Chapter 7 Linear Discriminant Function

Definition

$$g_i : \mathbf{R}^d \to \mathbf{R} \quad (1 \leq i \leq c)$$

- Useful way to represent classifiers
- One function per category

$$\begin{aligned}&\text{Decide } \omega_i\\&\text{if } g_i(\mathbf{x}) > g_j(\mathbf{x}) \text{ for all } j \neq i\end{aligned}$$

- Minimum Risk

$$g_i(\mathbf{x}) = -R(\alpha_i \mid \mathbf{x}) \quad (1 \leq i \leq c)$$

- Minimum

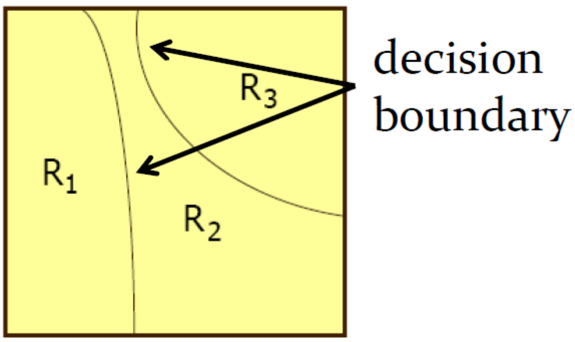
$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}) \quad (1 \leq i \leq c)$$

Decision region

$$\mathcal{R}_i = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{R}^d : g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall j \neq i\}$$
$$\text{where } \mathcal{R}_i \cap \mathcal{R}_j = \emptyset (i \neq j) \text{ and } \bigcup_{i=1}^{c} \mathcal{R}_i = \mathbf{R}^d$$

Decision boundary

surface in feature space where ties occur among several largest discriminant functions



Linear Discriminant Functions

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0}$$

- $\mathbf{w}_i$: weight vector which is d-dimensional
- $w_i 0$: bias/threshold

e.g.

$$\begin{aligned}
&\mathrm{x} = (x_1, x_2, x_3)^t && d = 3, c = 3\\
&g_1(\mathrm{x}) = x_1 - 2x_2 + 4x_3 && \mathrm{w}_1 = (1, -2, 4)^t, w_{10} = 0\\
&g_2(\mathrm{x}) = x_1 + 3x_3 + 4 && \mathrm{w}_2 = (1, 0, 3)^t, w_{20} = 4\\
&g_3(\mathrm{x}) = -2 && \mathrm{w}_3 = (0, 0, 0)^t, w_{30} = -2
\end{aligned}$$

Generalized Linear Discriminant Function

Quadratic discriminant function (二次判别函数)

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

Polynomial discriminant function (多项式判别函数)

- e.g. 3rd-order polynomial

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d} \sum_{j=1}^{d} \sum_{k=1}^{d} w_{ijk} x_i x_j x_k$$

Nesting version

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

- $\mathbf{a}$: the $\hat{d}$ -dimensional weight vector $(a_1, a_2, \ldots, a_{\hat{d}})^t$
- $\mathbf{y}$: the $\hat{d}$ -dimensional transformed feature vector $(y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_{\hat{d}}(\mathbf{x}))^t$, where $y_i(\mathbf{x})$ can be viewed as as feature detecting subsystem
- The resulting discriminant function $g(x)$ may not be linear in x, but it is linear in y.

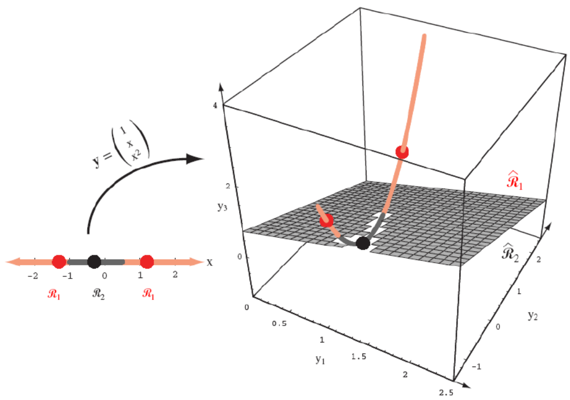
An example (quadratic in $\mathbf{x}$ ➔ linear in $\mathbf{y}$)

$$g(\mathbf{x}) = a_1 + a_2 x + a_3 x^2$$

$$\mathbf{a} = (a_1, a_2, a_3)^t$$

$$\mathbf{y} = \begin{pmatrix} y_1(\mathbf{x}) \\ y_2(\mathbf{x}) \\ y_3(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

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



$$\mathbf{a} = (-1, 1, 2)^t$$

Augmentation Representation

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i = w_0 + \mathbf{w}^t \mathbf{x} \Leftrightarrow g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$

where the augmented weight vector $\mathbf{a}$

$$\mathbf{a} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix} = \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}$$

the augmented feature vector $\mathbf{y}$

$$\mathbf{y} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$$

The Two-Category Case

Training set

$$\mathcal{D} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \quad \left(\mathbf{y}_i \in \mathbf{R}^{d+1}, \Omega = \{\omega_1, \omega_2\}\right)$$

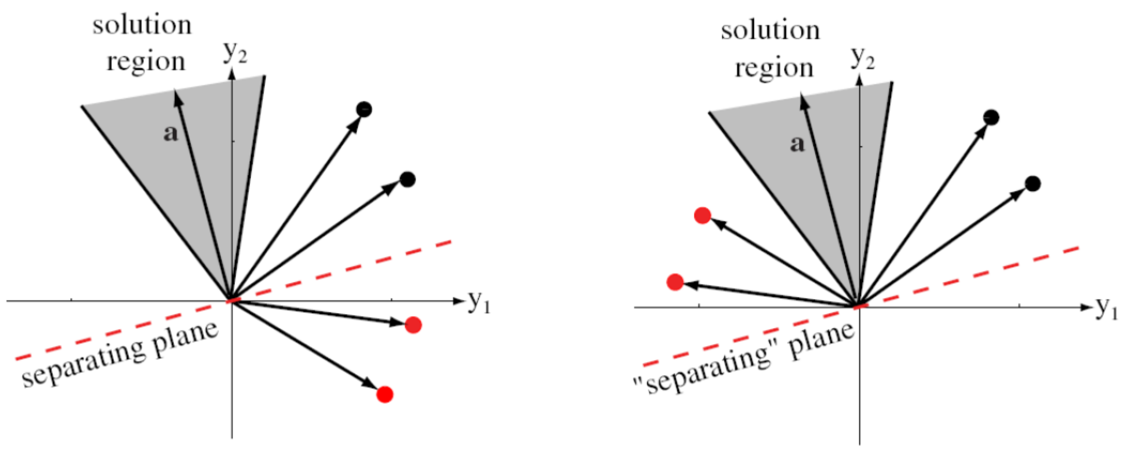
The task is determine the (augmented) weight vector $\mathbf{a} \in \mathbf{R}^{d+1}$ where $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$ can classify all training samples in $\mathcal{D}$ correctly, which is

$$\mathbf{a}^t\mathbf{y}_i > 0 \text{ if } \mathbf{y}_i \text{ is labeled } \omega_1$$
$$\mathbf{a}^t\mathbf{y}_i < 0 \text{ if } \mathbf{y}_i \text{ is labeled } \omega_2$$

Simplified treatment: "normalization", i.e. replace $\mathbf{y}_i$ with $-\mathbf{y}_i$ if it is labeled ω_2, that is

$$\mathbf{a}^t\mathbf{y}_i > 0 \text{ for all (normalized) } \mathbf{y}$$

a should be on the positive side of the hyperplane defined by $\mathbf{a}^t\mathbf{y}_i = 0$ with $\mathbf{y}_i$ being the norm vector



original training samples

normalized training samples

the **Solution Region** is the intersection of n half-spaces yielded by the n training samples,

Gradient Descent

Minimize a criterion/objective function (准则函数) $J(\mathbf{a})$ based on the normalized training samples $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ using Gradient Descent

- Basic strategy

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) - \eta \cdot \nabla f(\mathbf{x})^t \cdot \nabla f(\mathbf{x}) + O\left(\Delta\mathbf{x}^t \cdot \Delta\mathbf{x}\right)$$

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

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

Basic Gradient Descent

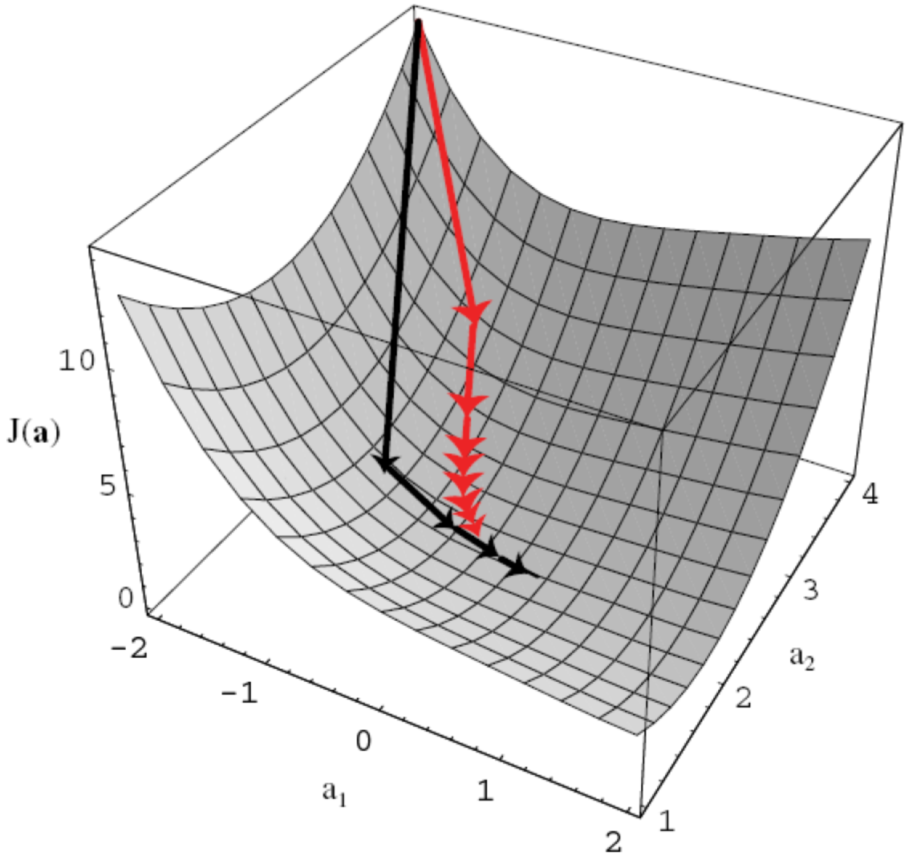
- Newton's Algorithm

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

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

1. **begin initialize** $\mathbf{a}$, threshold θ
2. **do**
3. $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{H}^{-1}\nabla J(\mathbf{a})$
4. **until** $\|\mathbf{H}^{-1}\nabla J(\mathbf{a})\| < \theta$
5. **return** $\mathbf{a}$
6. **end**

- Advantage: better step size than simple gradient descent
- Disadvantage: $O\left(d^3\right)$ complexity for matrix inversion; even not applicable if $\boldsymbol{H}$ is singular

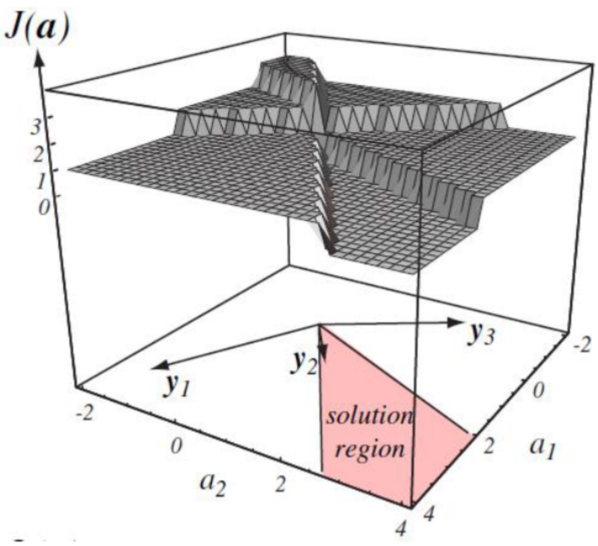


Perceptron Criterion Function

Given the **normalized** training samples $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$, set the criterion function $J(\mathbf{a})$ as the **number of examples misclassified by a**

$$J(\mathbf{a}) = \sum_{i=1}^{n} 1_{\mathbf{a}^t \mathbf{y}_i \leq 0}$$

which is a Piecewise constant function (分段常数函数)



Drawback

- Not compatible with the **gradient descent** procedure for function minimization
    - The gradient is almost 0 (except on the boundary of piecewise region)

A better choice

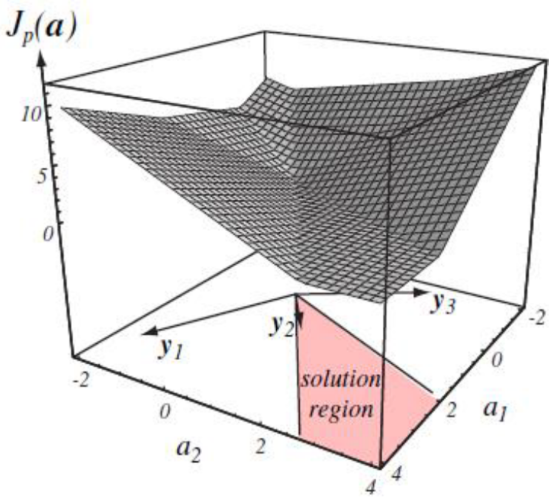
Define

$$\mathcal{Y} = \{\mathbf{y}_i \mid \mathbf{a}^t\mathbf{y}_i \leq 0, 1 \leq i \leq n\}$$

which is the set of samples misclassified by $\boldsymbol{a}$

then define criterion function as

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



- The gradient

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

- The iterative update rule

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

where $\mathcal{Y}_k$ is the set of samples misclassified by $\mathbf{a}(k)$

Batch Perception Algorithm

- The next weight vector is obtained by **adding some multiple of the sum of the misclassified samples** to the present weight vector
- Update (correct) a based on a **set** of misclassified samples

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

```
begin initialize a, threshold θ, η(·), k ← 0
    do k ← k + 1
        Y_k = {y_i | a^t y_i ≤ 0, 1 ≤ i ≤ n}
        a ← a + η(k) Σ_{y∈Y_k} y
    until ||η(k) Σ_{y∈Y_k} y|| < θ
  return a
end
```

Batch Perceptron

Single-Sample Correction

- Update (correct) a sequentially whenever it misclassifies a **single sample**
- Consider all training examples **cyclically**
- mark the misclassified samples in **sequence**

$$
\begin{array}{ccccccccc}
\downarrow & & \downarrow & \downarrow & \downarrow & & & \downarrow & \\
\mathbf{y}_1, & \mathbf{y}_2, & \mathbf{y}_3, & \mathbf{y}_1, & \mathbf{y}_2, & \mathbf{y}_3, & \mathbf{y}_1, & \mathbf{y}_2, & \cdots \\
\mathbf{y}^1 & & \mathbf{y}^2 & \mathbf{y}^3 & \mathbf{y}^4 & & & \mathbf{y}^5 &
\end{array}
$$

Fixed Increment

Setting learning rate $\eta(k) = 1$

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

```
begin initialize a, j ← 0, k ← 0
    do j ← j + 1
        i = ((j − 1) mod n) + 1
        if y_i is misclassified by a
        then k ← k + 1; y^k = y_i; a ← a + y^k
    until k is kept unchanged for n consecutive rounds
  return a
end
```

Fixed-Increment
Single-Sample Perceptron

We can prove the fact that

- If all training samples are linearly separable
- then the single-sample perception converges to a solution vector

Variable Increment

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

begin **initialize** $\mathbf{a}$, margin b, $\eta(\cdot)$, $j \leftarrow 0$, $k \leftarrow 0$

  do $j \leftarrow j+1$

    $i = ((j-1) \bmod n)+1$

    if $\mathbf{a}^t\mathbf{y}_i \leq b$

    then $k \leftarrow k+1$; $\mathbf{y}^k = \mathbf{y}_i$; $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\mathbf{y}^k$

  until k is kept unchanged for n consecutive rounds

 return $\mathbf{a}$

end

Variable-Increment Perceptron with Margin

where we required $\eta(k)$ that

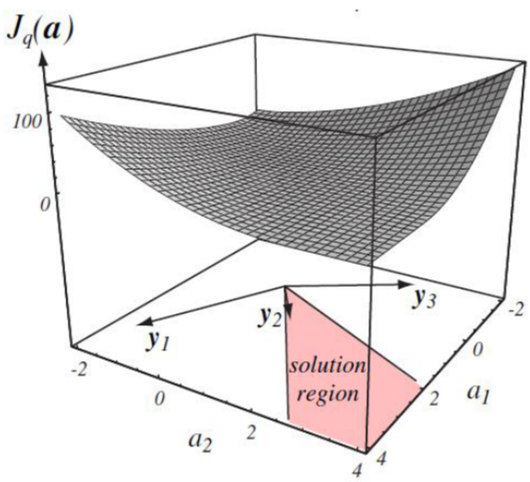
$$\eta(k) \geq 0$$
$$\lim_{m\to\infty}\sum_{k=1}^{m}\eta(k) = \infty$$
$$\lim_{m\to\infty}\frac{\sum_{k=1}^{m}\eta^2(k)}{\left(\sum_{k=1}^{m}\eta(k)\right)^2} = 0$$

so that it could converges with linearly separable training samples

Other Criterion Functions

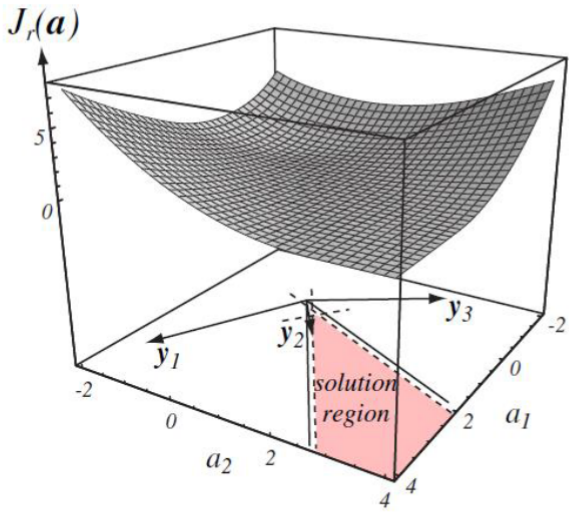
Quadratic Form

$$J_q(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}}\left(\mathbf{a}^t\mathbf{y}\right)^2$$
$$\mathcal{Y} = \{\mathbf{y}_i \mid \mathbf{a}^t\mathbf{y}_i \leq 0, 1 \leq i \leq n\}$$
$$\nabla J_q = 2\sum_{y\in\mathcal{Y}}\left(\mathbf{a}^t\mathbf{y}\right)\mathbf{y}$$



Distance Form

$$J_r(\mathbf{a}) = \frac{1}{2}\sum_{\mathbf{y}\in\mathcal{Y}}\frac{\left(\mathbf{a}^t\mathbf{y}-b\right)^2}{\|\mathbf{y}\|^2}$$
$$\mathcal{Y} = \{\mathbf{y}_i \mid \mathbf{a}^t\mathbf{y}_i \leq 0, 1 \leq i \leq n\}$$
$$\nabla J_r = \sum_{\mathbf{y}\in\mathcal{Y}}\frac{\mathbf{a}^t\mathbf{y}-b}{\|\mathbf{y}\|^2}\mathbf{y}$$



Minimum Squared Error (MSE)

$$\mathbf{a}^t\mathbf{y}_i = b_i$$

where

- b_i is some **arbitrarily** chosen **positive** constant
- updating for **all samples**

In other words, we need to solve

$$\mathbf{Y}\mathbf{a} = \mathbf{b}$$

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

where

- $\mathbf{Y}$ is a $n \times (d+1)$ matrix
- $\mathbf{a}$ is a $d+1$ dimensional weight vector
- $\mathbf{b}$ is a n dimensional column vector

Usually $n \geq d+1$, which means there is no exact solution for $\mathbf{a}$

Sum-of-squared-error criterion function

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

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

Let $\nabla J_s = 0$, we have

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

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

For Batch mode

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

For Single-sample mode

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

Algorithm

<u>**begin**</u> <u>**initialize**</u> $\mathbf{a}$, $\mathbf{b}$, threshold θ, $\eta(\cdot)$, $k \leftarrow 0$

    <u>**do**</u> $k \leftarrow k+1$

        $i = ((k-1) \bmod n) + 1$

        $\mathbf{y}^k \leftarrow \mathbf{y}_i;\ b_k \leftarrow b_i;\ \mathbf{a} \leftarrow \mathbf{a} + \eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k$

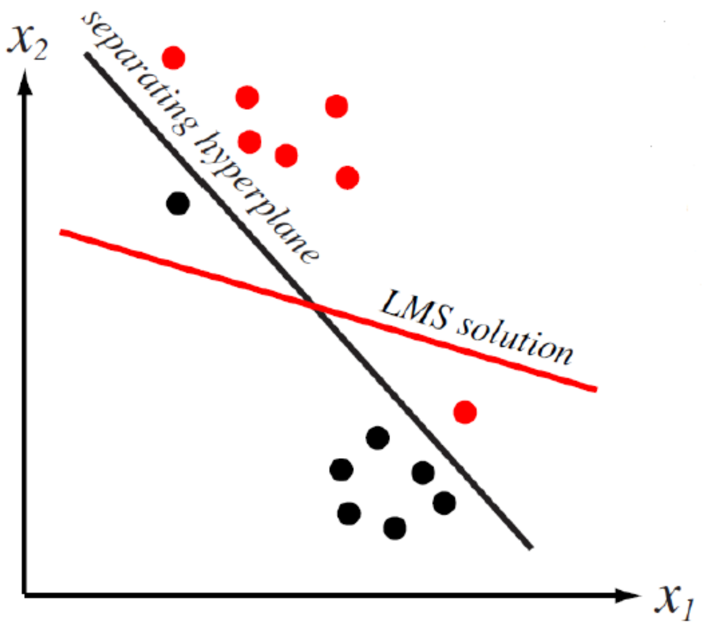
    <u>**until**</u> $\|\eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k\| < \theta$

  <u>**return**</u> $\mathbf{a}$

<u>**end**</u>

LMS
(least-mean-squared)

- A common choice of $\eta(k)$ for LMS: $\eta(k) = \frac{1}{k}$
- The LMS solution minimizes the sum of squared errors
    - i.e. the (squared) distance of the training samples to the decision hyperplane
    - LMS solution may not be a separating hyperplane



Multi-Category Generalization

$$\mathcal{D} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}\ \left(\mathbf{y}_k \in \mathbf{R}^{d+1}, \Omega = \{\omega_1, \ldots, \omega_c\}\right)$$

Task

Determine the set of (augmented) weight vectors $\{\mathbf{a}_i \in \mathbf{R}^{d+1} \mid 1 \leq i \leq c\}$ where $g_i(\mathbf{x}) = \mathbf{a}_i^t \mathbf{y} (1 \leq i \leq c)$ can classify all training samples in $\mathcal{D}$ correctly

if $\mathbf{y}_k (1 \leq k \leq n)$ is labeled ω_i

$$\forall j \neq i : \mathbf{a}_i^t \mathbf{y}_k > \mathbf{a}_j^t \mathbf{y}_k$$

Keslar's Construction

Transform the **multi-category** classification problem into the **binary** classification problem

$\{\mathbf{a}_i \in \mathbf{R}^{d+1} \mid 1 \leq i \leq c\}$

$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_c \end{bmatrix}$$

binary classification model: ***c*(*d*+1)-dimensional weight vector**

if $\mathbf{y}_k$ $(1 \leq k \leq n)$ is labeled ω_i

$$\boldsymbol{\eta}_{i1}^k = \begin{bmatrix} -\mathbf{y}_k \\ \vdots \\ \mathbf{y}_k \\ \vdots \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \quad \boldsymbol{\eta}_{ij}^k = \begin{array}{c} \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{y}_k \\ \vdots \\ -\mathbf{y}_k \\ \vdots \\ \mathbf{0} \end{bmatrix} \end{array} \begin{array}{l} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \\ \end{array} \quad \ldots, \boldsymbol{\eta}_{in}^k = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{y}_k \\ \vdots \\ \mathbf{0} \\ \vdots \\ -\mathbf{y}_k \end{bmatrix}$$

binary (normalized) training set: ***n*(*c*-1) training samples each being *c*(*d*+1)-dimensional**