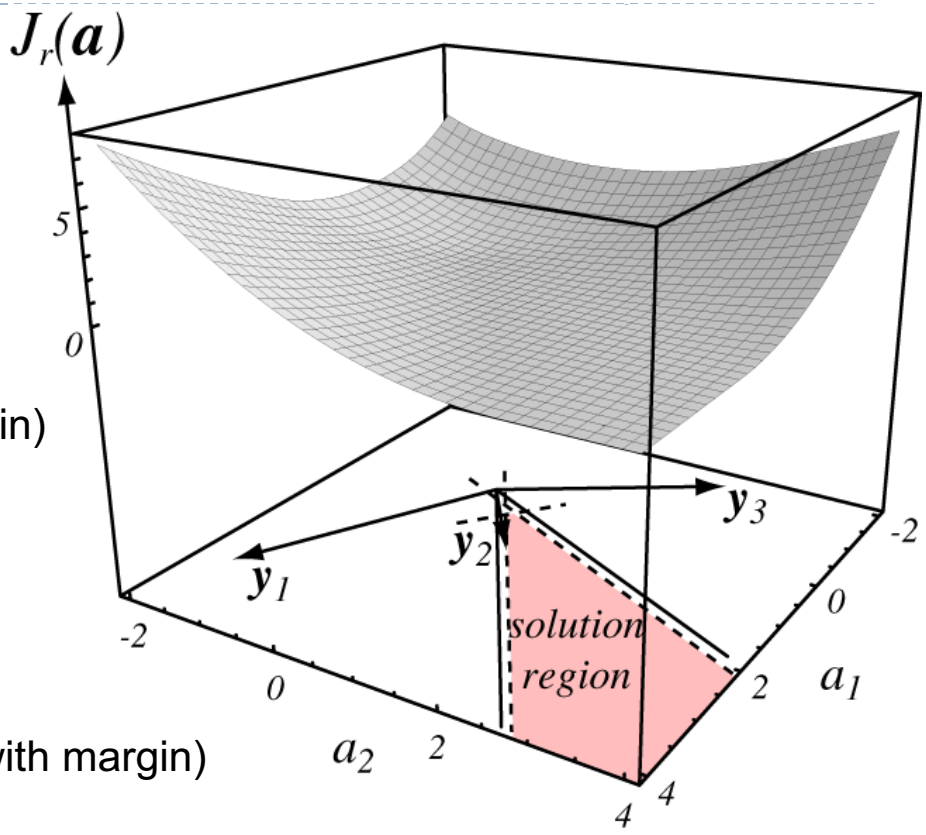
$$J_r(\mathbf{a}) = \frac{1}{2}\sum_{\mathbf{y}\in Y} \frac{(\mathbf{a}^t\mathbf{y} - b)^2}{\|\mathbf{y}\|^2}$$



Relaxation procedures

$$J_r(\mathbf{a}) = \frac{1}{2}\sum_{\mathbf{y}\in Y}\frac{(\mathbf{a}^t\mathbf{y}-b)^2}{\|\mathbf{y}\|^2}$$

$$\nabla J_r(\mathbf{a}) = \sum_{\mathbf{y}\in Y}\frac{\mathbf{a}^t\mathbf{y}-b}{\|\mathbf{y}\|^2}\mathbf{y}$$



Algorithm 8 (Batch relaxation with margin)

$\mathbf{a}(1)$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\sum_{y\in Y}\frac{b-\mathbf{a}^t\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}$$

Algorithm 9 (Single-sample relaxation with margin)

$\mathbf{a}(1)$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\frac{b-\mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|^2}\mathbf{y}^k$$

Relaxation procedures

- Single-sample relaxation rule with margin

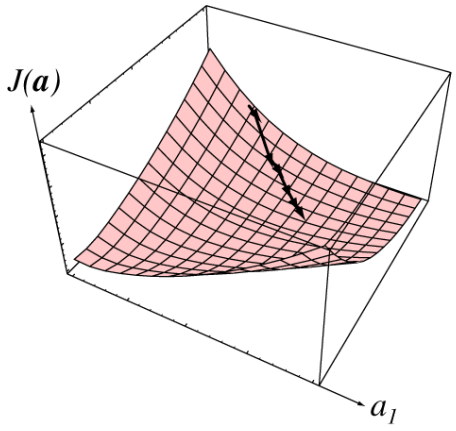
$$r(k)=\frac{b-\mathbf{a}^t(k)\mathbf{y}^k}{\left\|\mathbf{y}^k\right\|}$$ the distance from $\mathbf{a}(k)$ to the hyperplane $\mathbf{a}^t\mathbf{y}_k=b$

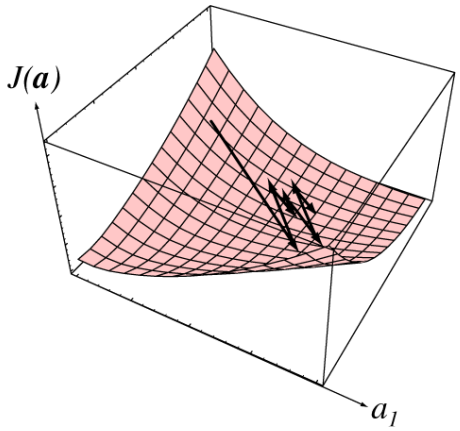
$$\mathbf{a}(1)$$

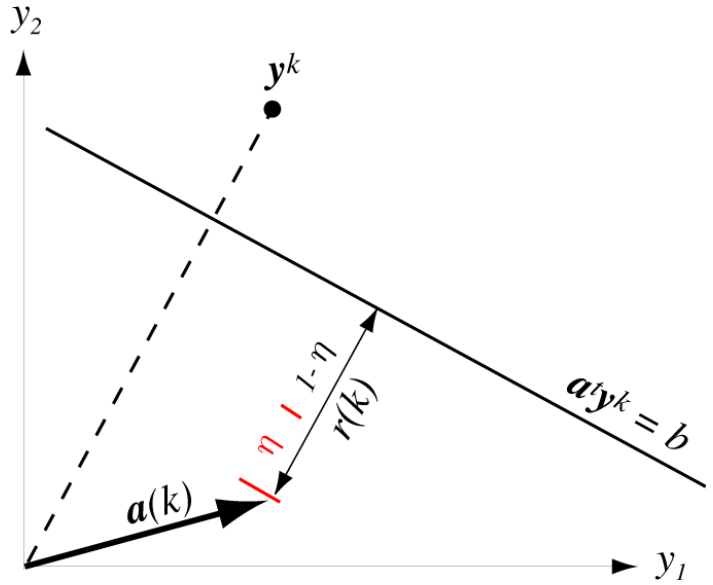
$$\mathbf{a}(k+1)=\mathbf{a}(k)+\eta(k)\frac{b-\mathbf{a}^t(k)\mathbf{y}^k}{\left\|\mathbf{y}^k\right\|^2}\mathbf{y}^k$$

$$\mathbf{a}^t(k+1)\mathbf{y}^k-b=(1-\eta)(\mathbf{a}^t(k)\mathbf{y}^k-b)$$

- under-relaxation: $\eta < 1$
- over-relaxation: $\eta > 1$







Nonseparable behavior

- Unfortunately, sufficiently large design sets are almost certainly not linearly separable.
  - How the error-correction procedures will behave when the samples are nonseparable.
  - The corrections in an error-correction procedure can never cease.
- A number of heuristic modifications to the error correction rule
  - the use of a variable increment $\eta(k)$, with $\eta(k)$ approaching zero as k approaches infinity.
    - to choose $\eta(k)$ is to make it a function of recent performance, decreasing it as performance improves.
    - to program $\eta(k)$ by a choice such as $\eta(k) = \eta(1)/k$.

Minimum Squared-Error procedures

- A criterion function that involves all of the samples.
  - The criterion function we have considered thus far have focused on the *misclassified* samples.
    - a criterion function that involves all of the samples will be considered.
    - the problem of finding the solution to a set of linear inequalities ($\mathbf{a}^t\mathbf{y}_i > 0$) is replaced with one of finding the solution to a set of linear equations ($\mathbf{a}^t\mathbf{y}_i = b_i$).

Minimum Squared-Error procedures

- MSE and the Pseudoinverse

$$\mathbf{Ya} = \mathbf{b}$$

$$\begin{pmatrix} y_{10} & y_{11} & \cdots & y_{1d} \\ y_{20} & y_{21} & \cdots & y_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n0} & y_{n1} & \cdots & y_{nd} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\mathbf{a} = \mathbf{Y}^{-1}\mathbf{b}$$

  - If $\mathbf{Y}$ were nonsingular, a formal solution could be obtained at once:
  - $\mathbf{Y}$ is usually with more rows than columns.
    - more equations than unknowns – *overdetermined*: no exact solution exists.
    - we can seek a weight vector $\mathbf{a}$ that *minimizes* some function of the error between $\mathbf{Ya}$ and $\mathbf{b}$.

Minimum Squared-Error procedures

- MSE and the Pseudoinverse (cont.)
  - error vector $\mathbf{e} = \mathbf{Ya} - \mathbf{b}$
  - sum-of-squared error criterion function $J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2 = \sum_{i=1}^{n} (\mathbf{a}^t \mathbf{y}_i - b_i)^2$
  - it can be solved by a gradient search procedure.

$$\nabla J_s = \sum_{i=1}^{n} 2(\mathbf{a}^t \mathbf{y}_i - b_i)\mathbf{y}_i = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b}) \Rightarrow \boxed{\mathbf{Y}^t\mathbf{Ya} = \mathbf{Y}^t\mathbf{b}}$$

    - now we have converted the problem of solving $\mathbf{Ya}=\mathbf{b}$ to that of solving $\mathbf{Y}^t\mathbf{Ya}=\mathbf{Y}^t\mathbf{b}$.

$$\mathbf{a} = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{b}$$

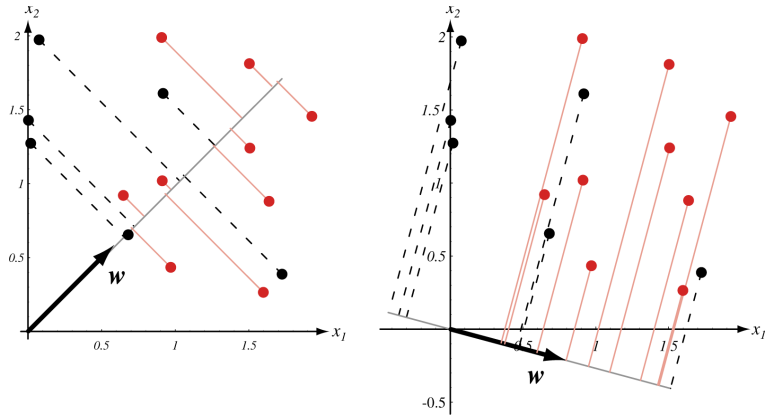
$$= \mathbf{Y}^+\mathbf{b}$$ MSE solution to $\mathbf{Ya}=\mathbf{b}$.

    - *pseudoinverse* of $\mathbf{Y}$: $\mathbf{Y}^+ = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t$
  - by minimizing the squared-error criterion function we might obtain a useful discriminant function in both the separable and the nonseparable cases.

Example 1 in page 241

Relation to Fisher's linear discriminant

- With the proper choice of $\mathbf{b}$, the MSE discriminant function $\mathbf{a}^t\mathbf{y}$ is directly related to Fisher's linear discriminant. (see 3.8.2 at p117)



  - n d-dim samples: $\mathbf{x}_1,\ldots,\mathbf{x}_n$
  - n_1 of these : w_1
  - n_2 of these : w_2
  - $\mathbf{y}_i$: augmented pattern vector of $\mathbf{x}_i$

$$\mathbf{Y}=\begin{bmatrix}\mathbf{1}_1 & \mathbf{X}_1\\ -\mathbf{1}_2 & -\mathbf{X}_2\end{bmatrix} \quad \mathbf{a}=\begin{bmatrix}w_0\\ \mathbf{w}\end{bmatrix} \quad \mathbf{b}=\begin{bmatrix}\frac{n}{n_1}\mathbf{1}_1\\ \frac{n}{n_2}\mathbf{1}_2\end{bmatrix}$$

- we shall now show that this special choice for $\mathbf{b}$ links the MSE solution to Fisher's linear discriminant.

Relation to Fisher's linear discriminant

- From Eq (45) $\mathbf{Y}^t\mathbf{Y}\mathbf{a} = \mathbf{Y}^t\mathbf{b}$

$$\begin{bmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{bmatrix} \begin{bmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & -\mathbf{X}_2 \end{bmatrix} \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{bmatrix} \begin{bmatrix} \frac{n}{n_1}\mathbf{1}_1 \\ \frac{n}{n_2}\mathbf{1}_2 \end{bmatrix}$$

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{\mathbf{x}\in D_i}\mathbf{x} \qquad i = 1,2$$

$$\mathbf{S}_W = \sum_{i=1}^{2}\sum_{\mathbf{x}\in D_i}(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t$$

$$\begin{bmatrix} n & (n_1\mathbf{m}_1 + n_2\mathbf{m}_2)^t \\ (n_1\mathbf{m}_1 + n_2\mathbf{m}_2) & \mathbf{S}_W + n_1\mathbf{m}_1\mathbf{m}_1^t + n_2\mathbf{m}_2\mathbf{m}_2^t \end{bmatrix} \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 0 \\ n(\mathbf{m}_1 - \mathbf{m}_2) \end{bmatrix}$$

$$w_0 = -\mathbf{m}^t\mathbf{w}$$

$$\left[\frac{1}{n}\mathbf{S}_W + \frac{n_1 n_2}{n^2}(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^t\right]\mathbf{w} = \mathbf{m}_1 - \mathbf{m}_2 \quad \Leftarrow \quad \frac{n_1 n_2}{n^2}(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^t\mathbf{w} = (1-\alpha)(\mathbf{m}_1-\mathbf{m}_2)$$

$$\mathbf{w} = \alpha n\mathbf{S}_W^{-1}(\mathbf{m}_1-\mathbf{m}_2) \qquad \mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_1-\mathbf{m}_2)$$ (106) in Ch. 3

- the maximum ratio of between-class scatter to within-class scatter
- from a d-dimensional problem to a more manageable one-dimensional one.

$$w_1 \quad \text{if } \mathbf{w}^t(\mathbf{x}-\mathbf{m}) > 0$$
$$w_2 \quad \text{otherwise}$$

The Widrow-Hoff or LMS procedure

$$J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2 = \sum_{i=1}^{n} (\mathbf{a}^t \mathbf{y}_i - b_i)^2$$

- Advantage of computing minimization of the function by a *gradient descent procedure* over the pseudoinverse:
  - It avoids the problems that arise when $\mathbf{Y}^t\mathbf{Y}$ is singular
  - It avoids the need for working with large matrices.

$$\mathbf{a} = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{b} = \mathbf{Y}^+\mathbf{b}$$

$$\nabla J_s(\mathbf{a}) = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b})$$

$$\boxed{\begin{array}{l}\mathbf{a}(1) \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{Y}^t(\mathbf{Ya}(k) - \mathbf{b})\end{array}}$$

The descent algorithm always yields a solution regardless of whether or not $\mathbf{Y}^t\mathbf{Y}$ is singular.

$$\mathbf{a}(1)$$

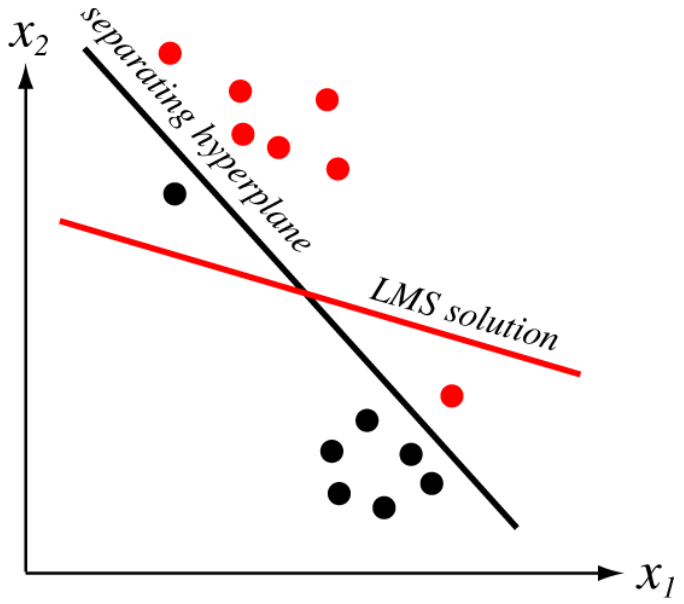
$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)(b(k) - \mathbf{a}^t(k)\mathbf{y}^k)\mathbf{y}^k$$

(Algorithm 10)

Widrow-Hoff or LMS(least-mean-squared) rule:

Storage requirement can be reduced by considering samples sequentially.

The Widrow-Hoff or LMS procedure



The LMS algorithm need not converge to a separating hyperplane, even if one exists. Because the LMS solution minimizes the sum of the squares of the distances of the training points to the hyperplane, for this example the plane is rotated clockwise compared to a separating hyperplane.

Ho-Kashyap procedure

- We shall now see how the MSE procedure can be modified to obtain both a separating vector $\mathbf{a}$ and a margin vector $\mathbf{b}$.
  - The underlying idea comes from the observation that if the samples are separable, and if both $\mathbf{a}$ and $\mathbf{b}$ in the criterion function

$$J_s(\mathbf{a},\mathbf{b}) = \|\mathbf{Ya}-\mathbf{b}\|^2 \qquad (74)$$

  are allowed to vary, then the minimum value of J_s is zero, and the $\mathbf{a}$ that achieves that minimum is a separating vector.
- The Descent Procedure

$$\nabla_a J_s(\mathbf{a},\mathbf{b}) = 2\mathbf{Y}^t(\mathbf{Ya}-\mathbf{b}) \quad \Rightarrow \quad \mathbf{a} = \mathbf{Y}^+\mathbf{b}$$

$$\nabla_b J_s(\mathbf{a},\mathbf{b}) = -2(\mathbf{Ya}-\mathbf{b})$$

We must respect the constraint $\mathbf{b}>0$ and avoid a descent procedure that converges to $\mathbf{b}=0$.

$$\mathbf{b}(k+1) = \mathbf{b}(k) - \eta\frac{1}{2}\left[\nabla_{\mathbf{b}} J_s - \left|\nabla_{\mathbf{b}} J_s\right|\right]$$

Ho-Kashyap procedure

- Ho-Kashyap rule for minimizing $J_s(\mathbf{a},\mathbf{b})$

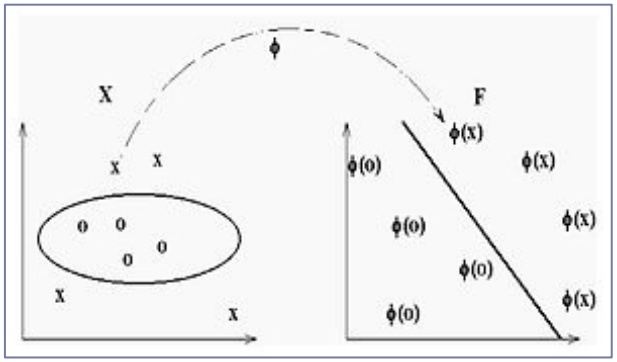
$$\mathbf{b}(1) > 0$$
$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\eta(k)\mathbf{e}^+(k)$$

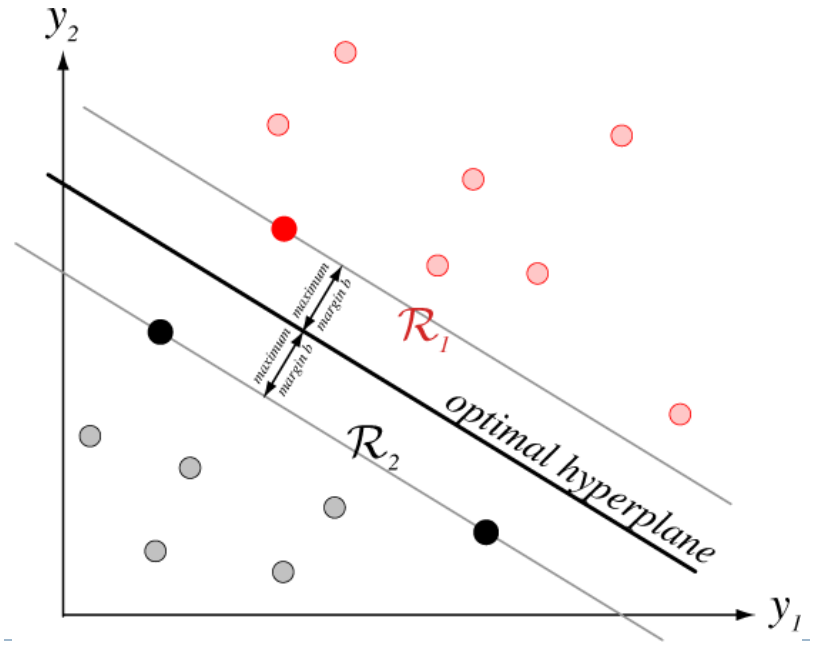
where

$$\mathbf{e}(k) = \mathbf{Y}\mathbf{a}(k) - \mathbf{b}(k) \qquad \mathbf{e}^+(k) = \frac{1}{2}\left(\mathbf{e}(k) + |\mathbf{e}(k)|\right) \qquad \mathbf{a}(k) = \mathbf{Y}^+\mathbf{b}(k)$$

Support vector machines (SVM)

- SVM are motivated by many of the same considerations, which have been mentioned in training linear machines with margins.
  - with an appropriate nonlinear mapping to a sufficiently high dimension, data from two categories can always be separated by a hyperplane.
- Support vectors





Support vector machines (SVM)

- The goal in training a SVM is to find the separating hyperplane with the largest margin:
  - We expect that the larger the margin, the better generalization of the classifier.
- A conceptually simple method of training an SVM is based on a small modification to the familiar Perceptron training rule.
- In practice, finding the worst-case pattern is computationally expensive, since for each update we must search through the entire training set to find the worst-classified pattern.

Support vector machines (SVM)

- An important benefit of the SVM approach is that the complexity of the resulting classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space.
  - As a result, SVMs tend to be less prone to problems of overfitting than some other methods.

Multicategory generalizations

- There is no uniform way to extend all of the two-category procedures to the multicategory case.
  - Linear machine

$$\mathbf{g}_i(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + \omega_{i0}$$

$$\mathbf{g}_i(\mathbf{x}) = \mathbf{a}^t\mathbf{y} \quad \text{by letting } \mathbf{y}(\mathbf{x}) \text{ be a vector of functions of } \mathbf{x}$$

  - The generalization of procedures from a two-category linear classifier to a multicategory linear machine is simplest in the linearly-separable case.

$$\hat{\mathbf{a}}_i^t\mathbf{y}_k > \hat{\mathbf{a}}_j^t\mathbf{y}_k \qquad \text{for all } j \neq i$$

Multicategory generalizations

- Kesler's construction

$$\hat{\mathbf{a}}_i^t \mathbf{y}_k - \hat{\mathbf{a}}_j^t \mathbf{y}_k > 0 \qquad \text{for all } j = 2,\ldots,c$$

$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_c \end{bmatrix} \qquad \boldsymbol{\eta}_{12} = \begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\eta}_{13} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \\ -\mathbf{y} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \cdots \boldsymbol{\eta}_{1c} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ -\mathbf{y} \end{bmatrix}$$

$$\hat{\boldsymbol{\alpha}}^t \boldsymbol{\eta}_{ij} > 0 \quad for \quad j \neq i$$

  - The construction multiplies the dimensionality of the data by c and the number of samples by c-1, which does not make its direct use attractive.
  - Its importance resides in the fact that it allows us to convert many multicategory error-correction procedures to two-category procedures.

Multicategory generalizations

- Convergence of the fixed-increment rule
  - an error-correction rule in which weight changes are made *iff* the present linear machine misclassifies a sample.
  - Let $\mathbf{y}_k$ denote the kth sample requiring correction, and suppose that $\mathbf{y}_k \in Y_i$.
    - there must be at least one $j \neq i$ for which

$$\mathbf{a}_i^t(k)\mathbf{y}^k \leq \mathbf{a}_j^t(k)\mathbf{y}^k$$

$$\mathbf{a}_i(k+1) = \mathbf{a}_i(k) + \mathbf{y}^k$$

$$\mathbf{a}_j(k+1) = \mathbf{a}_j(k) - \mathbf{y}^k$$

$$\mathbf{a}_l(k+1) = \mathbf{a}_l(k), \quad l \neq i \text{ and } l \neq j$$

$$\boldsymbol{\alpha}_k = \begin{bmatrix} \mathbf{a}_1(k) \\ \vdots \\ \vdots \\ \mathbf{a}_c(k) \end{bmatrix} \quad \boldsymbol{\eta}_{ij}^k = \begin{bmatrix} \vdots \\ \mathbf{y}_k \\ \vdots \\ -\mathbf{y}_k \\ \mathbf{0} \end{bmatrix} \begin{matrix} \\ \leftarrow i \\ \\ \leftarrow j \\ \end{matrix}$$

$$\boxed{\boldsymbol{\alpha}_k^t \boldsymbol{\eta}_{ij}^k \leq 0}$$

$$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \boldsymbol{\eta}_{ij}^k$$

    - the use of Kesler's construction to establish equivalences between multicategory and two-category procedures is a powerful tool.

Multicategory generalizations

- Generalization for MSE procedure

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \vdots \\ \mathbf{Y}_c \end{bmatrix}$$

with the samples labeled ω_i comprising the rows of Y_i

$n \times \hat{d}$

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_c \end{bmatrix}$$

$\hat{d} \times c$ weight matrix

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_c \end{bmatrix}$$

$n \times c$

all of the element of B_i are zero except for those in the ith column, which are unity.

$$\mathbf{A} = \mathbf{Y}^{+}\mathbf{B}$$