Linear Discriminant Functions

Linear discriminant functions and decision surfaces

- Definition

  It is a function that is a linear combination of the components of x

  $$g(x) = w^t x + w_0 \qquad (1)$$

  where w is the weight vector and w_0 the bias

- A two-category classifier with a discriminant function of the form (1) uses the following rule:

  Decide ω_1 *if*

  $g(x) > 0$ and ω_2 if $g(x) < 0$

  $\Leftrightarrow$ Decide ω_1 *if*

  $w^t x > -w_0$ and ω_2 otherwise

  If $g(x) = 0 \Rightarrow x$ is assigned to either class

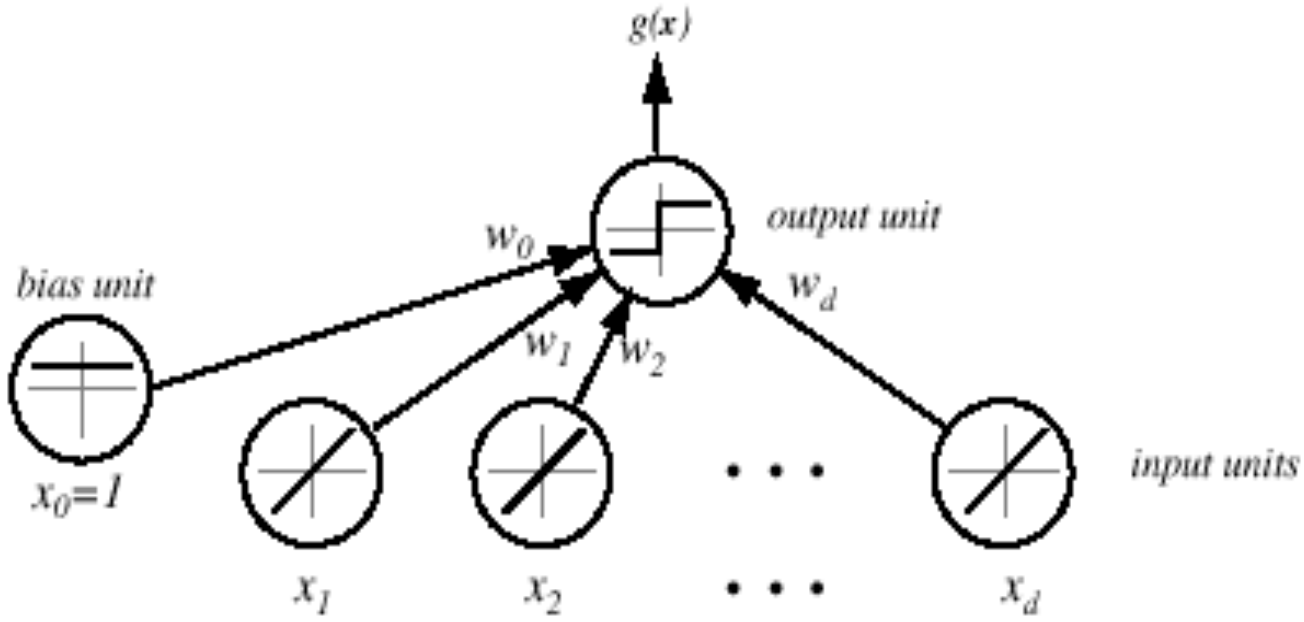


FIGURE 5.1. A simple linear classifier having d input units, each corresponding to the values of the components of an input vector. Each input feature value x_i is multiplied by its corresponding weight w_i; the effective input at the output unit is the sum all these products, $\sum w_i x_i$. We show in each unit its effective input-output function. Thus each of the d input units is linear, emitting exactly the value of its corresponding feature value. The single bias unit unit always emits the constant value 1.0. The single output unit emits a $+1$ if $\mathbf{w}^t\mathbf{x} + w_0 > 0$ or a -1 otherwise. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

– The equation $g(x) = 0$ defines the decision surface that separates points assigned to the category ω_1 from points assigned to the category ω_2

– When $g(x)$ is linear, the decision surface is a hyperplane

– Algebraic measure of the distance from x to the hyperplane (interesting result!)

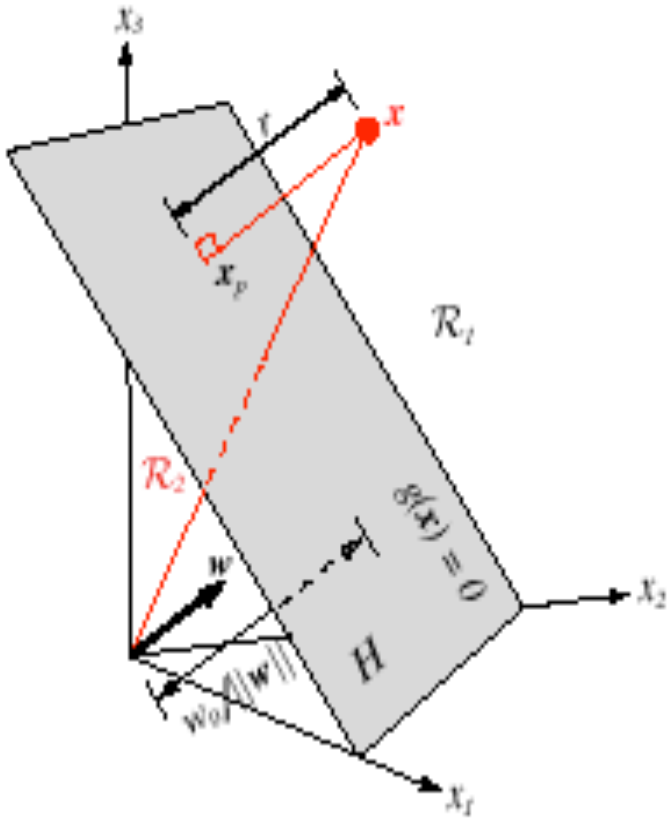


FIGURE 5.2. The linear decision boundary H, where $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x}+w_0 = 0$, separates the feature space into two half-spaces $\mathcal{R}_1$ (where $g(\mathbf{x}) > 0$) and $\mathcal{R}_2$ (where $g(\mathbf{x}) < 0$). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

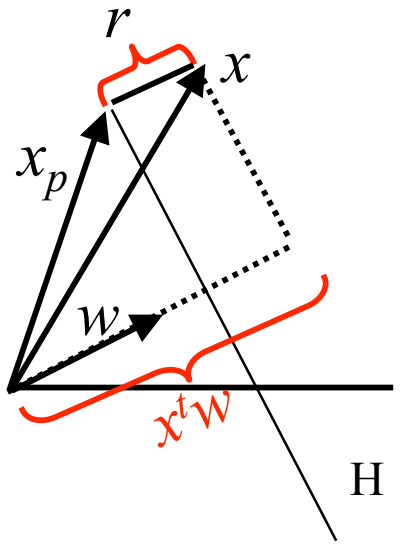
$$\mathbf{x} = \mathbf{x_p} + \frac{r\mathbf{w}}{\|\mathbf{w}\|}$$

$$since\ g(\mathbf{x_p}) = 0 \text{ and } \mathbf{w^t w} = \|\mathbf{w}\|^2$$

$$g(\mathbf{x}) = \mathbf{w^t x} + w_0 \Rightarrow \mathbf{w^t}\left(\mathbf{x_p} + \frac{r\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0$$

$$= g(\mathbf{x_p}) + \mathbf{w^t w}\frac{r}{\|\mathbf{w}\|}$$

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



$$in\ particular\ \mathrm{d}([0,0],\mathrm{H}) = \frac{\mathrm{w}_0}{\|\mathrm{w}\|}$$

- In conclusion, a linear discriminant function divides the feature space by a hyperplane decision surface

- The orientation of the surface is determined by the normal vector w and the location of the surface is determined by the bias

– The multi-category case

- We define c linear discriminant functions

$$g_i(x) = w_i^t x + w_{i0} \qquad \text{i} = 1,\ldots,\text{c}$$

  and assign x to ω_i if $g_i(x) > g_j(x)\ \forall j \neq i$; in case of ties, the classification is undefined
- In this case, the classifier is a "linear machine"
- A linear machine divides the feature space into c decision regions, with $g_i(x)$ being the largest discriminant if x is in the region R_i
- For a two contiguous regions $\mathcal{R}_i$ and $\mathcal{R}_j$; the boundary that separates them is a portion of hyperplane H_{ij} defined by:

$$g_i(x) = g_j(x)$$
$$\Leftrightarrow (w_i - w_j)^t x + (w_{i0} - w_{j0}) = 0$$

- $w_i - w_j$ is normal to H_{ij} and

$$d(x, H_{ij}) = \frac{g_i - g_j}{\|w_i - w_j\|}$$

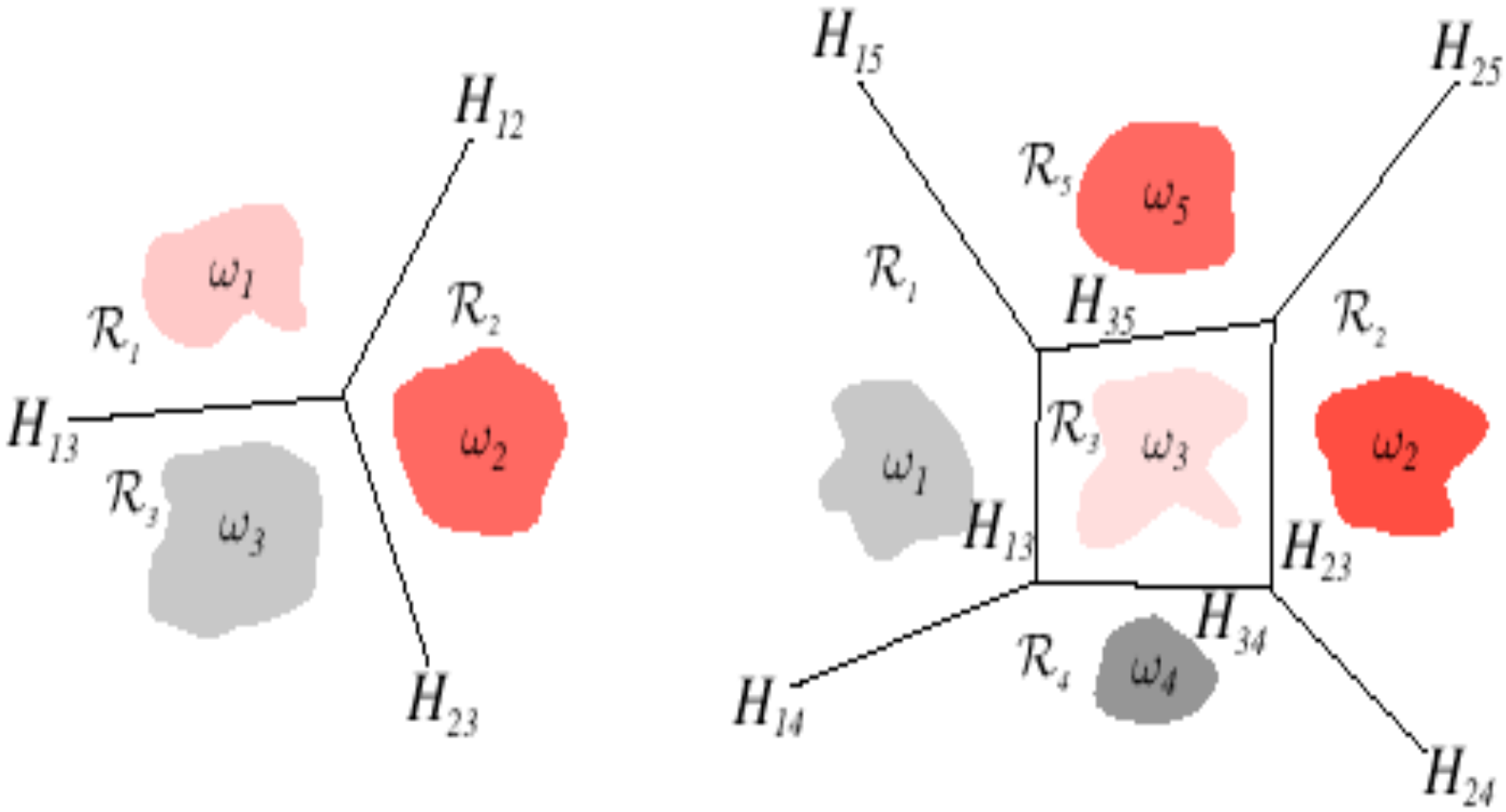


FIGURE 5.4. Decision boundaries produced by a linear machine for a three-class problem and a five-class problem. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Generalized Linear Discriminant Functions

- Decision boundaries which separate between classes may not always be linear
- The complexity of the boundaries may sometimes request the use of highly non-linear surfaces
- A popular approach to generalize the concept of linear decision functions is to consider a generalized decision function as:

$$g(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_N f_N(x) + w_{N+1} \quad (1)$$

where $f_i(x)$, $1 \le i \le N$ are scalar functions of the pattern x, $x \in R^n$ *(Euclidean Space)*

- Introducing $f_{n+1}(x) = 1$ we get:

$$g(x) = \sum_{i=1}^{N+1} w_i f_i(x) = \mathbf{w}^{\mathbf{T}} \mathbf{.y}$$

where $\mathbf{w} = (w_1, w_2, ..., w_N, w_{N+1})^T$

and $\mathbf{y} = (f_1(x), f_2(x), ..., f_N(x), f_{N+1}(x))^T$

- This latter representation of $g(x)$ implies that any decision function defined by equation (1) can be treated as linear in the $(N + 1)$ dimensional space $(N + 1 > n)$

- $g(x)$ maintains its non-linearity characteristics in R^n

- The most commonly used generalized decision function is $g(x)$ for which $f_i(x)$ $(1 \le i \le N)$ are polynomials

$$g(x) = w_0 + \sum_{i=1:N} w_i x_i + \sum_{i=1:N} \sum_{j=0} \alpha_{ij} x_i x_j + \sum_{k=1:N} \sum_{i=1:N} \sum_{j=1:N} \beta_{ijk} x_i x_j x_k + \ldots.$$

- Quadratic decision functions for a 2-dimensional feature space

$$g(x) = w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2^2 + w_4 x_1 + w_5 x_2 + w_6$$
$$here: \ \mathbf{w} = (w_1, w_2, \ldots, w_6)^T \text{ and } \mathbf{y} = (x_1^2, x_1 x_2, x_2^2, x_1, x_2, 1)^T$$

- For patterns $x \in R^n$, the most general quadratic decision function is given by:

$$g(x) = \sum_{i=1}^{n} \omega_{ii} x_i^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \omega_{ij} x_i x_j + \sum_{i=1}^{n} w_i x_i + w_{n+1} \qquad (2)$$

- The number of terms at the right-hand side is:

$$\begin{aligned} l &= N + 1 \\ &= n + \frac{n(n-1)}{2} + n + 1 \\ &= \frac{(n+1)(n+2)}{2} \end{aligned}$$

This is the total number of weights which are the free parameters of the problem

- n = 3 => 10-dimensional
- n = 10 => 65-dimensional

- In the case of polynomial decision functions of order m, a typical $f_i(x)$ is given by:

$$f_i(x) = x_{i_1}^{e_1} x_{i_2}^{e_2} \ldots x_{i_m}^{e_m}$$

where $1 \le i_1, i_2, \ldots, i_m \le n$ and $e_i, 1 \le i \le m$ is 0 or 1.

  - It is a polynomial with a degree between 0 and m. To avoid repetitions, $i_1 \le i_2 \le \ldots \le i_m$

$$g^m(x) = \sum_{i_1=1}^{n} \sum_{i_2=i_1}^{n} \ldots \sum_{i_m=i_{m-1}}^{n} w_{i_1 i_2 \ldots i_m} x_{i_1} x_{i_2} \ldots x_{i_m} + g^{m-1}(x)$$

where $g^m(x)$ is the most general polynomial decision function of order m

Example 1: Let $n = 3$ and $m = 2$ then:

$$g^2(x) = \sum_{i_1=1}^{3} \sum_{i_2=i_1}^{3} w_{i_1 i_2} x_{i_1} x_{i_2} + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4$$

$$= w_{11} x_1^2 + w_{12} x_1 x_2 + w_{13} x_1 x_3 + w_{22} x_2^2 + w_{23} x_2 x_3 + w_{33} x_3^2$$

$$+ w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4$$

Example 2: Let $n = 2$ and $m = 3$ then:

$$g^3(x) = \sum_{i_1=1}^{2} \sum_{i_2=i_1}^{2} \sum_{i_3=i_2}^{2} w_{i_1 i_2 i_3} x_{i_1} x_{i_2} x_{i_3} + g^2(x)$$

$$= w_{111} x_1^3 + w_{112} x_1^2 x_2 + w_{122} x_1 x_2^2 + w_{222} x_2^3 + g^2(x)$$

$$\text{where } g^2(x) = \sum_{i_1=1}^{2} \sum_{i_2=i_1}^{2} w_{i_1 i_2} x_{i_1} x_{i_2} + g^1(x)$$

$$= w_{11} x_1^2 + w_{12} x_1 x_2 + w_{22} x_2^2 + w_1 x_1 + w_2 x_2 + w_3$$

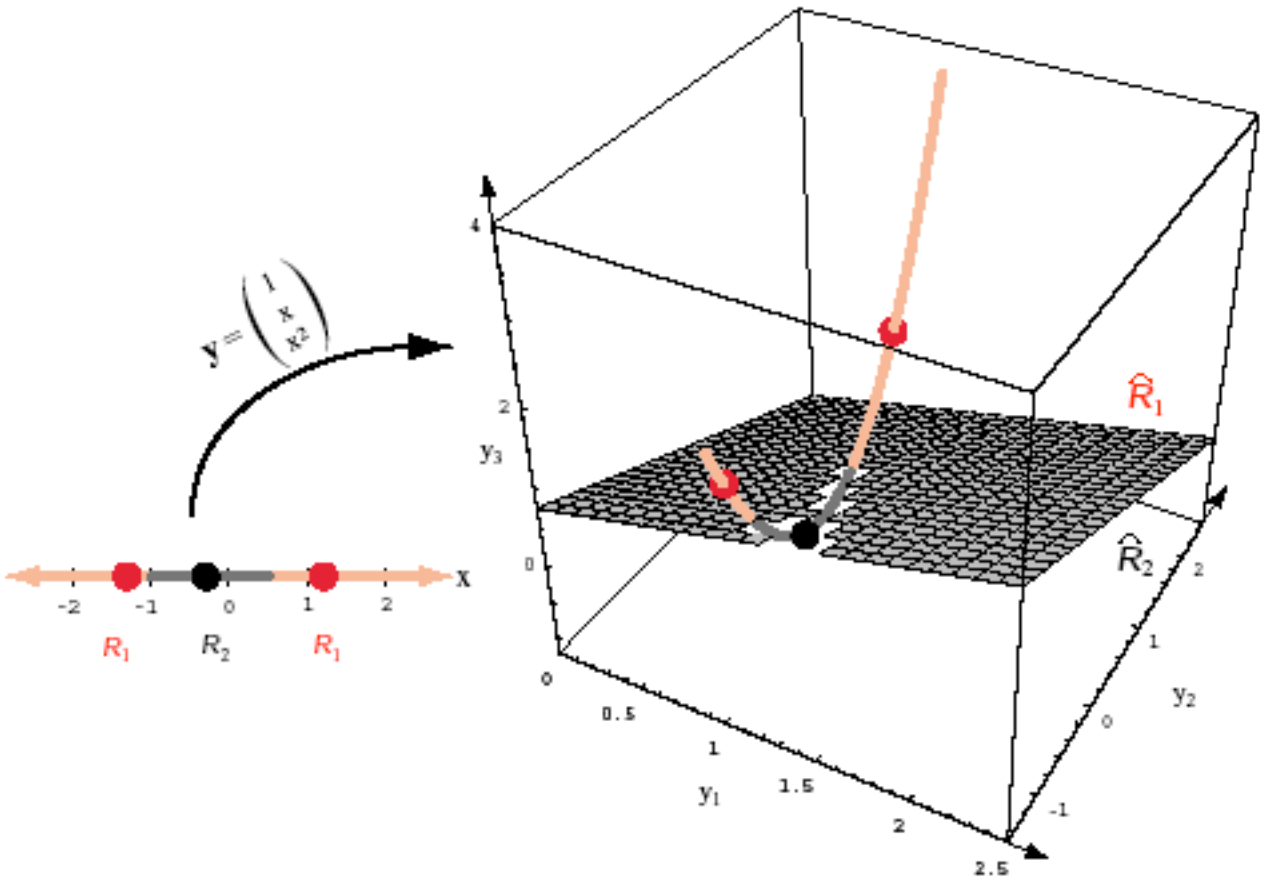


Figure 5.5: The mapping $\mathbf{y} = (1, x, x^2)^t$ takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting $\mathbf{y}$ space into regions corresponding to two categories, and this in turn gives a non-simply connected decision region in the one-dimensional x space.

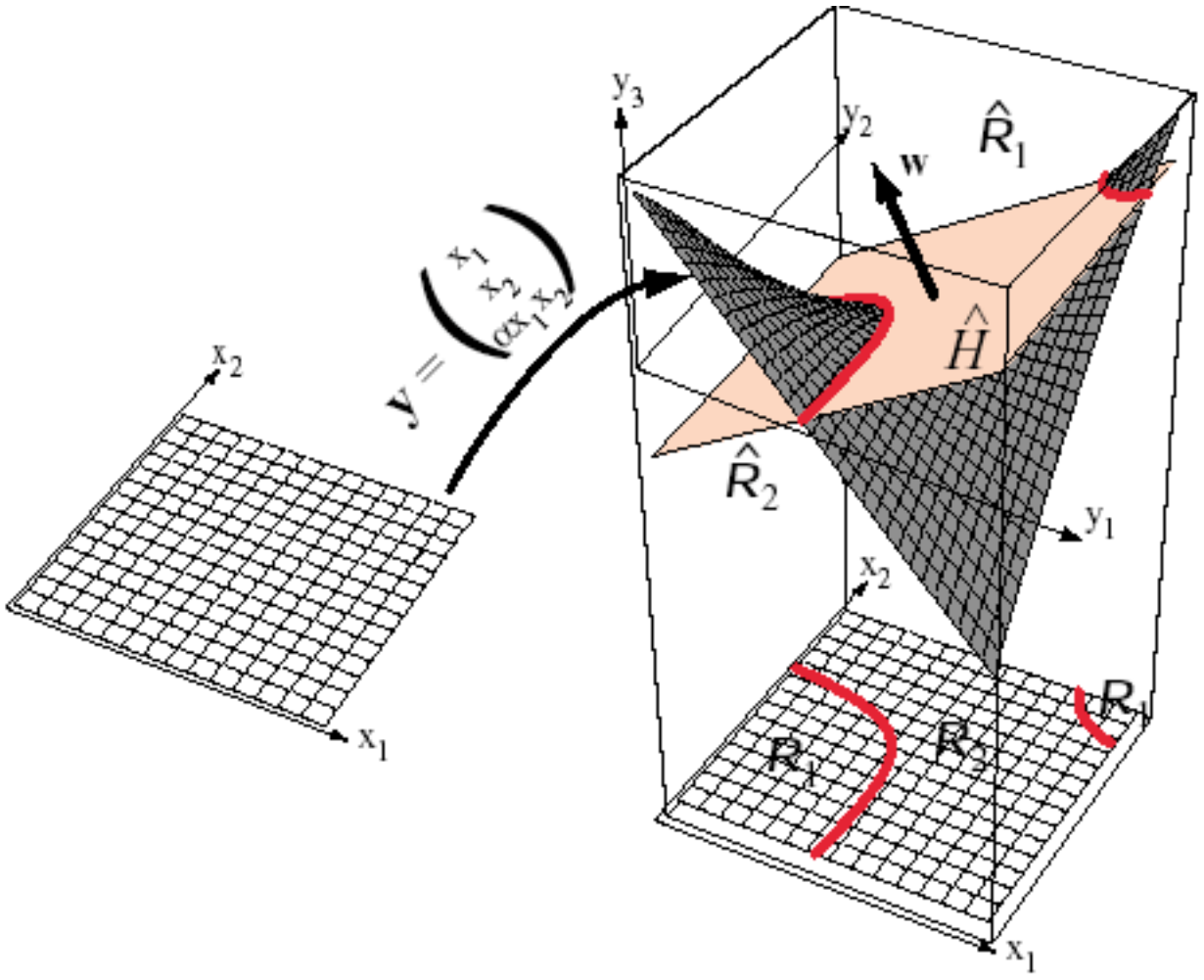


Figure 5.6: The two-dimensional input space $\mathbf{x}$ is mapped through a polynomial function f to $\mathbf{y}$. Here the mapping is $y_1 = x_1$, $y_2 = x_2$ and $y_3 \propto x_1 x_2$. A linear discriminant in this transformed space is a hyperplane, which cuts the surface. Points to the positive side of the hyperplane $\hat{H}$ correspond to category ω_1, and those beneath it ω_2. Here, in terms of the $\mathbf{x}$ space, $\mathcal{R}_1$ is a not simply connected.

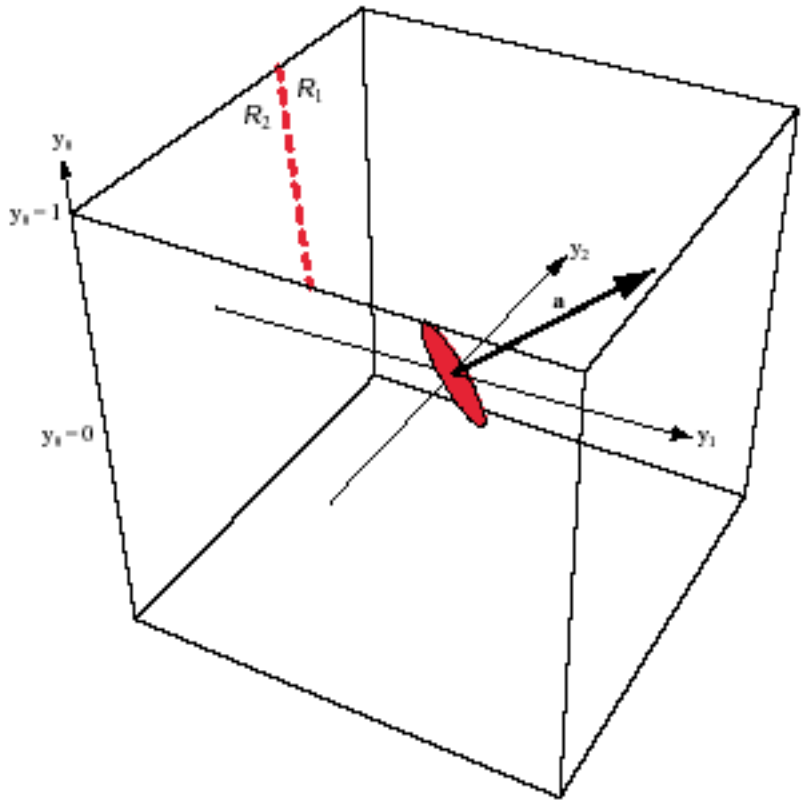


Figure 5.7: A three-dimensional augmented feature space $\mathbf{y}$ and augmented weight vector $\mathbf{a}$ (at the origin). The set of points for which $\mathbf{a}^t\mathbf{y} = 0$ is a plane (or more generally, a hyperplane) perpendicular to $\mathbf{a}$ and passing through the origin of $\mathbf{y}$-space, as indicated by the red disk. Such a plane need not pass through the origin of the two-dimensional $\mathbf{x}$-space at the top, of course, as shown by the dashed line. Thus there exists an augmented weight vector $\mathbf{a}$ that will lead to any straight decision line in $\mathbf{x}$-space.

- Augmentation
- Incorporate labels into data
- Margin

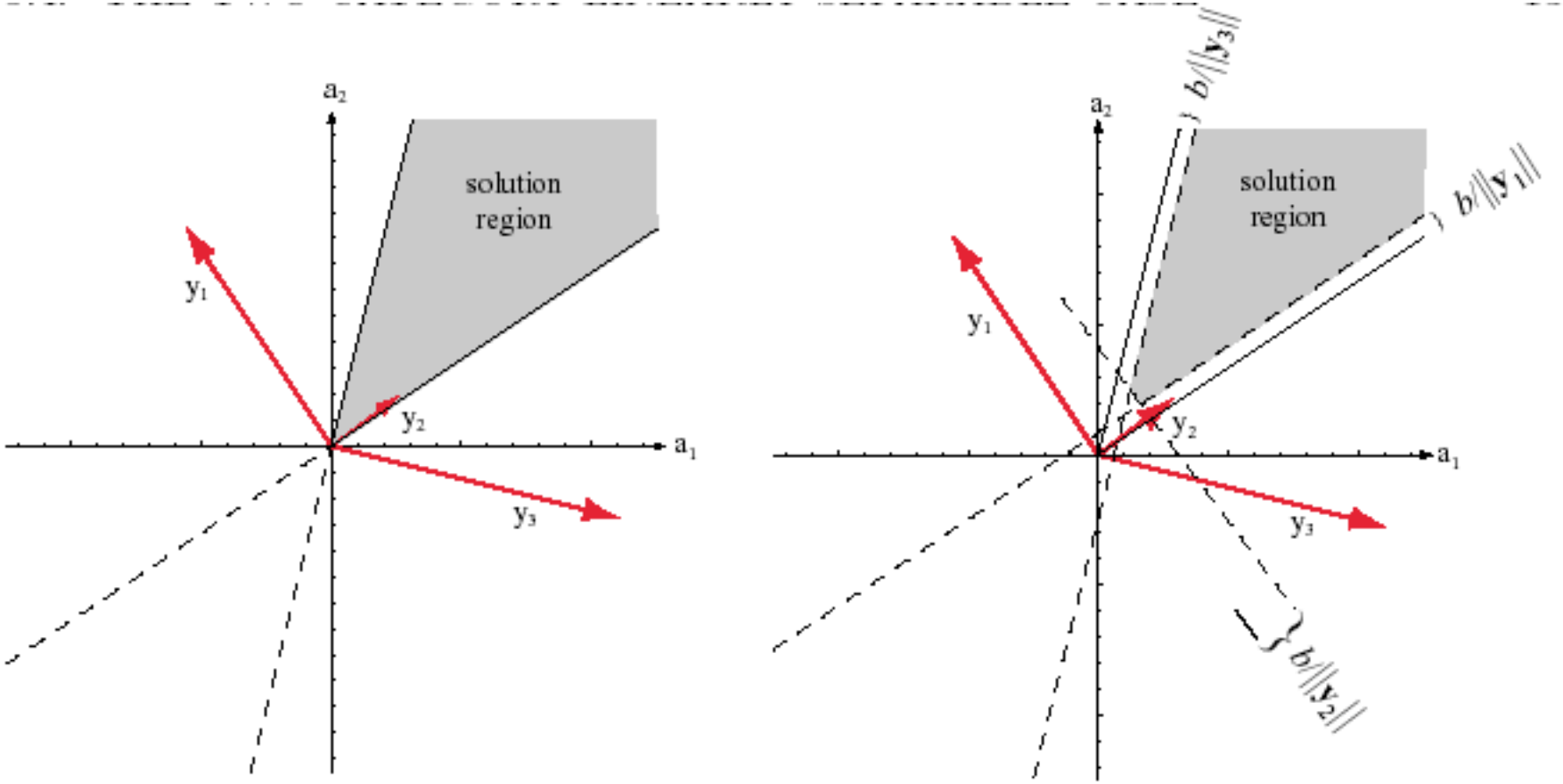


Figure 5.9: The effect of the margin on the solution region. At the left, the case of no margin ($b = 0$) equivalent to a case such as shown at the left in Fig. 5.8. At the right is the case $b > 0$, shrinking the solution region by margins $b/\|\mathbf{y}_i\|$.

Learning Linear Classifiers

Basic Ideas

Directly "fit" linear boundary in feature space.

- 1) Define an error function for classification
  - Number of errors?
  - Total distance of mislabeled points from boundary?
  - Least squares error
  - Within/between class scatter
- Optimize the error function on training data
  - Gradient descent
  - Modified gradient descent
  - Various search proedures.
  - How much data is needed?

Two-category case

- Given $x_1, x_2, \ldots, x_n$ sample points, with true category labels: $\alpha_1, \alpha_2, \ldots, \alpha_n$

$$\left.\begin{array}{l}\alpha_i = 1 \\ \alpha_i = -1\end{array}\right\} \text{if point } x_i \text{ is from class } \omega_1 \text{ or } \omega_2$$

- Decision are made according to:

$$if\ \mathbf{a}^t x_i > 0 \quad \text{class } \omega_1 \text{ is chosen}$$

$$if\ \mathbf{a}^t x_i < 0 \quad \text{class } \omega_2 \text{ is chosen}$$

- Now these decisions are wrong when $\mathbf{a^t x_i}$ is negative and belongs to class ω_1.

  Let $y_i = \alpha_i x_i$ Then $y_i > 0$ when correctly labelled, negative otherwise.

- Separating vector (Solution vector)
  - Find **a** such that $\mathbf{a}^t\mathbf{y}_i>0$ for all i.
- $\mathbf{a}^t\mathbf{y}_i=0$ defines a hyperplane with **a** as normal

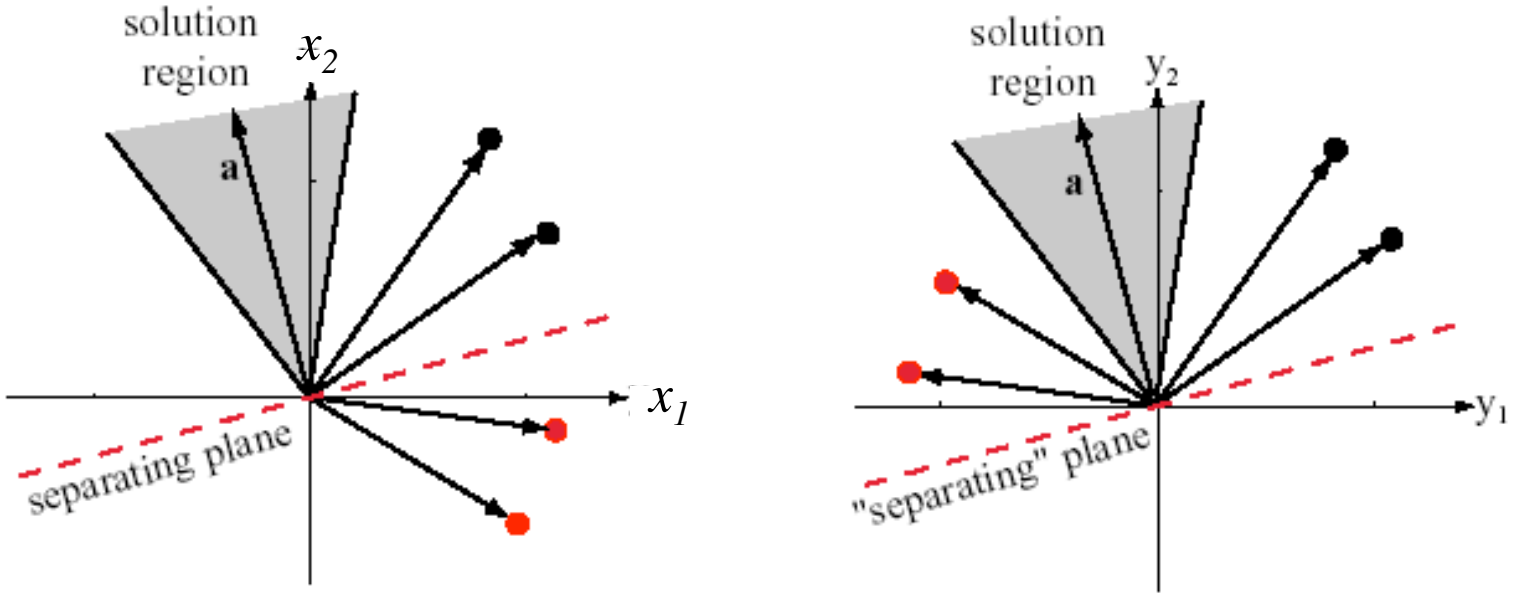


Figure 5.8: Four training samples (black for ω_1, red for ω_2) and the solution region in feature space. The figure on the left shows the raw data; the solution vectors leads to a plane that separates the patterns from the two categories. In the figure on the right, the red points have been "normalized" — i.e., changed in sign. Now the solution vector leads to a plane that places all "normalized" points on the same side.

- Problem: solution not unique
- Add Additional constraints:

  Margin, the distance away from boundary $\mathbf{a}^t\mathbf{y}_i > b$

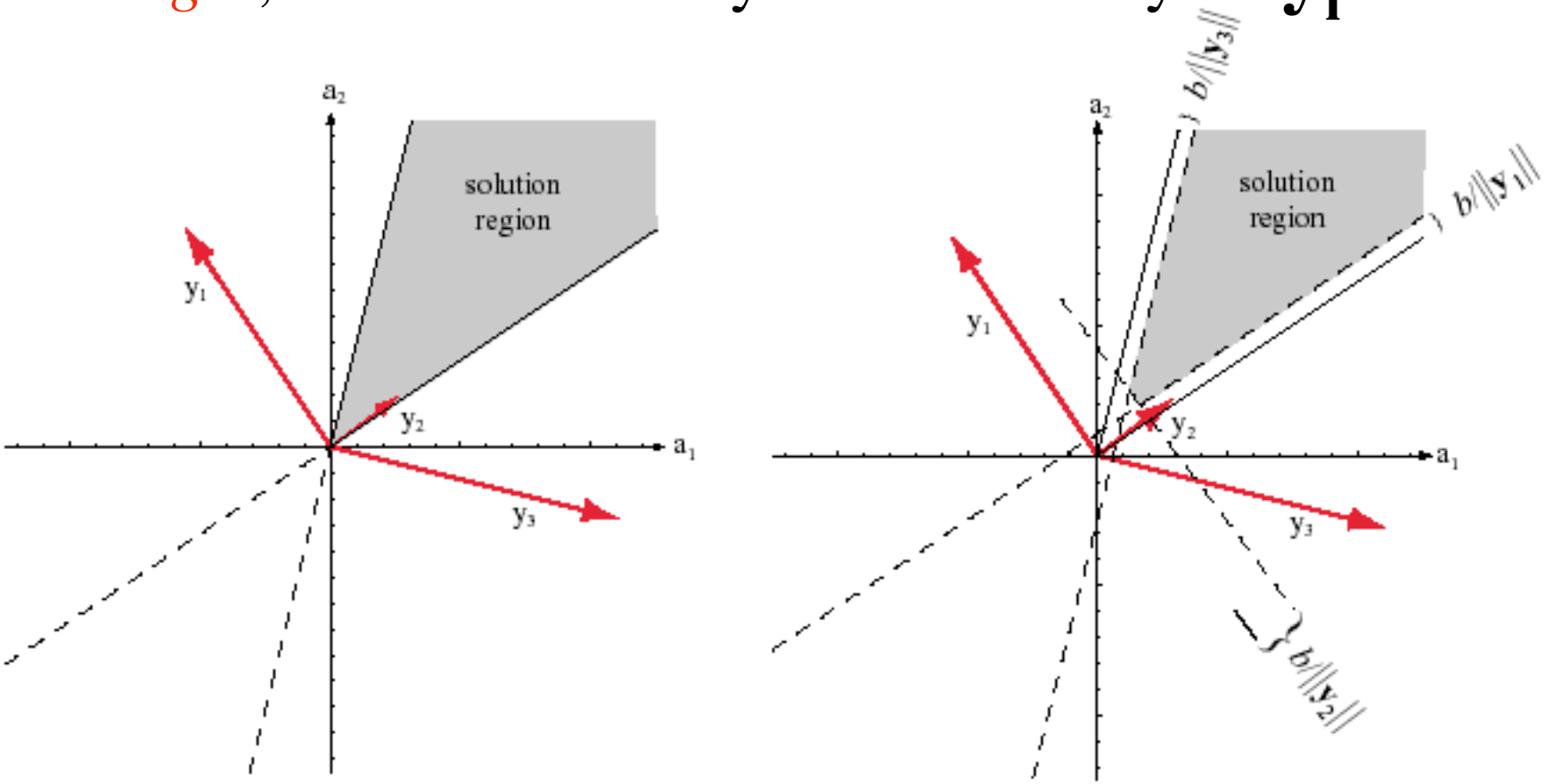
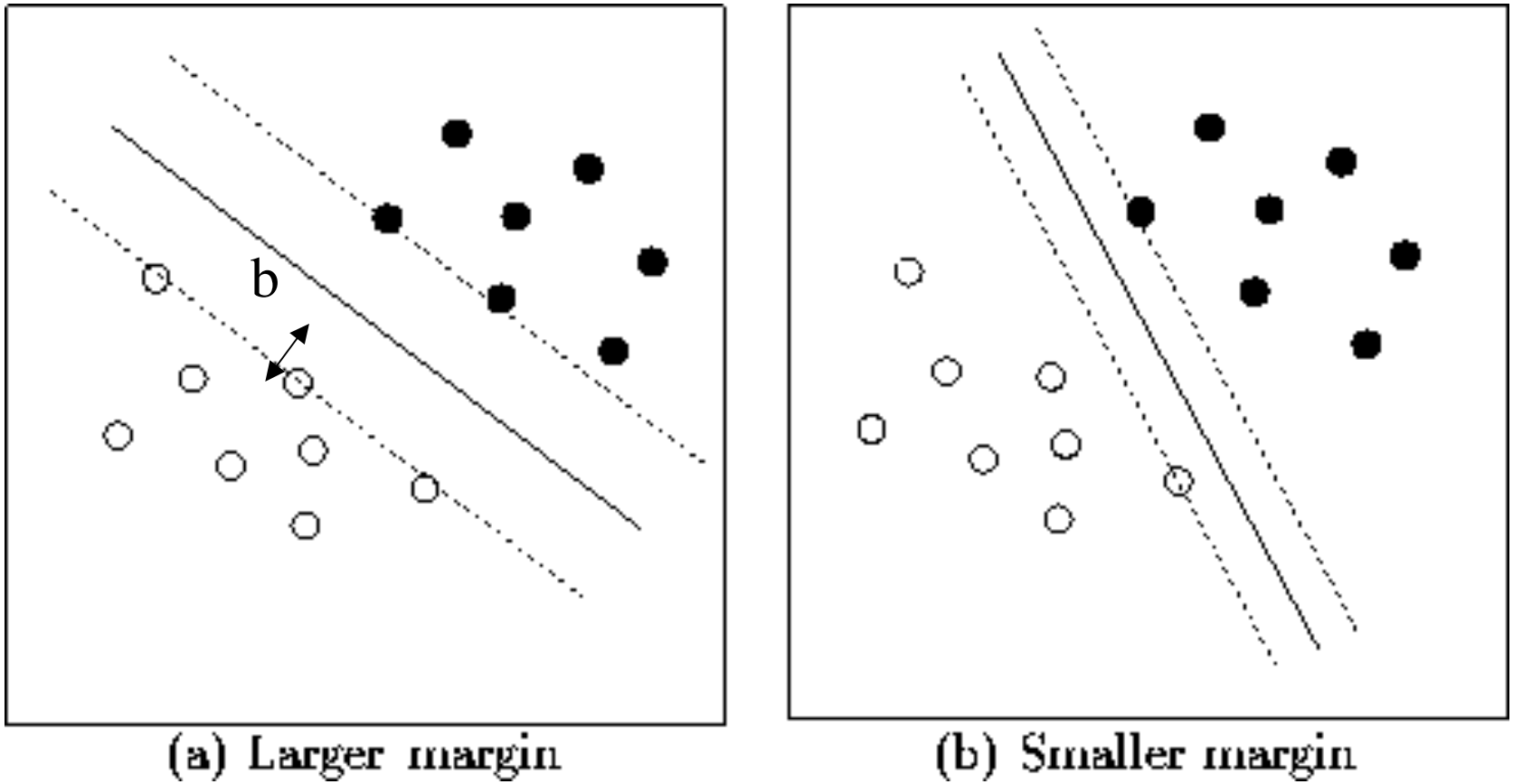


Figure 5.9: The effect of the margin on the solution region. At the left, the case of no margin ($b = 0$) equivalent to a case such as shown at the left in Fig. 5.8. At the right is the case $b > 0$, shrinking the solution region by margins $b/\|\mathbf{y}_i\|$.

Margins in data space



(a) Larger margin

(b) Smaller margin

Larger margins promote uniqueness for underconstrained problems

Finding a solution vector by minimizing an error criterion

What error function?

How do we weight the points? All the points or only error points?

Only error points:

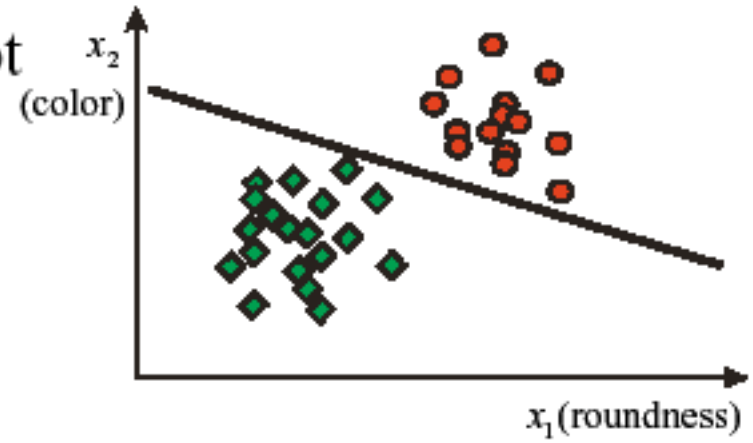
Perceptron Criterion

$$J_P(\mathbf{a}^t) = \sum_{y_i \in \mathcal{Y}} -(\mathbf{a}^t y_i) \qquad \mathcal{Y} = \{y_i \mid \mathbf{a}^t y_i < 0\}$$

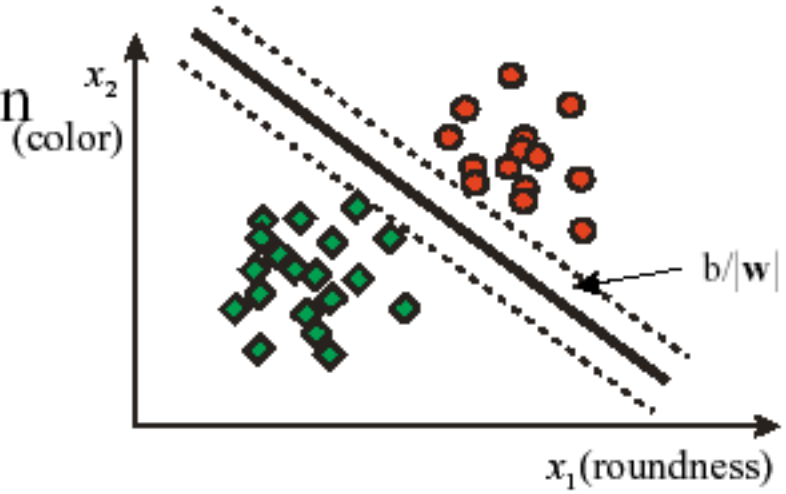
Perceptron Criterion with Margin

$$J_P(\mathbf{a}^t) = \sum_{y_i \in \mathcal{Y}} -(\mathbf{a}^t y_i) \qquad \mathcal{Y} = \{y_i \mid \mathbf{a}^t y_i < b\}$$

- Generalization is not good in this case:



- But better if a margin is introduced:



Minimizing Error via Gradient Descent

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

Algorithm 1 (Basic gradient descent)

1 **begin initialize** $\mathbf{a}$, criterion θ, $\eta(\cdot)$, $k = 0$
2 **do** $k \leftarrow k + 1$
3 $\mathbf{a} \leftarrow \mathbf{a} - \eta(k)\boldsymbol{\nabla} J(\mathbf{a})$
4 **until** $\eta(k)\boldsymbol{\nabla} J(\mathbf{a}) < \theta$
5 **return** $\mathbf{a}$
6 **end**

- The min(max) problem:

$$\min_{x} f(x)$$

- But we learned in calculus how to solve that kind of question!

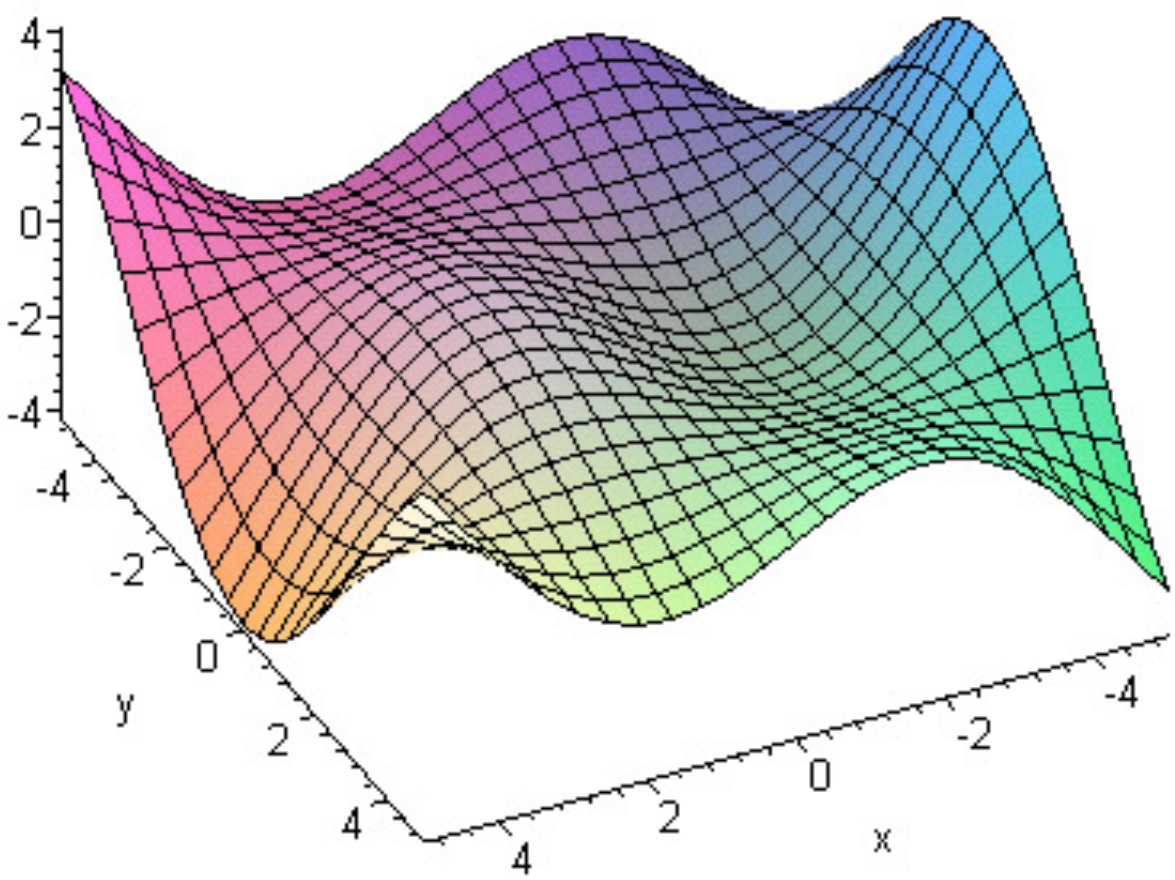
Motivation

- Not exactly,
- Functions: $f : R^n \rightarrow R$
- High order polynomials:

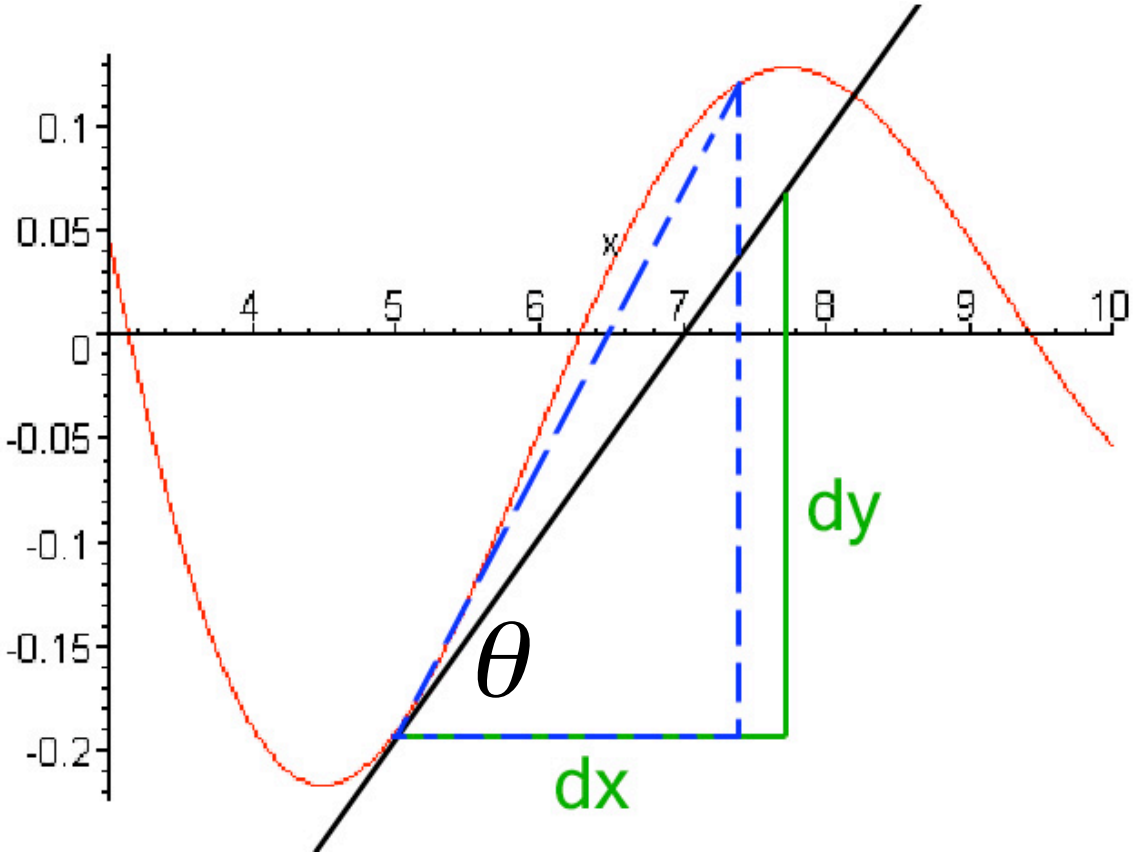
$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7$$

- What about function that don't have an analytic presentation: "Black Box"

$$f := (x, y) \rightarrow \cos\left(\frac{1}{2}x\right)\cos\left(\frac{1}{2}y\right)x$$



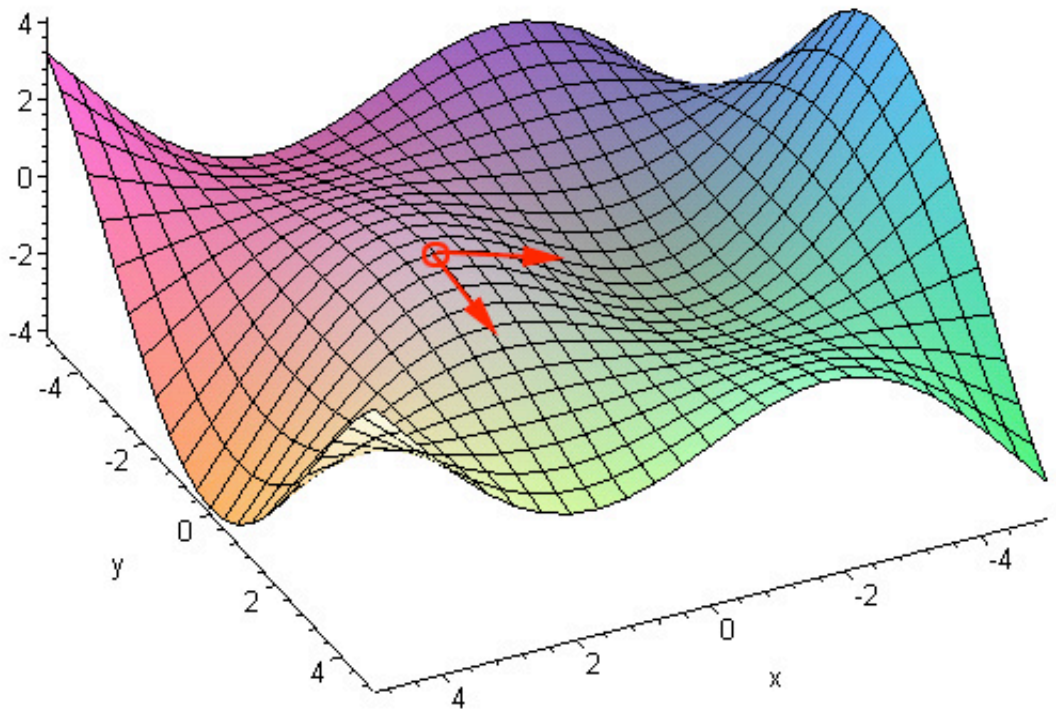
Directional Derivatives:
first, the one dimension derivative:



Directional Derivatives : Along the Axes…

$$\frac{\partial f(x,y)}{\partial y}$$

$$\frac{\partial f(x,y)}{\partial x}$$

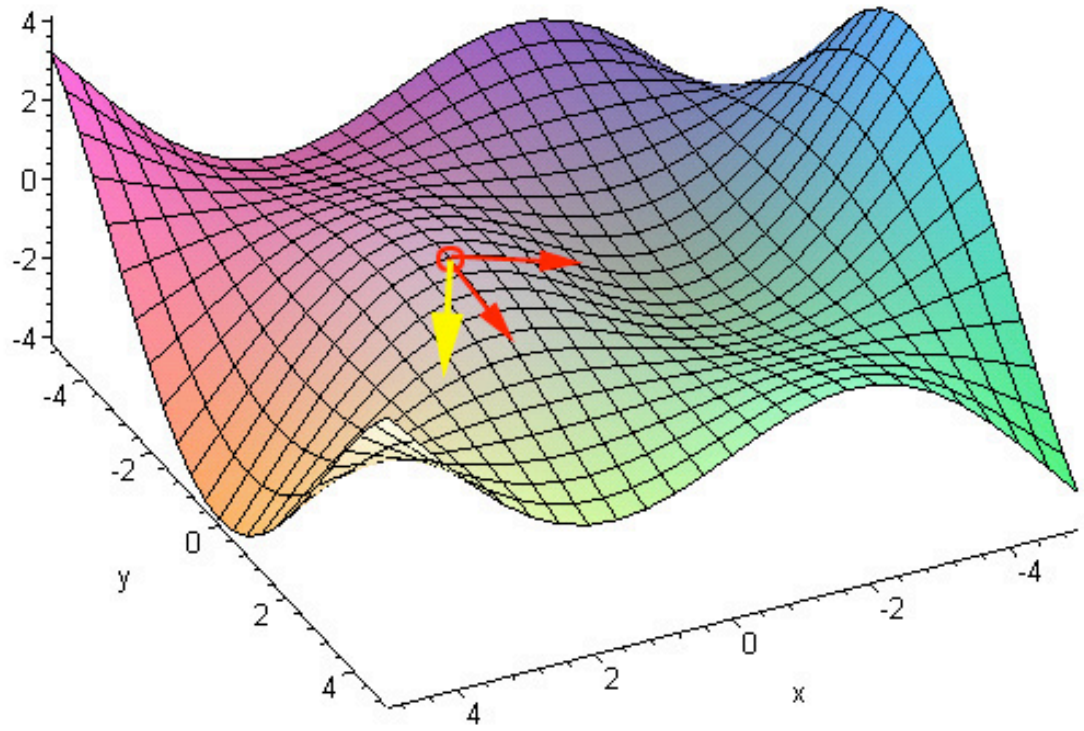


Directional Derivatives : In general direction…

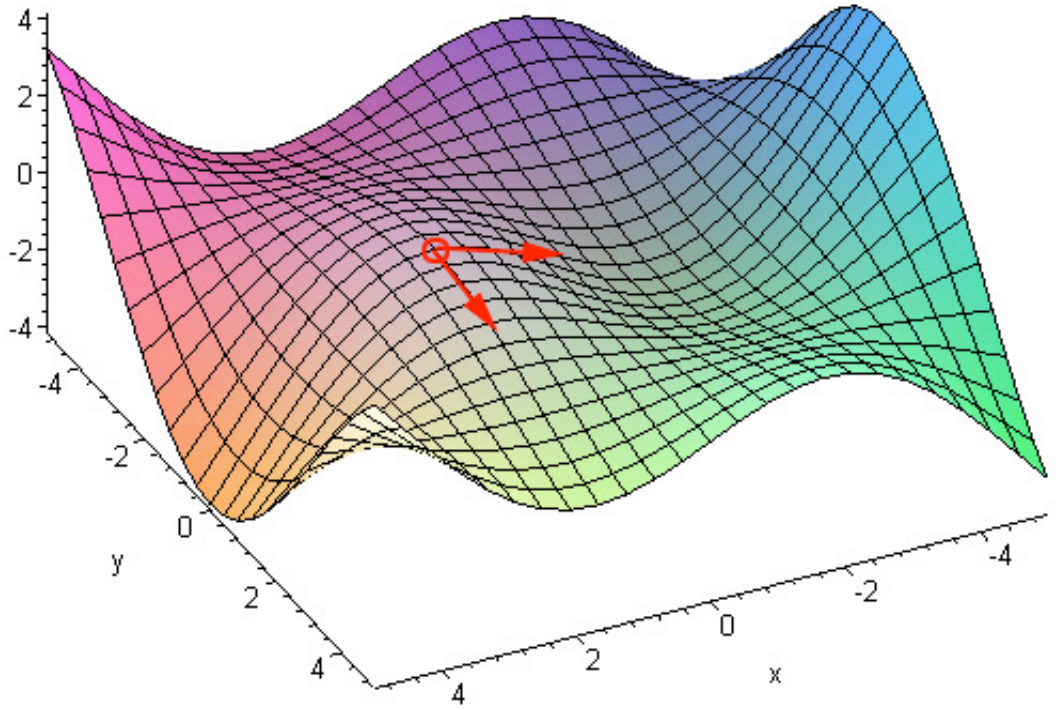
$$v \in R^2$$

$$\|v\| = 1$$

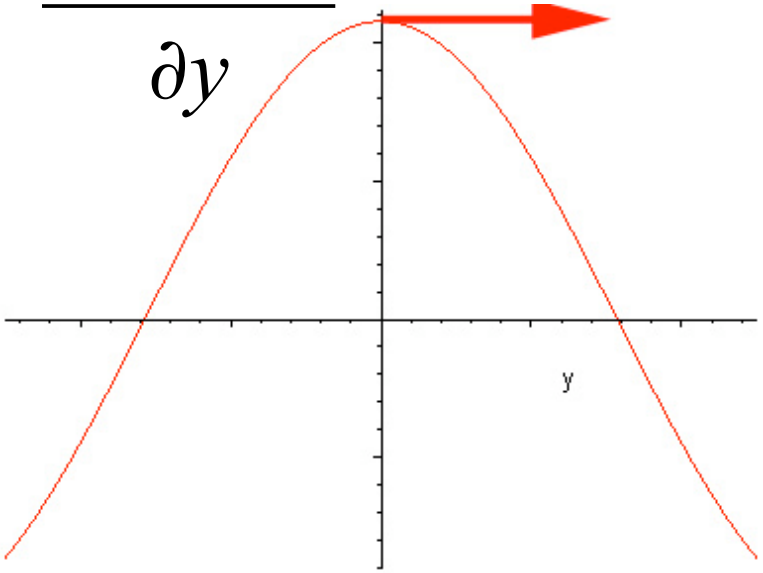
$$\frac{\partial f(x,y)}{\partial v}$$



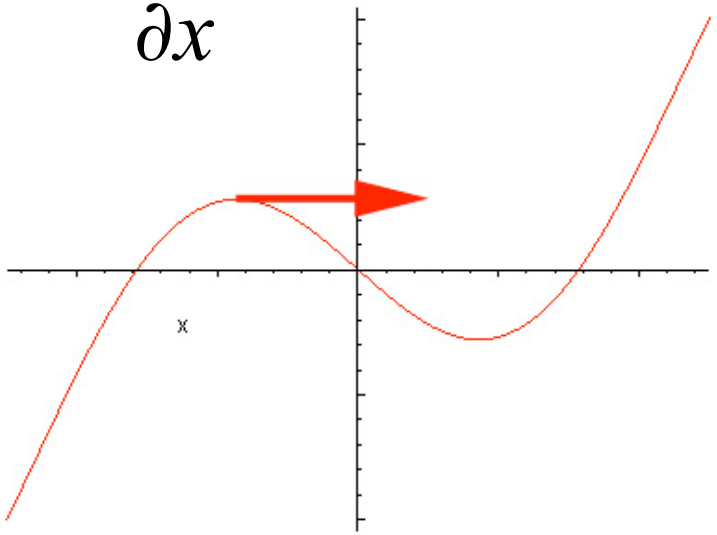
Directional Derivatives



$$\frac{\partial f(x,y)}{\partial y}$$



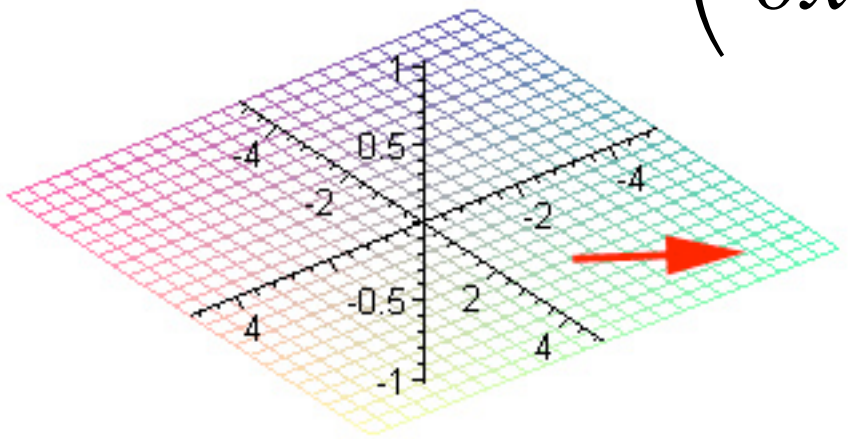
$$\frac{\partial f(x,y)}{\partial x}$$



The Gradient: Definition in R^2

$$f : R^2 \to R$$

$$\nabla f(x, y) := \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{pmatrix}$$



In the plane

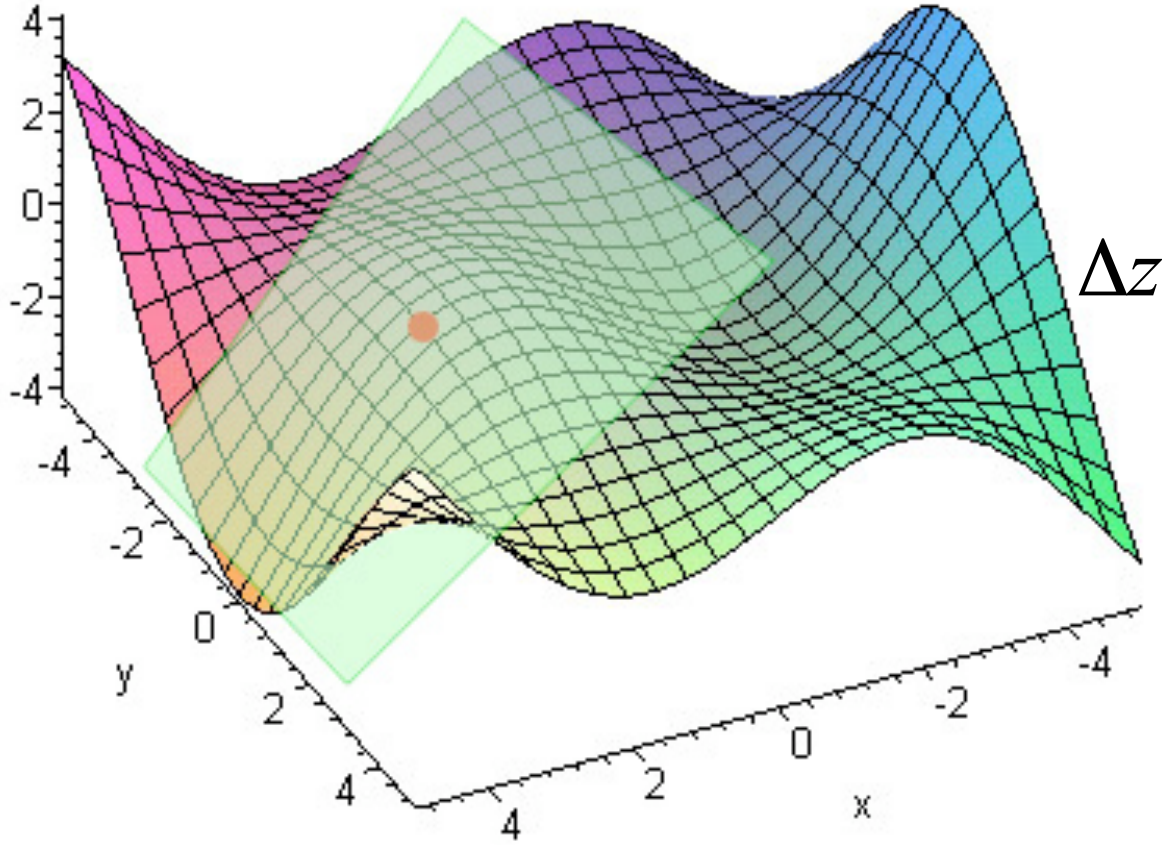
The Gradient: Definition

$$f : R^n \rightarrow R$$

$$\nabla f(x_1,\ldots,x_n) := \left(\frac{\partial f}{\partial x_1},\ldots, \quad \frac{\partial f}{\partial x_n} \right)$$

The Gradient Properties

- The gradient defines (hyper) plane approximating the function infinitesimally



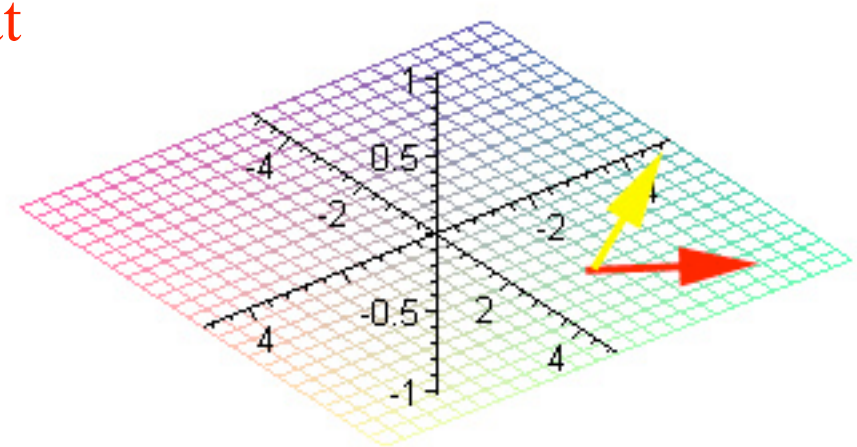
$$\Delta z = \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y$$

The Gradient properties

- Computing directional derivatives
- By the chain rule: (important for later use)

$$\|v\| = 1 \qquad \frac{\partial f}{\partial v}(p) = \left\langle (\nabla f)_p, v \right\rangle$$

Want rate of change along v, at a point p



The Gradient properties

$\frac{\partial f}{\partial v}$ **is maximal choosing** $v = \frac{1}{\left\|(\nabla f)_p\right\|} \cdot (\nabla f)_p$

is minimal choosing $v = \frac{-1}{\left\|(\nabla f)_p\right\|} \cdot (\nabla f)_p$

(intuition: the gradient points to the greatest change in direction)

The Gradient Properties

- **L**et $f : R^n \longrightarrow R$ be a smooth function around P,

  if f has **local minimum** (maximum) at p

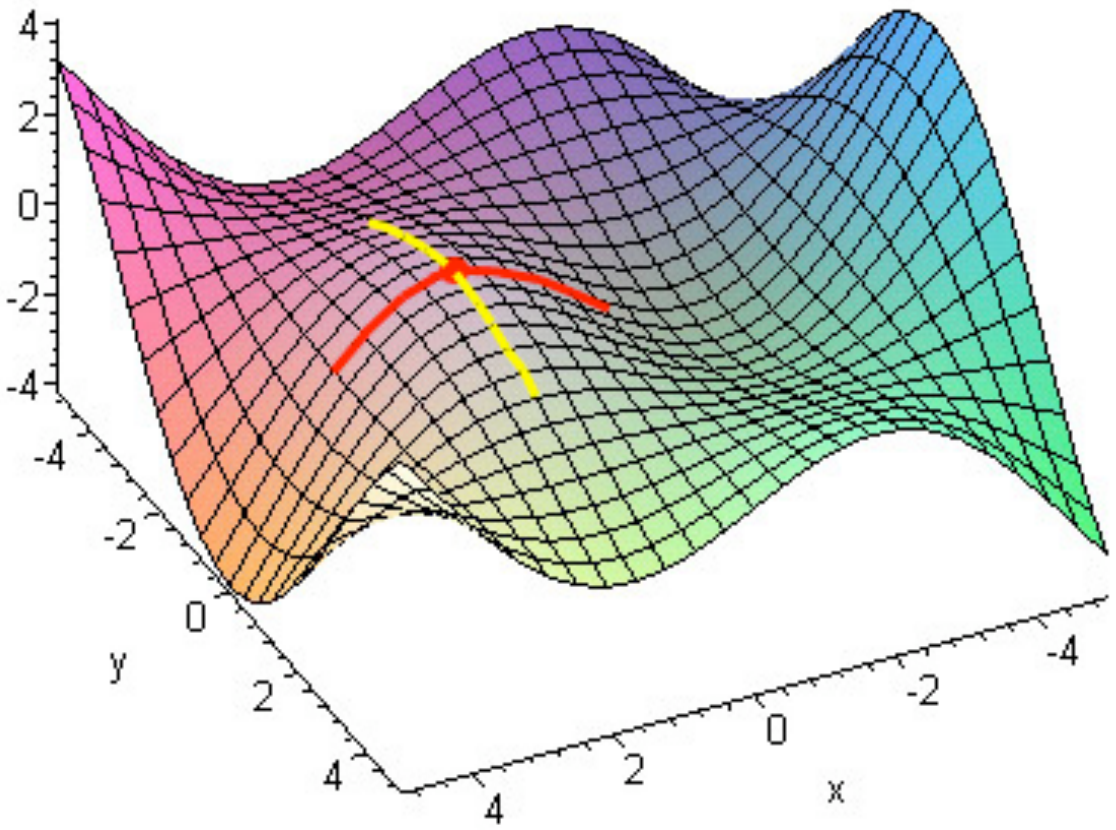
  then,

$$(\nabla f)_p = \vec{0}$$

(Necessary for local min(max))

The Gradient Properties

Intuition



The Gradient Properties

Formally: for any $v \in R^n \setminus \{0\}$

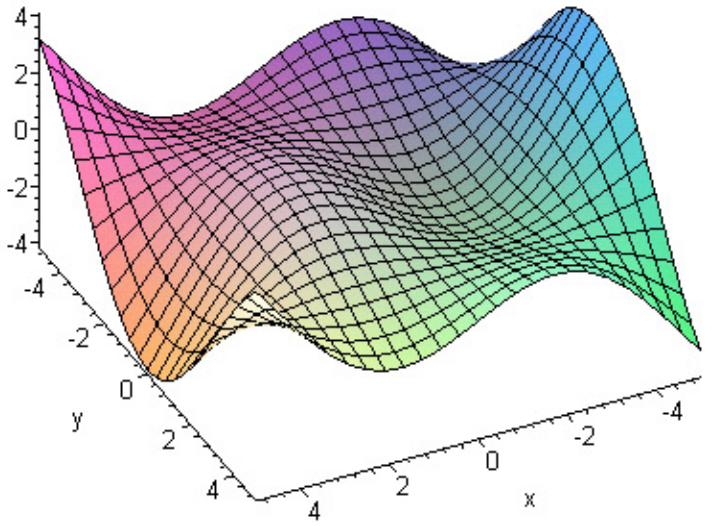
We get:

$$0 = \frac{df(p + t \cdot v)}{dt} = \left\langle (\nabla f)_p, v \right\rangle$$

$$\Rightarrow (\nabla f)_p = \vec{0}$$

The Gradient Properties

- We found the best **INFINITESIMAL DIRECTION** at each point**,**
- Looking for minimum using “blind man” procedure
- How can get to the minimum using this knowledge?



Steepest Descent

- Algorithm

Initialization: $x_0 \in R^n$

loop while $\nabla f(x_i) > \varepsilon$

compute ***search direction*** $h_i = \nabla f(x_i)$

update $x_{i+1} = x_i - \lambda \cdot h_i$

end loop

Setting the Learning Rate Intelligently

$$J(\mathbf{a}) \simeq J(\mathbf{a}(k)) + \nabla J^t(\mathbf{a} - \mathbf{a}(k)) + \frac{1}{2}(\mathbf{a} - \mathbf{a}(k))^t \mathbf{H}\ (\mathbf{a} - \mathbf{a}(k)),$$

where $\mathbf{H}$ is the *Hessian matrix* of second partial derivatives $\partial^2 J/\partial a_i \partial a_j$ evaluated at $\mathbf{a}(k)$. Then, substituting $\mathbf{a}(k+1)$ from Eq. 12 into Eq. 13 we find:

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

Minimizing this approximation:

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

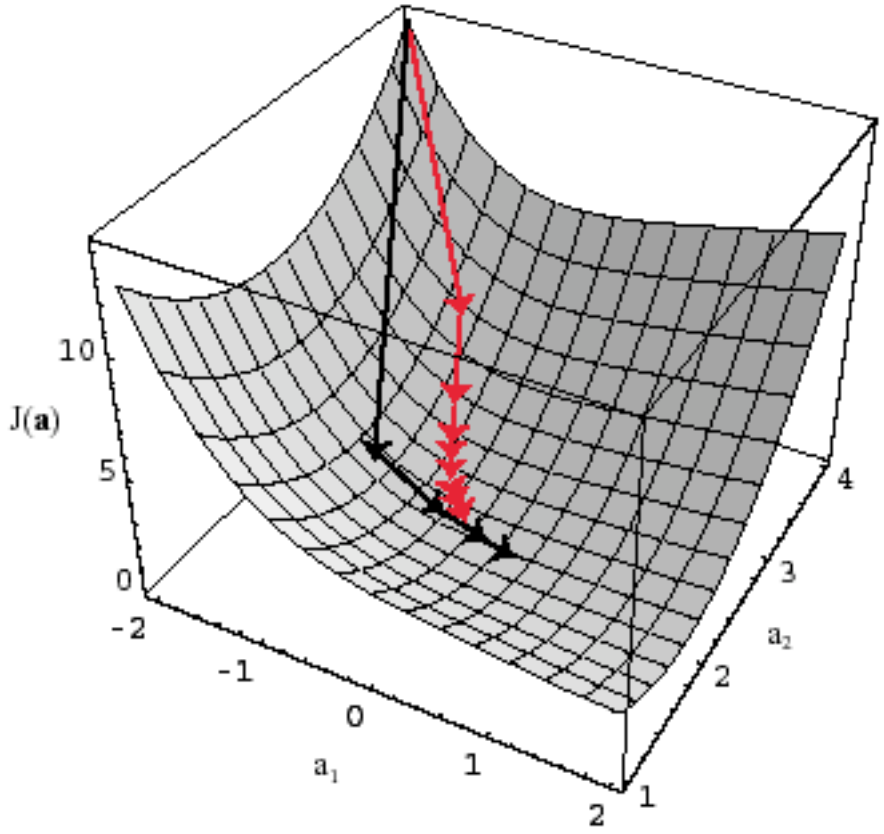
Minimizing Perceptron Criterion

Perceptron Criterion

$$J_P(\mathbf{a}^t) = \sum_{y_i \in \mathcal{Y}} -\mathbf{a}^t y_i \qquad \mathcal{Y} = \{y_i \mid \mathbf{a}^t y_i < 0\}$$

$$\nabla_{\mathbf{a}}\left(J_P(\mathbf{a}^t)\right) = \nabla_{\mathbf{a}} \sum_{y_i \in \mathcal{Y}} -\mathbf{a}^t y_i$$

$$= \sum_{y_i \in \mathcal{Y}} -y_i$$



Which gives the update rule :

$$\mathbf{a}(j+1) = \mathbf{a}(k) + \eta_k \sum_{y_i \in \mathcal{Y}_k} y_i$$

Where $\mathcal{Y}_k$ is the misclassified set at step k

Batch Perceptron Update

Algorithm 3 (Batch Perceptron)

1 **begin initialize** $\mathbf{a}, \eta(\cdot), \text{criterion } \theta, k = 0$
2 **do** $k \leftarrow k + 1$
3 $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y}$
4 **until** $\eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y} < \theta$
5 **return** $\mathbf{a}$
6 **end**

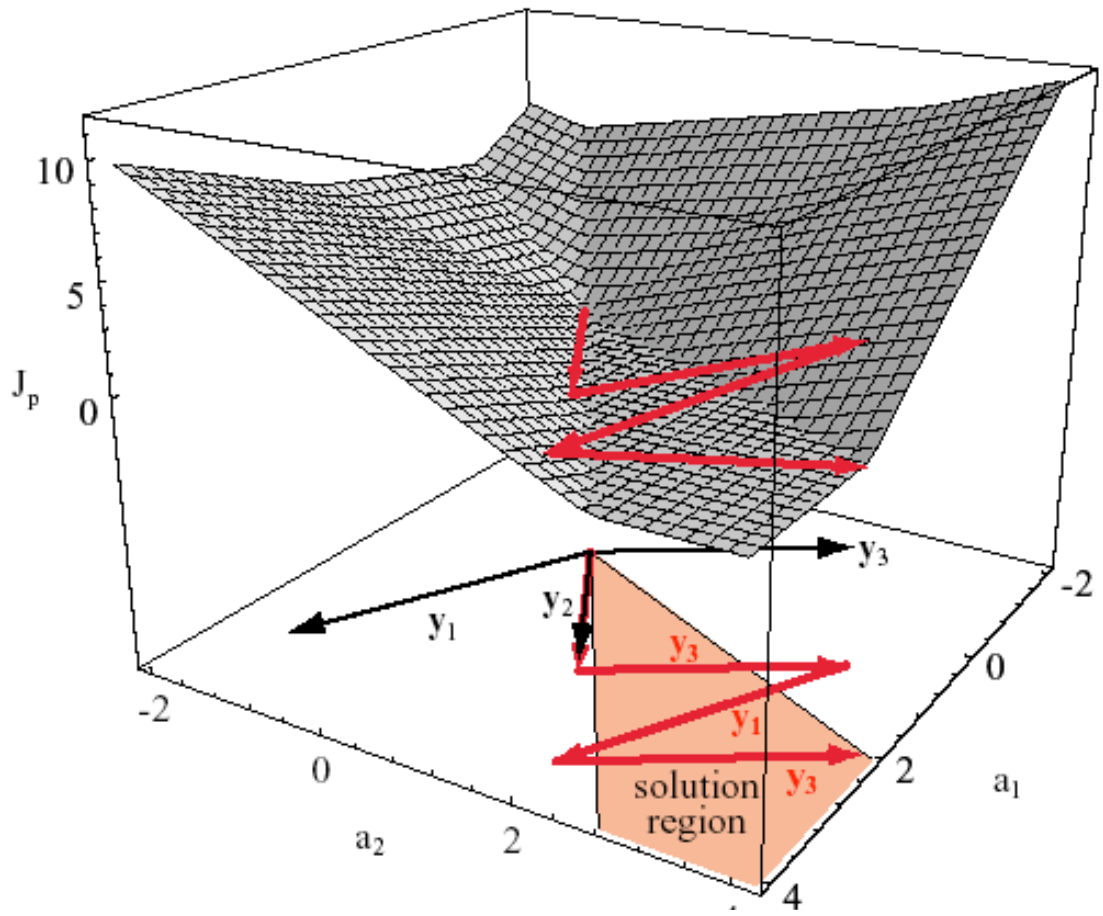


Figure 5.12: The Perceptron criterion, J_p is plotted as a function of the weights a_1 and a_2 for a three-pattern problem. The weight vector begins at $\mathbf{0}$, and the algorithm sequentially adds to it vectors equal to the "normalized" misclassified patterns themselves. In the example shown, this sequence is $\mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_1, \mathbf{y}_3$, at which time the vector lies in the solution region and iteration terminates. Note that the second update (by $\mathbf{y}_3$) takes the candidate vector *farther* from the solution region than after the first update (cf. Theorem 5.1. (In an alternate, batch method, *all* the misclassified points are added at each iteration step leading to a smoother trajectory in weight space.)

Simplest Case

$$\overset{\downarrow}{\mathbf{y}_1},\ \mathbf{y}_2,\ \overset{\downarrow}{\mathbf{y}_3},\ \overset{\downarrow}{\mathbf{y}_1},\ \overset{\downarrow}{\mathbf{y}_2},\ \mathbf{y}_3,\ \mathbf{y}_1,\ \overset{\downarrow}{\mathbf{y}_2},\ \ldots \tag{19}$$

are misclassified, then the sequence $\mathbf{y}^1,\ \mathbf{y}^2,\ \mathbf{y}^3,\ \mathbf{y}^4,\ \mathbf{y}^5, \ldots$ denotes the sequence $\mathbf{y}_1,\ \mathbf{y}_3,\ \mathbf{y}_1,\ \mathbf{y}_2,\ \mathbf{y}_2, \ldots$ With this understanding, the *fixed-increment rule* for generating a sequence of weight vectors can be written as

$$\left.\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k & k \geq 1 \end{array}\right\} \tag{20}$$

where $\mathbf{a}^t(k)\mathbf{y}^k \leq 0$ for all k. If we let n denote the total number of patterns, the algorithm is:

Algorithm 4 (Fixed-increment single-sample Perceptron)

```
1 begin initialize  a, k = 0
2         do  k ← (k + 1) mod n
3             if  y_k is misclassified by a  then  a ← a − y_k
4         until all patterns properly classified
5   return a
6 end
```

When does perceptron rule converge?

Theorem 5.1 (Perceptron Convergence) *If training samples are linearly separable then the sequence of weight vectors given by Algorithm 4 will terminate at a solution vector.*

$$\left.\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k & k \geq 1, \end{array}\right\}$$

where now $\mathbf{a}^t(k)\mathbf{y}^k \leq b$ for all k. Thus for n patterns, our algorithm is:

Algorithm 5 (Variable increment Perceptron with margin)

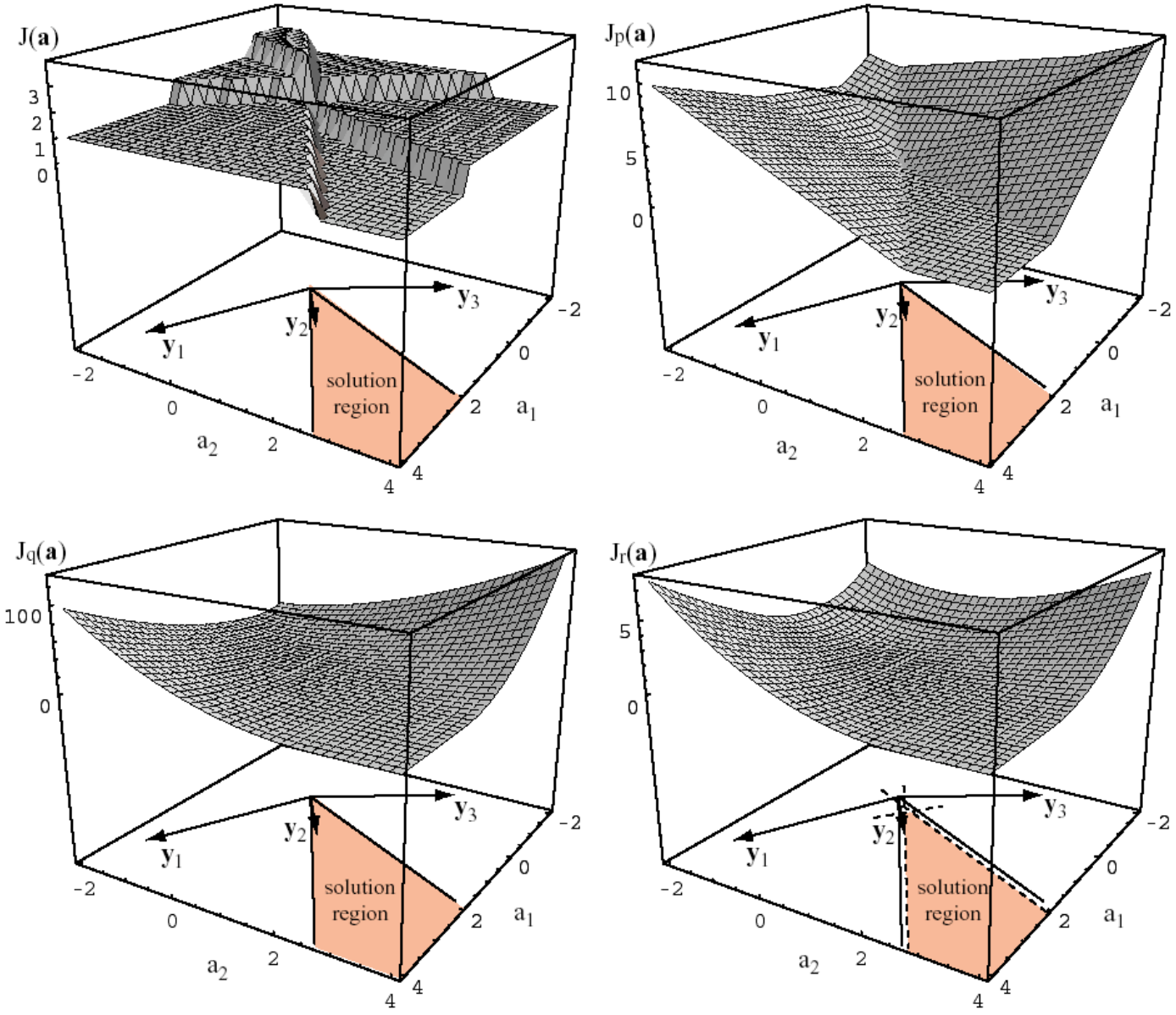
1 **begin initialize** $\mathbf{a}$, criterion θ, margin $b, \eta(\cdot), k = 0$
2 **do** $k \leftarrow k + 1$
3 **if** $\mathbf{a}^t\mathbf{y}_k + b < 0$ **then** $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\mathbf{y}_k$
4 **until** $\mathbf{a}^t\mathbf{y}_k + b \leq 0$ for all k
5 **return** $\mathbf{a}$
6 **end**

It can be shown that if the samples are linearly separable and if

$$\eta(k) \geq 0,$$

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = \infty$$

Different Error Functions



Four learning criteria as a function of weights in a linear classifier.

At the upper left is the total number of patterns misclassified, which is piecewise constant and hence unacceptable for gradient descent procedures.

At the upper right is the Perceptron criterion (Eq. 16), which is piecewise linear and acceptable for gradient descent.

The lower left is squared error (Eq. 32), which has nice analytic properties and is useful even when the patterns are not linearly separable.

The lower right is the square error with margin. A designer may adjust the margin b in order to force the solution vector to lie toward the middle of the $b = 0$ solution region in hopes of improving generalization of the resulting classifier.

Name	Criterion	Algorithm	Conditions
Fixed Increment	$J_p = \sum_{\mathbf{a}^t\mathbf{y}\le 0} (-\mathbf{a}^t\mathbf{y})$	$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k$ $(\mathbf{a}^t(k)\mathbf{y}^k \le 0)$	—
Variable Increment	$J'_p = \sum_{\mathbf{a}^t\mathbf{y}\le 0} -(\mathbf{a}^t\mathbf{y} - b)$	$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k$ $(\mathbf{a}^t(k)\mathbf{y}^k \le b)$	$\eta(k) \ge 0$ $\sum \eta(k) \to \infty$ $\frac{\sum \eta^2(k)}{(\sum \eta(k))^2} \to 0$
Relaxation	$J_r = \frac{1}{2}\sum_{\mathbf{a}^t\mathbf{y}\le b} \frac{(\mathbf{a}^t\mathbf{y}-b)^2}{\|\mathbf{y}\|^2}$	$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta\frac{b-\mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|^2}\mathbf{y}^k$ $(\mathbf{a}^t(k)\mathbf{y}^k \le b)$	$0 < \eta < 2$
Widrow-Hoff (LMS)	$J_s = \sum_i (\mathbf{a}^t\mathbf{y}_i - b_i)^2$	$\mathbf{a}(k+1) =$ $\mathbf{a}(k) + \eta(k)(b_k - \mathbf{a}^t(k)\mathbf{y}^k)\mathbf{y}^k$	$\eta(k) > 0$ $\eta(k) \to 0$
Stochastic Approx.	$J_m = \mathcal{E}\left[(\mathbf{a}^t\mathbf{y} - z)^2\right]$	$\mathbf{a}(k+1) =$ $\mathbf{a}(k) + \eta(k)(z_k - \mathbf{a}^t(k)\mathbf{y}^k)\mathbf{y}^k$	$\sum \eta(k) \to \infty$ $\sum \eta^2(k) \to L < \infty$
		$\mathbf{a}(k+1) =$ $\mathbf{a}(k) + \mathbf{R}(k)(z(k) - \mathbf{a}(k)^t\mathbf{y}^k)\mathbf{y}^k$	$\mathbf{R}^{-1}(k+1) = \mathbf{R}^{-1}(k) + \mathbf{y}_k\mathbf{y}_k^t$
Pseudo-inverse	$J_s = \|\mathbf{Ya} - \mathbf{b}\|^2$	$\mathbf{a} = \mathbf{Y}^\dagger\mathbf{b}$	—
Ho-Kashyap	$J_s = \|\mathbf{Ya} - \mathbf{b}\|^2$	$\mathbf{b}(k+1) = \mathbf{b}(k) + \eta(\mathbf{e}(k) + \|\mathbf{e}(k)\|)$ $\mathbf{e}(k) = \mathbf{Ya}(k) - \mathbf{b}(k)$ $\mathbf{a}(k) = \mathbf{Y}^\dagger\mathbf{b}(k)$	$0 < \eta < 1$ $\mathbf{b}(1) > 0$
		$\mathbf{b}(k+1) = \mathbf{b}(k) + \eta(\mathbf{e}(k) + (\|\mathbf{e}(k)\|)$ $\mathbf{a}(k+1) = \mathbf{a}(k) + \eta\mathbf{RY}^t\|\mathbf{e}(k)\|$	$\eta(k) =$ $\frac{\|\mathbf{e}(k)\|^t\mathbf{YRY}^t\|\mathbf{e}(k)\|}{\|\mathbf{e}(k)\|^t\mathbf{YRY}^t\mathbf{YRY}^t\|\mathbf{e}(k)\|}$ is optimum; $\mathbf{R}$ sym., pos. def.; $\mathbf{b}(1) > 0$

Three Different issues

1) Class boundary expressiveness
2) Error definition
3) Learning method

Fisher's linear disc.

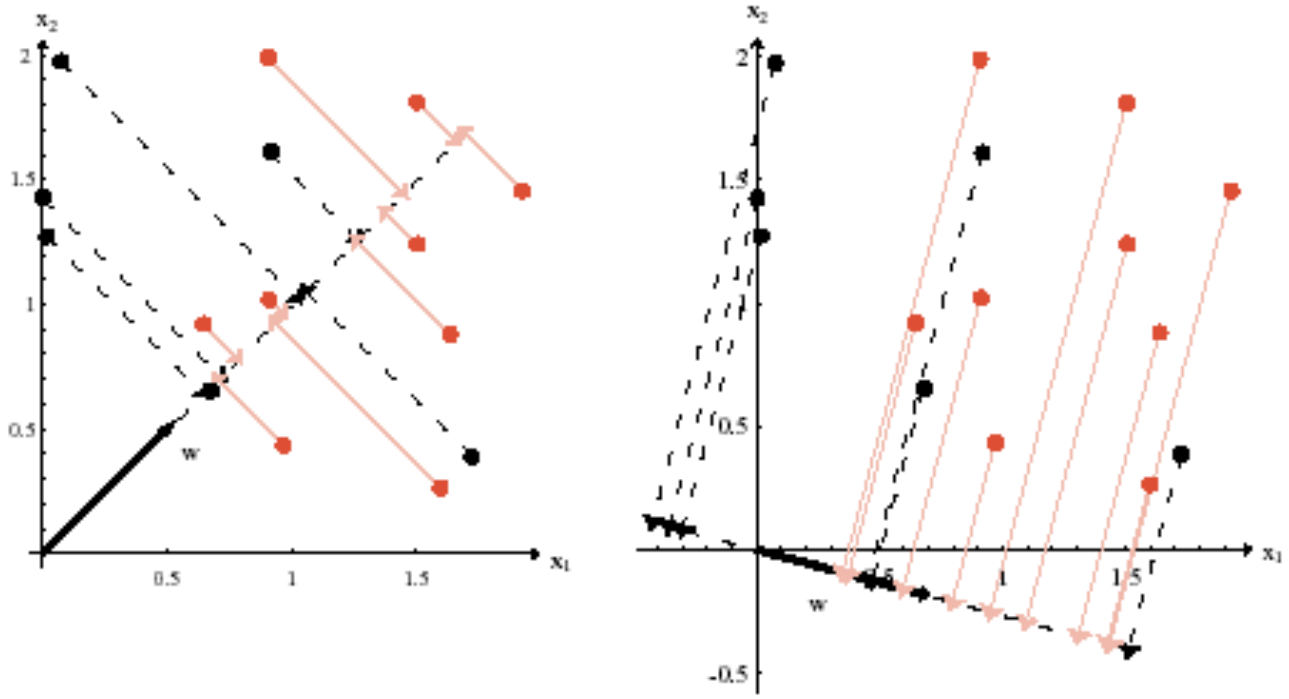
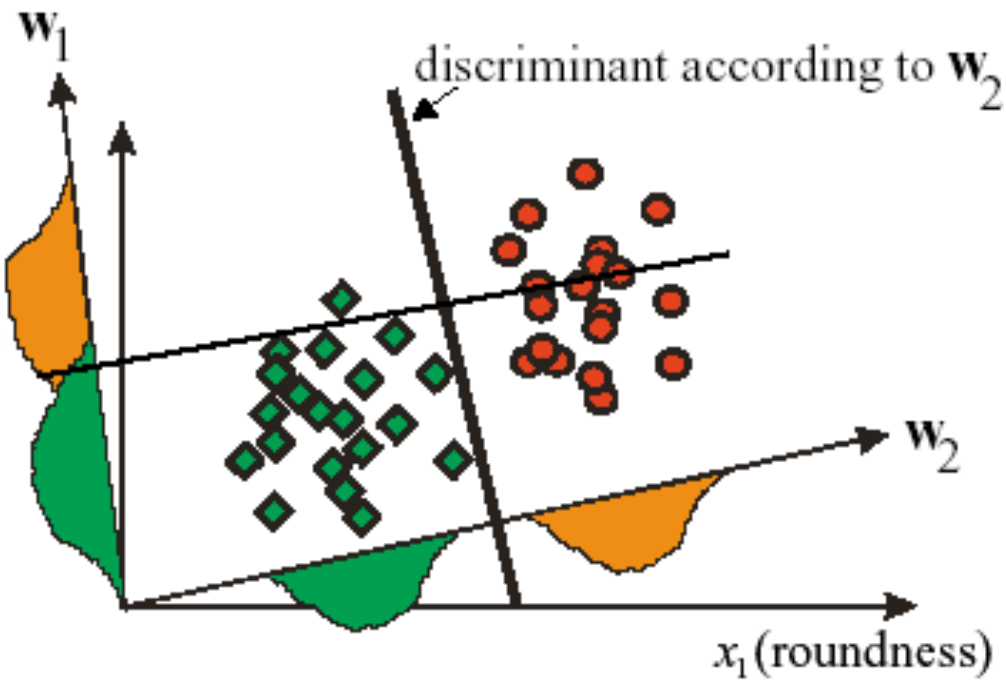


Figure 4.27: Projection of samples onto two different lines. The figure on the right shows greater separation between the red and black projected points.

Fisher's Linear Discriminant Intuition

What is the best discriminant?
The one for which the projected distributions are as wide apart as possible!

Wide apart, of course with respect to their variances.
Seek a **w** for which $\frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$ is maximal



$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

We are looking for a good projection $y = \mathbf{w}^t\mathbf{x}$

Write this in terms of *w*

$$\tilde{m}_i = \frac{1}{n_i}\sum_{y\in\mathcal{Y}_i} y = \frac{1}{n_i}\sum_{y\in\mathcal{Y}_i} \mathbf{w}^t\mathbf{x} = \mathbf{w}^t\mathbf{m}_i.$$

So, $|\tilde{m}_1 - \tilde{m}_2| = |\mathbf{w}^t(\mathbf{m}_1 - \mathbf{m}_2)|,$

Next, $$\tilde{s}_i^2 = \sum_{y\in\mathcal{Y}_i} (y - \tilde{m}_i)^2.$$

Define the scatter matrix:

$$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

Then,

$$\begin{aligned} \tilde{s}_i^2 &= \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{w}^t \mathbf{x} - \mathbf{w}^t \mathbf{m}_i)^2 \\ &= \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{w}^t (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t \mathbf{w} \\ &= \mathbf{w}^t \mathbf{S}_i \mathbf{w}; \end{aligned}$$

Thus the denominator can be written:

$$\tilde{s}_1^2 + \tilde{s}_2^2 = \mathbf{w}^t \mathbf{S}_W \mathbf{w}. \qquad \mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2.$$

Similarly, the separations of the projected means obeys

$$\begin{aligned}(\tilde{m}_1 - \tilde{m}_2)^2 &= (\mathbf{w}^t\mathbf{m}_1 - \mathbf{w}^t\mathbf{m}_2)^2 \\ &= \mathbf{w}^t(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t\mathbf{w} \\ &= \mathbf{w}^t\mathbf{S}_B\mathbf{w},\end{aligned}$$

where

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t.$$

Rayleigh Quotient

$$J(\mathbf{w}) = \frac{\mathbf{w}^t\mathbf{S}_B\mathbf{w}}{\mathbf{w}^t\mathbf{S}_W\mathbf{w}}.$$

Maximizing equivalent to solving:

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w},$$

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w}.$$

With solution:

$$\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2).$$

Fisher's criterion

Seek a $\mathbf{w}$ for which

$$J_F = \frac{\left|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)\right|^2}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \text{ is maximum.}$$

$\mathbf{m}_1$ and $\mathbf{m}_2$ are group means, and $\mathbf{S}_W$ is the `pooled within - class sample covariance matrix, given by :

$$\frac{1}{n\text{-}2}\left(n_1\hat{\Sigma}_1 + n_2\hat{\Sigma}_2\right)$$

where $\hat{\Sigma}_1$ and $\hat{\Sigma}_2$ are the covariance matrices of the classes respectively. The solution can be found by differentiating J_F with respect to $\mathbf{w}$ and put to zero. The solution is :

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

Fisher's criterion

But we still need a classification rule!

$\mathbf{w}^T\mathbf{x} + w_0 > 0$, but what is w_0 ?

From previous work on normal distributions with equal covariance matrix :

$$\mathbf{w} = \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

$$w_0 = -\tfrac{1}{2}(\mathbf{m}_1 + \mathbf{m}_2)\mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2) - \log\left(\frac{p(\omega_2)}{p(\omega_1)}\right)$$

The optimality of the following rule is of course only guaranteed for normal distributed data! Assign to ω_1 if

$$\left\{\mathbf{x} - \tfrac{1}{2}(\mathbf{m}_1 + \mathbf{m}_2)\right\}^T \mathbf{w} > \log\left(\frac{p(\omega_2)}{p(\omega_1)}\right)$$

Least mean squared error procedure

In the perceptron rule we only looked at the sign of $\mathbf{v}^T\mathbf{y}_i$.

In the LMS error procedure we assign a target value t_i to each sample.

The criterion is defined as :

$$J_s = \|\mathbf{Y}\mathbf{v} - \mathbf{t}\|,$$

in which $\mathbf{Y}$ is the matrix consisting of all $\mathbf{y}_i$'s and $\mathbf{t}$ the vector consisting of all t_i's.

The solution is :

$$\mathbf{v} = \left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{t}.$$

How well do we approximate the targets?

$$\hat{\mathbf{t}} = \mathbf{Y}\mathbf{v} = \mathbf{Y}\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T\mathbf{t}.$$

Sum of squared errors :

$$\left\|\hat{\mathbf{t}} - \mathbf{t}\right\|^2 = \left\|\left\{\mathbf{Y}\left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T - \mathbf{I}\right\}\mathbf{t}\right\|^2$$

Relationship to Fisher's discriminant

Assume that all samples of class ω_1 get label t_1, and all samples from ω_2 get label t_2. In that case the solution for $\mathbf{w}$ is :

$$\mathbf{w} = \frac{\alpha}{n}\mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2).$$

This is the same solution as Fisher's discriminant.

But how do we assign an unknown pattern $\mathbf{x}$?

If $\mathbf{w}^T\mathbf{x} + w_0$ is closer to t_1 than $-(\mathbf{w}^T\mathbf{x} + w_0)$ is to t_2 then assign it to class ω_1.

Or, assign to class ω_1 if :

$$\left(\mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)\right)^T(\mathbf{x} - \mathbf{m}) > \frac{t_1 + t_2}{2}\frac{n_2 - n_1}{\alpha}$$