

LINEAR CLASSIFIERS

❖ The Problem: Consider a two class task with ω_1, ω_2

$$\begin{aligned} g(\underline{x}) &= \underline{w}^T \underline{x} + w_0 = 0 = \\ \Rightarrow w_1 x_1 + w_2 x_2 + \dots + w_l x_l + w_0 \end{aligned}$$

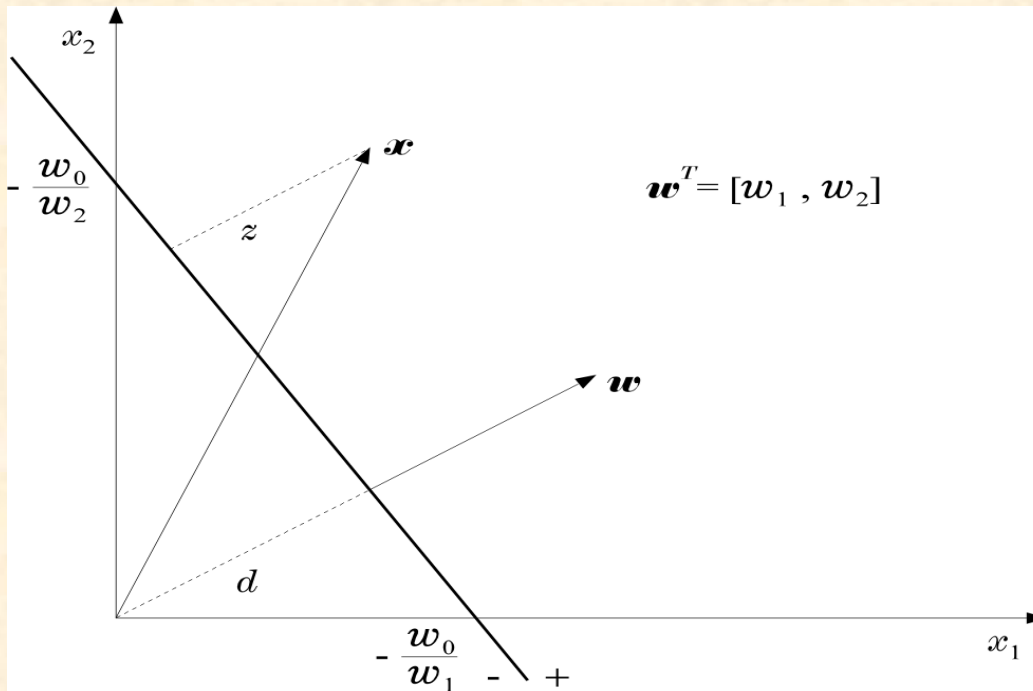
Assume $\underline{x}_1, \underline{x}_2$ on the decision hyperplane:

$$\begin{aligned} \Rightarrow 0 &= \underline{w}^T \underline{x}_1 + w_0 = \underline{w}^T \underline{x}_2 + w_0 \Rightarrow \\ \underline{w}^T (\underline{x}_1 - \underline{x}_2) &= 0 \quad \forall \underline{x}_1, \underline{x}_2 \end{aligned}$$

➤ Hence:

$\underline{w}$ $\perp$ on the hyperplane

$$g(\underline{x}) = \underline{w}^T \underline{x} + w_0 = 0$$



$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}}, \quad z = \frac{|g(\underline{x})|}{\sqrt{w_1^2 + w_2^2}}$$

❖ The Perceptron Algorithm

- Assume linearly separable classes, i.e.,

$$\exists \underline{w}^* : \underline{w}^{*T} \underline{x} > 0 \quad \forall \underline{x} \in \omega_1$$

$$\underline{w}^{*T} \underline{x} < 0 \quad \forall \underline{x} \in \omega_2$$

- The case $\underline{w}^{*T} \underline{x} + w_0^*$ falls under the above formulation, since

$$\bullet \quad \underline{w}' \equiv \begin{bmatrix} \underline{w}^* \\ w_0^* \end{bmatrix}, \quad \underline{x}' = \begin{bmatrix} \underline{x} \\ 1 \end{bmatrix}$$

$$\bullet \quad \underline{w}^{*T} \underline{x} + w_0^* = \underline{w}'^T \underline{x}' = 0$$

- Our goal: Compute a solution, i.e., a hyperplane $\underline{w}$, so that

$$\underline{w}^T \underline{x} (> <) 0 \quad \underline{x} \in \begin{array}{l} \omega_1 \\ \omega_2 \end{array}$$

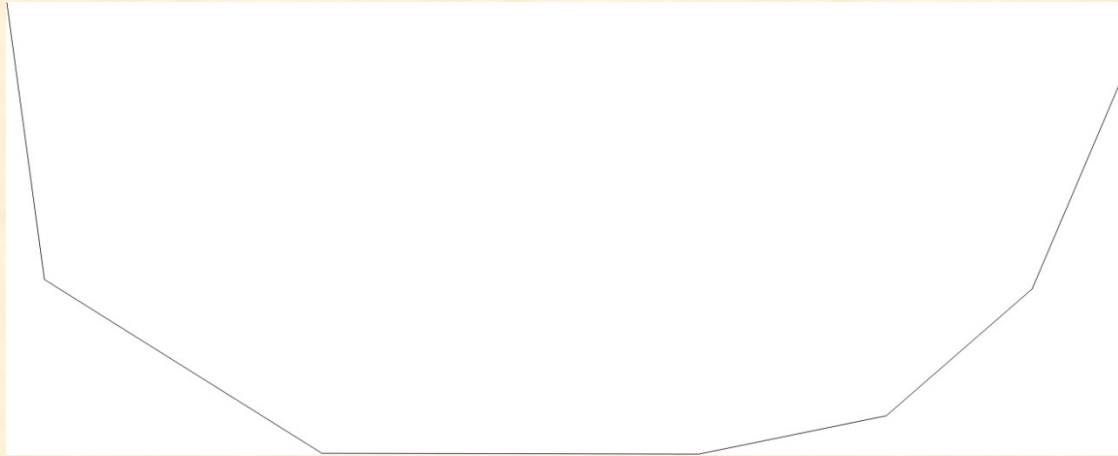
- The steps
 - Define a cost function to be minimized
 - Choose an algorithm to minimize the cost function
 - The minimum corresponds to a solution

➤ The Cost Function

$$J(\underline{w}) = \sum_{\underline{x} \in Y} (\delta_{\underline{x}} \underline{w}^T \underline{x})$$

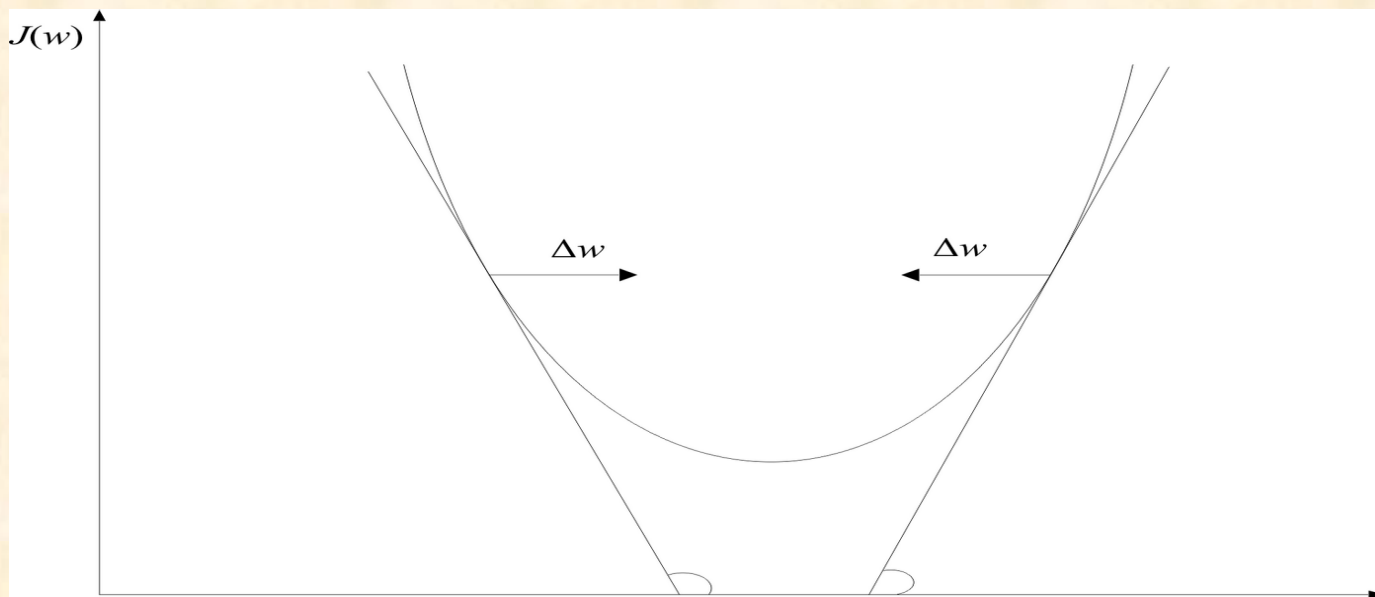
- Where Y is the subset of the vectors **wrongly** classified by $\underline{w}$. When Y =(empty set) a solution is achieved and
- $J(\underline{w}) = 0$
- $\delta_{\underline{x}} = -1$ if $\underline{x} \in Y$ and $\underline{x} \in \omega_1$
 $\delta_{\underline{x}} = +1$ if $\underline{x} \in Y$ and $\underline{x} \in \omega_2$
- $J(\underline{w}) \geq 0$

- $J(\underline{w})$ is piecewise linear (WHY?)



➤ The Algorithm

- The philosophy of the gradient descent is adopted.



$$\underline{w}(\text{new}) = \underline{w}(\text{old}) + \Delta \underline{w}$$

$$\Delta \underline{w} = -\mu \frac{\partial J(\underline{w})}{\partial \underline{w}} \Big|_{\underline{w} = \underline{w}(\text{old})}$$

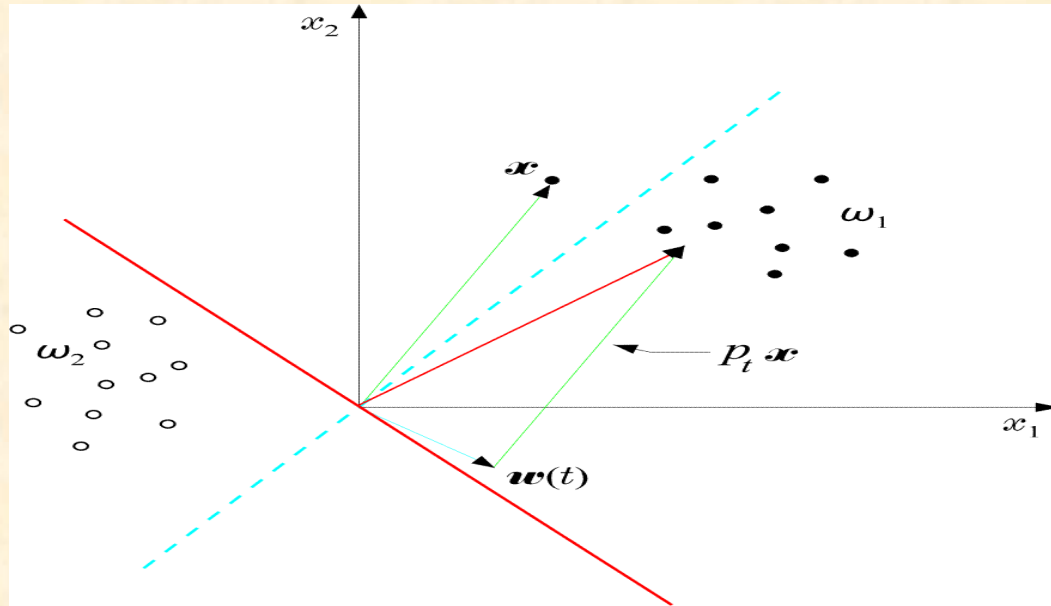
- Wherever valid

$$\frac{\partial J(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}} \left(\sum_{\underline{x} \in Y} \delta_x \underline{w}^T \underline{x} \right) = \sum_{\underline{x} \in Y} \delta_x \underline{x}$$

- $$\underline{w}(t+1) = \underline{w}(t) - \rho_t \sum_{\underline{x} \in Y} \delta_x \underline{x}$$

This is the celebrated **Perceptron Algorithm**

➤ An example:



$$\begin{aligned}\underline{w}(t+1) &= \underline{w}(t) + \rho_t x \\ &= \underline{w}(t) - \rho_t \delta_x x \quad (\delta_x = -1)\end{aligned}$$

- The perceptron algorithm **converges** in a **finite** number of iteration steps to a solution if

$$\lim_{t \rightarrow \infty} \sum_{k=0}^t \rho_k \rightarrow \infty,$$

$$\lim_{t \rightarrow \infty} \sum_{k=0}^t \rho_k^2 < +\infty$$

e.g., $\rho_t = \frac{c}{t}$

❖ A useful variant of the perceptron algorithm

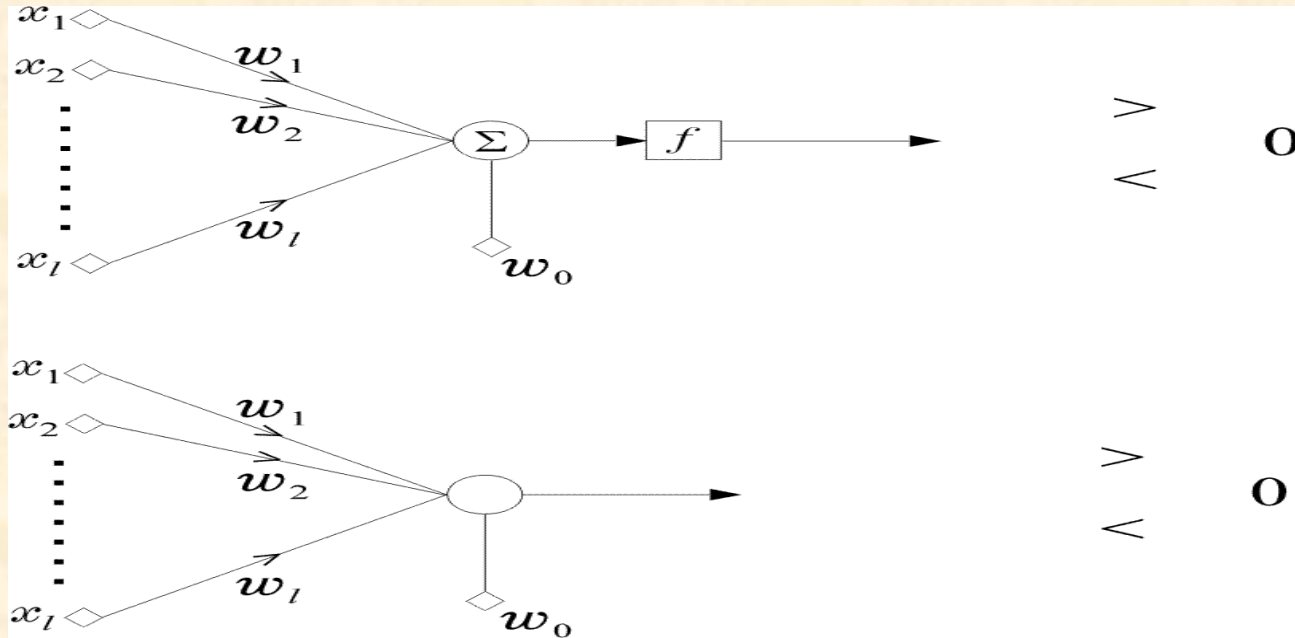
$$\underline{w}(t+1) = \underline{w}(t) + \rho \underline{x}_{(t)}, \quad \begin{array}{l} \underline{w}^T(t) \underline{x}_{(t)} \leq 0 \\ \underline{x}_{(t)} \in \omega_1 \end{array}$$

$$\underline{w}(t+1) = \underline{w}(t) - \rho \underline{x}_{(t)}, \quad \begin{array}{l} \underline{w}^T(t) \underline{x}_{(t)} \geq 0 \\ \underline{x}_{(t)} \in \omega_2 \end{array}$$

$$\underline{w}(t+1) = \underline{w}(t) \quad \text{otherwise}$$

- It is a **reward and punishment** type of algorithm

❖ The perceptron



w_i 's synapses or synaptic weights

w_0 threshold

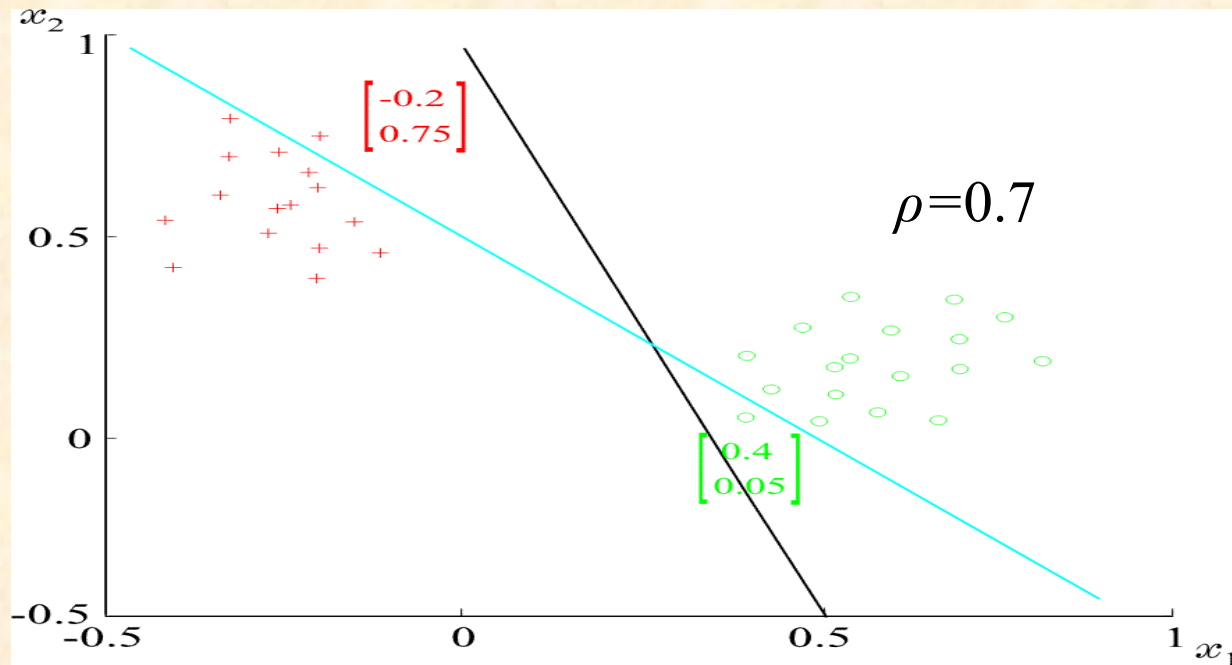
- The network is called **perceptron or neuron**
 - It is a **learning machine** that **learns** from the **training vectors** via the **perceptron algorithm**

➤ **Example:** At some stage t the perceptron algorithm results in

$$w_1 = 1, w_2 = 1, w_0 = -0.5$$

$$x_1 + x_2 - 0.5 = 0$$

The corresponding hyperplane is



$$\underline{w}(t+1) = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7(-1) \begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} - 0.7(+1) \begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$

❖ Least Squares Methods

- If classes are linearly separable, the perceptron output results in
- If classes are NOT linearly separable, we shall compute the weights w_0

so that the difference between

- The actual output of the classifier, $\underline{w}^T \underline{x}$, and
- The desired outputs, e.g.
 - +1 if $\underline{x} \in \omega_1$
 - 1 if $\underline{x} \in \omega_2$

to be SMALL

➤ **SMALL**, in the **mean square** error sense, means to choose $\underline{w}$ so that the cost function

- $J(\underline{w}) \equiv E[(y - \underline{w}^T \underline{x})^2]$ is minimum
- $\hat{\underline{w}} = \arg \min_{\underline{w}} J(\underline{w})$
- y the corresponding desired responses

➤ Minimizing

$J(\underline{w})$ w.r. to $\underline{w}$ results in :

$$\begin{aligned}\frac{\partial J(\underline{w})}{\partial \underline{w}} &= \frac{\partial}{\partial \underline{w}} E[(y - \underline{w}^T x)^2] = 0 \\ &= 2E[\underline{x}(y - \underline{x}^T \underline{w})] \Rightarrow \\ E[\underline{x}\underline{x}^T] \underline{w} &= E[\underline{x}y] \Rightarrow\end{aligned}$$

where R_x is the autocorrelation matrix

$$\hat{\underline{w}} = R_x^{-1} E[\underline{x}y]$$

and $R_x \equiv E[\underline{x}\underline{x}^T] =$ $\begin{bmatrix} E[x_1x_1] & E[x_1x_2] \dots & E[x_1x_l] \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots \\ E[x_lx_1] & E[x_lx_2] \dots & E[x_lx_l] \end{bmatrix}$

the crosscorrelation vector

$$E[\underline{x}y] = \begin{bmatrix} E[x_1y] \\ \dots \\ E[x_ly] \end{bmatrix}$$

➤ Multi-class generalization

- The goal is to compute M linear discriminant functions:

$$g_i(\underline{x}) = \underline{w}_i^T \underline{x}$$

according to the MSE.

- Adopt as desired responses y_i :

$$y_i = 1 \quad \text{if } \underline{x} \in \omega_i$$
$$y_i = 0 \quad \text{otherwise}$$

- Let

- And the matrix $\underline{y} = [y_1, y_2, \dots, y_M]^T$

$$W = [\underline{w}_1, \underline{w}_2, \dots, \underline{w}_M]$$

- The goal is to compute : W

$$\hat{W} = \arg \min_W E \left[\left\| \underline{y} - W^T \underline{x} \right\|^2 \right] = \arg \min_W E \left[\sum_{i=1}^M \left(y_i - \underline{w}_i^T \cdot \underline{x} \right)^2 \right]$$

- The above is equivalent to a number M of MSE minimization problems. That is:

Design each $\underline{w}_i$ so that its desired output is 1 for $\underline{x} \in \omega_i$ and 0 for any other class.

➤ **Remark:** The MSE criterion belongs to a more general class of cost function with the following **important** property:

- The value of $\hat{w}_i$ is an **estimate, in the MSE sense**, of the a-posteriori probability $P(\omega_i | \underline{x})$, provided that the desired responses used during training are $y_i = 1, \underline{x} \in \omega_i$ and 0 otherwise.

$$g_i(\underline{x})$$

$$P(\omega_i | \underline{x})$$

$$y_i = 1, \underline{x} \in \omega_i$$

- **Mean square error regression:** Let $\underline{y} \in \mathbb{R}^M$, $\underline{x} \in \mathbb{R}^p$ be jointly distributed random vectors with a joint pdf $p(\underline{x}, \underline{y})$
- The goal: **Given** the value of $\underline{x}$ **estimate** the value of $\underline{y}$. In the pattern recognition framework, given $\underline{x}$ one wants to estimate the respective label $y = \pm 1$

- The MSE estimate $\hat{\underline{y}}$ of $\underline{y}$ given $\underline{x}$ is defined as:

$$\hat{\underline{y}} = \arg \min_{\tilde{\underline{y}}} E[\|\underline{y} - \tilde{\underline{y}}\|^2]$$

- It turns out that:

$$\hat{\underline{y}} = E[\underline{y} | \underline{x}] \equiv \int_{-\infty}^{+\infty} \underline{y} p(\underline{y} | \underline{x}) d\underline{y}$$

The above is known as the **regression** of $\underline{y}$ given $\underline{x}$ and it is, in general, a non-linear function of $\underline{x}$. If $p(\underline{x}, \underline{y})$ is **Gaussian** the **MSE regressor is linear**.

❖ SMALL in the **sum of error** squares sense means

$$\triangleright J(\underline{w}) = \sum_{i=1}^N (y_i - \underline{w}^T \underline{x}_i)^2$$

$(y_i, \underline{x}_i)$ training pairs
its

that is, the input $\underline{x}_i$ and
 y_i

$$\frac{\partial J(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}} \sum_{i=1}^N (y_i - \underline{w}^T \underline{x}_i)^2 \stackrel{\text{corresponding class label}}{=} 0 \Rightarrow$$

$\triangleright$

$$\left(\sum_{i=1}^N \underline{x}_i \underline{x}_i^T \right) \underline{w} = \sum_{i=1}^N \underline{x}_i y_i$$

❖ Pseudoinverse Matrix

➤ Define

$$X = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \dots \\ \underline{x}_N^T \end{bmatrix} \quad (\text{an } N \times l \text{ matrix})$$

$$\underline{y} = \begin{bmatrix} y_1 \\ \dots \\ y_N \end{bmatrix} \quad \text{corresponding desired responses}$$

➤

$$\text{➤} \quad X^T = [\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N] \quad (\text{an } l \times N \text{ matrix})$$

$$\text{➤} \quad X^T X = \sum_{i=1}^N \underline{x}_i \underline{x}_i^T$$

$$X^T \underline{y} = \sum_{i=1}^N \underline{x}_i y_i$$

Thus
$$\left(\sum_{i=1}^N \underline{x}_i^T \underline{x}_i\right) \underline{\hat{w}} = \left(\sum_{i=1}^N \underline{x}_i^T \underline{y}_i\right)$$

$$(X^T X) \underline{\hat{w}} = X^T \underline{y} \Rightarrow$$

$$\underline{\hat{w}} = (X^T X)^{-1} X^T \underline{y}$$

$$= X^\# \underline{y}$$

$$X^\# \equiv (X^T X)^{-1} X^T \quad \text{Pseudoinverse of } X$$

➤ Assume $N=l \Rightarrow X$ square and invertible. Then

$$(X^T X)^{-1} X^T = X^{-1} X^{-T} X^T = X^{-1} \Rightarrow$$

$$X^\# = X^{-1}$$

- Assume $N > l$. Then, in general, there is no solution to satisfy all equations simultaneously:

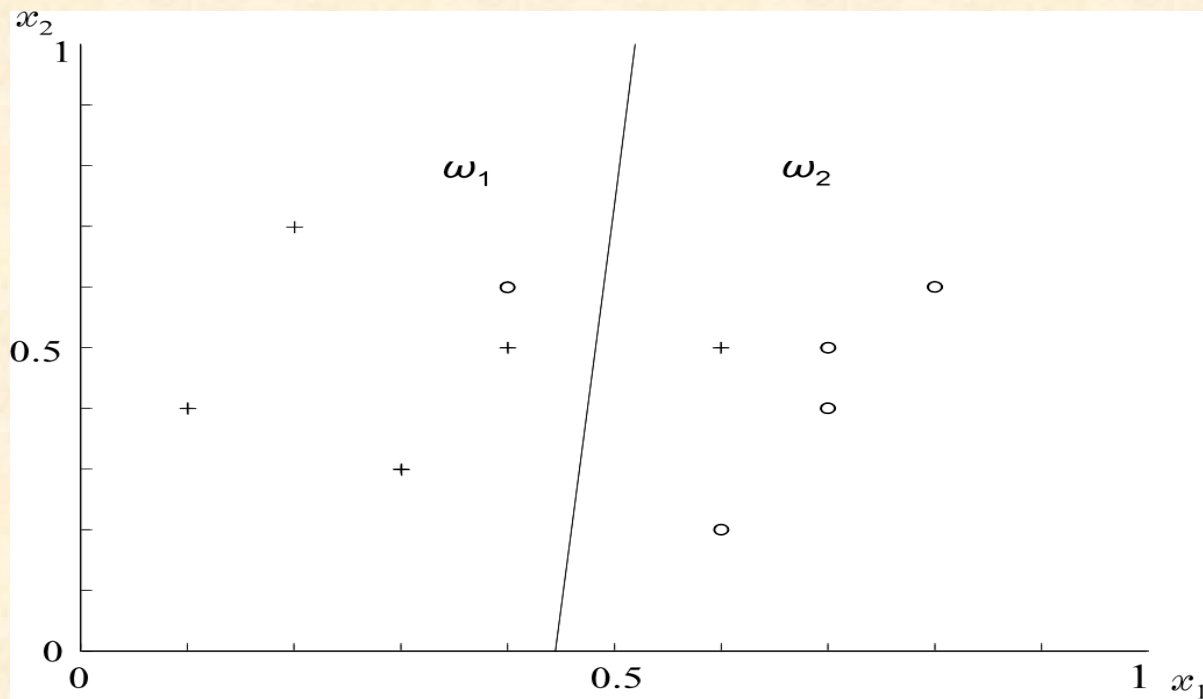
$$\begin{array}{lcl}
 & \underline{x}_1^T \underline{w} = y_1 & \\
 X \underline{w} = \underline{y}: & \underline{x}_2^T \underline{w} = y_2 & N \text{ equations} > l \text{ unknowns} \\
 & \dots & \\
 & \underline{x}_N^T \underline{w} = y_N &
 \end{array}$$

- The “solution” $\underline{w} = X^\# \underline{y}$ corresponds to the minimum sum of squares solution

➤ Example:

$$\omega_1 : \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0.6 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix}, \begin{bmatrix} 0.2 \\ 0.7 \end{bmatrix}, \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix}$$

$$\omega_2 : \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}, \begin{bmatrix} 0.6 \\ 0.2 \end{bmatrix}, \begin{bmatrix} 0.7 \\ 0.4 \end{bmatrix}, \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}, \begin{bmatrix} 0.7 \\ 0.5 \end{bmatrix}$$



$$X = \begin{bmatrix} 0.4 & 0.5 & 1 \\ 0.6 & 0.5 & 1 \\ 0.1 & 0.4 & 1 \\ 0.2 & 0.7 & 1 \\ 0.3 & 0.3 & 1 \\ 0.4 & 0.6 & 1 \\ 0.6 & 0.2 & 1 \\ 0.7 & 0.4 & 1 \\ 0.8 & 0.6 & 1 \\ 0.7 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} = \underline{y}$$

$$\blacktriangleright \quad X^T X = \begin{bmatrix} 2.8 & 2.24 & 4.8 \\ 2.24 & 2.41 & 4.7 \\ 4.8 & 4.7 & 10 \end{bmatrix}, X^T \underline{y} = \begin{bmatrix} -1.6 \\ 0.1 \\ 0.0 \end{bmatrix}$$

$$\underline{w} = (X^T X)^{-1} X^T \underline{y} = \begin{bmatrix} -3.13 \\ 0.24 \\ 1.34 \end{bmatrix}$$

❖ The Bias – Variance Dilemma

A classifier $g(\underline{x})$ is a **learning machine** that tries to **predict** the class label y of $\underline{x}$. In practice, a **finite** data set D is used for its training. Let us **write** $g(\underline{x}; D)$. Observe that:

- For **some** training sets $D = \{(y_i, \underline{x}_i), i = 1, 2, \dots, N\}$ the training may result to good estimates, for **some others** the result may be worse.
- The average performance of the classifier can be tested against the MSE optimal value, in the mean squares sense, that is:

$$E_D \left[(g(\underline{x}; D) - E[y | \underline{x}])^2 \right]$$

where E_D is the mean over all possible data sets D .

- The above is written as:

$$E_D \left[\left(g(\underline{x}; D) - E[y | \underline{x}] \right)^2 \right] = \\ \left(E_D \left[g(\underline{x}; D) \right] - E[y | \underline{x}] \right)^2 + E_D \left[\left(g(\underline{x}; D) - E_D \left[g(\underline{x}; D) \right] \right)^2 \right]$$

- In the above, the **first** term is the contribution of the **bias** and the second term is the contribution of the **variance**.
- For a finite D , there is a trade-off between the two terms. **Increasing bias it reduces variance and vice versa**. This is known as **the bias-variance dilemma**.
- Using a **complex** model results in **low-bias** but a **high variance**, as one changes from one training set to another. Using a **simple** model results in **high bias** but **low variance**.

❖ LOGISTIC DISCRIMINATION

- Let an M -class task, $\omega_1, \omega_2, \dots, \omega_M$. In logistic discrimination, the logarithm of the **likelihood ratios** are modeled via **linear** functions, i.e.,

$$\ln \left(\frac{P(\omega_i | \underline{x})}{P(\omega_M | \underline{x})} \right) = w_{i,0} + \underline{w}_i^T \underline{x}, \quad i = 1, 2, \dots, M-1$$

- Taking into account that

$$\sum_{i=1}^M P(\omega_i | \underline{x}) = 1$$

it can be easily shown that the above is equivalent with modeling posterior probabilities as:

$$P(\omega_M | \underline{x}) = \frac{1}{1 + \sum_{i=1}^{M-1} \exp(w_{i,0} + \underline{w}_i^T \underline{x})}$$

$$P(\omega_i | \underline{x}) = \frac{\exp(w_{i,0} + \underline{w}_i^T \underline{x})}{1 + \sum_{i=1}^{M-1} \exp(w_{i,0} + \underline{w}_i^T \underline{x})}, i = 1, 2, \dots, M-1$$

➤ For the two-class case it turns out that

$$P(\omega_2 | \underline{x}) = \frac{1}{1 + \exp(w_0 + \underline{w}^T \underline{x})}$$

$$P(\omega_1 | \underline{x}) = \frac{\exp(w_0 + \underline{w}^T \underline{x})}{1 + \exp(w_0 + \underline{w}^T \underline{x})}$$

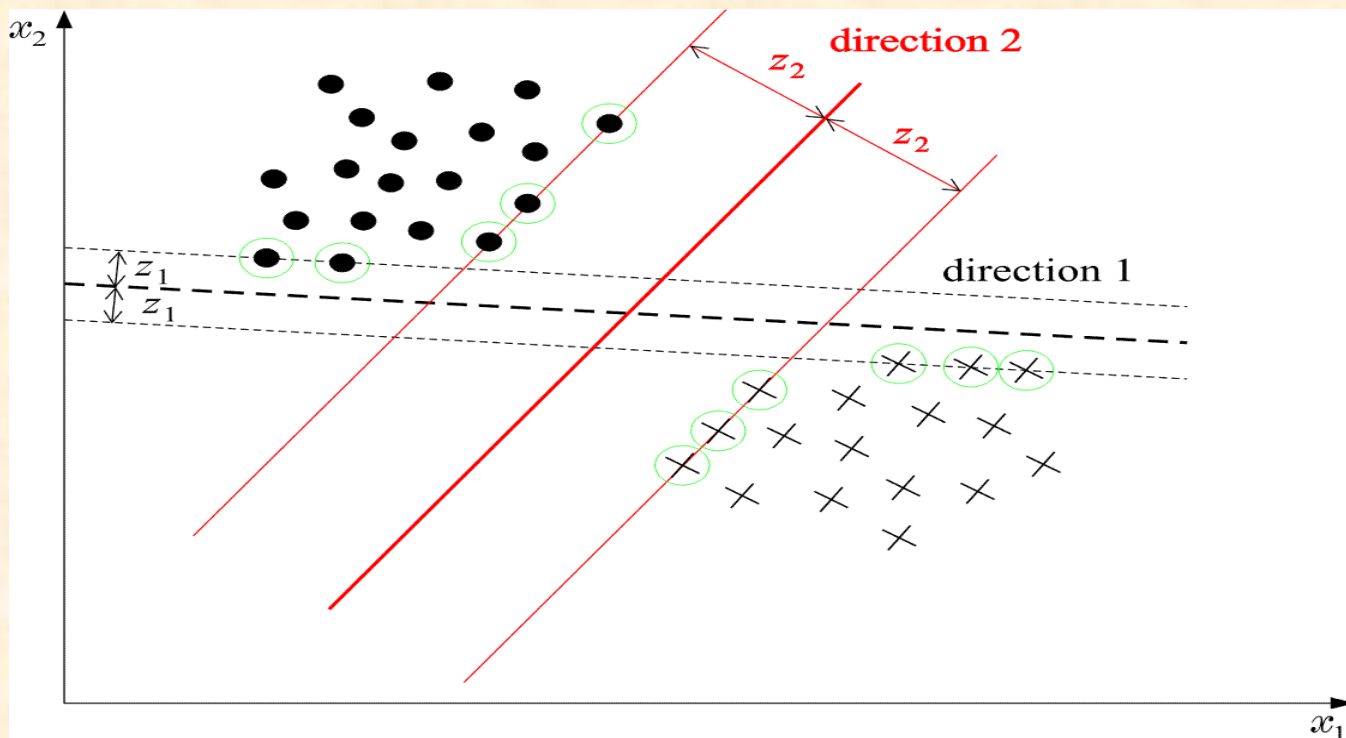
- The unknown parameters $\underline{w}_i, w_{i,0}, i = 1, 2, \dots, M-1$ are usually estimated by **maximum likelihood** arguments.
- Logistic discrimination is a useful tool, since it allows linear modeling and at the same time **ensures** posterior probabilities **to add to one**.

❖ Support Vector Machines

- The goal: Given two linearly separable classes, design the classifier

$$g(\underline{x}) = \underline{w}^T \underline{x} + w_0 = 0$$

that leaves the **maximum margin** from both classes



➤ **Margin:** Each hyperplane is characterized by

- Its direction in space, i.e., $\underline{w}$
- Its position in space, i.e., w_0
- For **EACH** direction, $\underline{w}$, choose the hyperplane that **leaves the SAME distance** from the **nearest** points from each class. The margin is twice this distance.

- The distance of a point $\hat{\underline{x}}$ from a hyperplane is given by

$$z_{\hat{\underline{x}}} = \frac{g(\hat{\underline{x}})}{\|\underline{w}\|}$$

- Scale, so that at the nearest points from each class the discriminant function is ± 1 :

- Thus the margin is given by $|g(\underline{x})| = 1 \left\{ g(\underline{x}) = +1 \text{ for } \omega_1 \text{ and } g(\underline{x}) = -1 \text{ for } \omega_2 \right\}$

- Also, the following is valid $\frac{1}{\|\underline{w}\|} + \frac{1}{\|\underline{w}\|} = \frac{2}{\|\underline{w}\|}$

$$\underline{w}^T \underline{x} + w_0 \geq 1 \quad \forall \underline{x} \in \omega_1$$

$$\underline{w}^T \underline{x} + w_0 \leq -1 \quad \forall \underline{x} \in \omega_2$$

➤ SVM (linear) classifier

$$g(\underline{x}) = \underline{w}^T \underline{x} + w_0$$

➤ Minimize

$$J(\underline{w}) = \frac{1}{2} \|\underline{w}\|^2$$

➤ Subject to

$$y_i (\underline{w}^T \underline{x}_i + w_0) \geq 1, \quad i = 1, 2, \dots, N$$

$$y_i = 1, \text{ for } \underline{x}_i \in \omega_1,$$

➤ The above is justified, for $\underline{w}_1$ and $\underline{w}_2$ minimizing

the margin is maximised

$$\|\underline{w}\|$$

$$\frac{2}{\|\underline{w}\|}$$

- The above is a **quadratic optimization task**, subject to a set of linear inequality constraints. The **Karush-Kuhh-Tucker** conditions state that the **minimizer** satisfies:

• (1)

$$\frac{\partial}{\partial \underline{w}} L(\underline{w}, w_0, \underline{\lambda}) = \underline{0}$$

• (2)

$$\frac{\partial}{\partial w_0} L(\underline{w}, w_0, \underline{\lambda}) = 0$$

• (3)

$$\lambda_i \geq 0, i = 1, 2, \dots, N$$

• (4)

• Where $\lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1] = 0, i = 1, 2, \dots, N$ is the **Lagrangian**

$$L(\bullet, \bullet, \bullet)$$

$$L(\underline{w}, w_0, \underline{\lambda}) \equiv \frac{1}{2} \underline{w}^T \underline{w} - \sum_{i=1}^N \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0)]$$

➤ The solution: from the above, it turns out that

- $$\underline{w} = \sum_{i=1}^N \lambda_i y_i \underline{x}_i$$

- $$\sum_{i=1}^N \lambda_i y_i = 0$$

➤ Remarks:

- The **Lagrange multipliers** can be either **zero** or **positive**.
Thus,

$$- \quad \underline{w} = \sum_{i=1}^{N_s} \lambda_i y_i \underline{x}_i$$

where λ_i are the Lagrange multipliers, corresponding to **positive** Lagrange multipliers

- From constraint (4) above, i.e.,

$$\lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1] = 0, \quad i = 1, 2, \dots, N$$

the vectors contributing to

satisfy

$$\underline{w}^T \underline{x}_i + w_0 = \pm 1$$

- These vectors are known as **SUPPORT VECTORS** and are the **closest vectors**, from each class, to the classifier.
- Once $\underline{w}$ is computed, w_0 is determined from conditions (4).
- The optimal hyperplane classifier of a support vector machine is **UNIQUE**.
- Although the solution is unique, the resulting Lagrange multipliers are **not** unique.

➤ Dual Problem Formulation

- The SVM formulation is a convex programming problem, with
 - Convex cost function
 - Convex region of feasible solutions
- Thus, its solution can be achieved by its dual problem, i.e.,

- maximize $L(\underline{w}, w_0, \underline{\lambda})$

- subject to

$$\underline{w} = \sum_{i=1}^N \lambda_i y_i \underline{x}_i$$

$$\sum_{i=1}^N \lambda_i y_i = 0$$

$$\underline{\lambda} \geq \underline{0}$$

- Combine the above to obtain

– maximize $\underline{\lambda} \left(\sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{ij} \lambda_i \lambda_j y_i y_j \underline{x}_i^T \underline{x}_j \right)$

– subject to

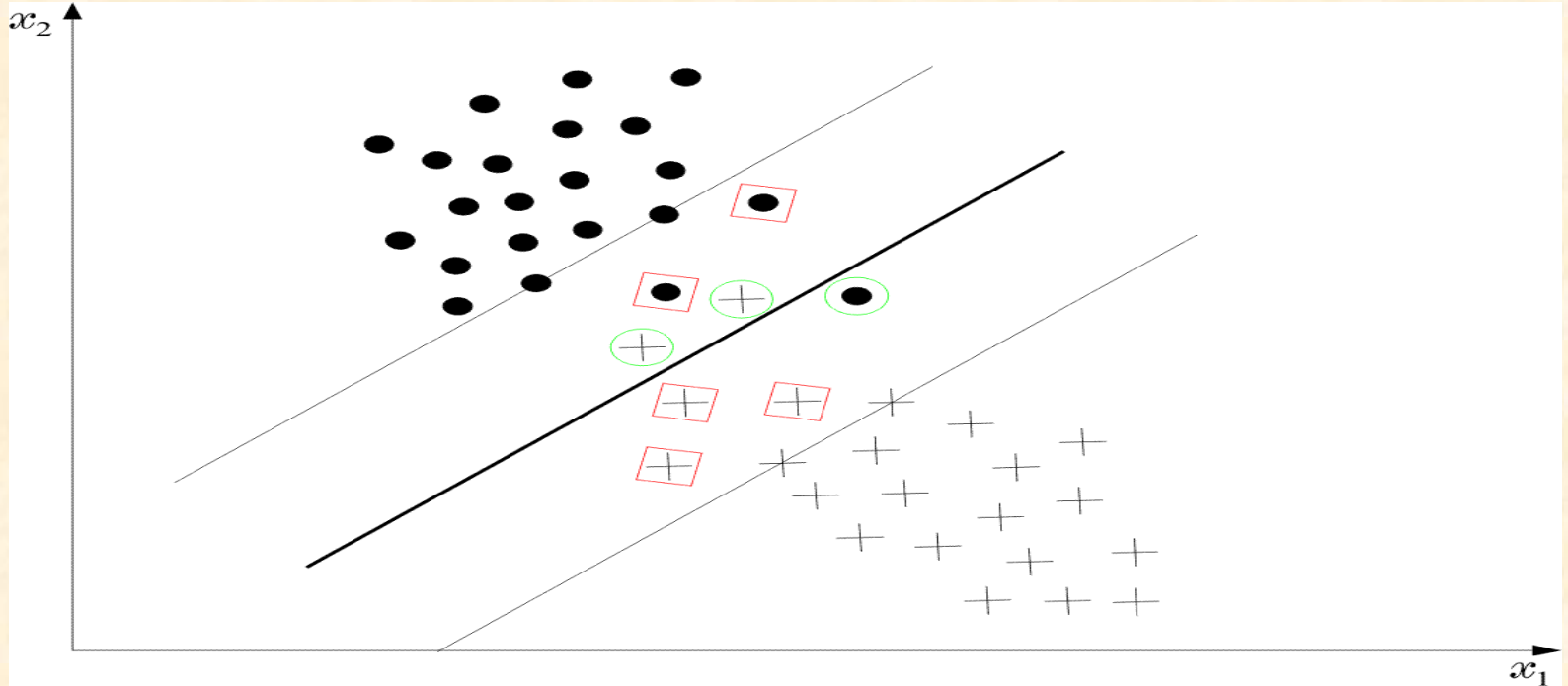
$$\sum_{i=1}^N \lambda_i y_i = 0$$

$$\underline{\lambda} \geq \underline{0}$$

➤ Remarks:

- Support vectors enter via **inner products**

➤ Non-Separable classes



In this case, there is no hyperplane such that

$$\underline{w}^T \underline{x} + w_0 (><) 1, \quad \forall \underline{x}$$

- Recall that the margin is defined as twice the distance between the following two hyperplanes

$$\underline{w}^T \underline{x} + w_0 = 1$$

and

$$\underline{w}^T \underline{x} + w_0 = -1$$

➤ The training vectors belong to one of three possible categories

1) Vectors **outside** the band which are **correctly** classified, i.e.,

$$y_i(\underline{w}^T \underline{x} + w_0) > 1$$

2) Vectors **inside** the band, and **correctly** classified, i.e.,

$$0 \leq y_i(\underline{w}^T \underline{x} + w_0) < 1$$

3) Vectors **misclassified**, i.e.,

$$y_i(\underline{w}^T \underline{x} + w_0) < 0$$

➤ All three cases above can be represented as

$$y_i(\underline{w}^T \underline{x} + w_0) \geq 1 - \xi_i$$

1) $\rightarrow \xi_i = 0$

2) $\rightarrow 0 < \xi_i \leq 1$

3) $\rightarrow 1 < \xi_i$

ξ_i are known as **slack variables**

➤ The goal of the optimization is now two-fold

- Maximize margin
- Minimize the number of patterns with

One way to achieve this goal is via the cost $\xi_i > 0$

where C is a constant and

$$J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2} \|\underline{w}\|^2 + C \sum_{i=1}^N I(\xi_i)$$

- $I(.)$ is not differentiable. In practice, we use an approximation

$$I(\xi_i) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases}$$

- Following a similar procedure as before we obtain

$$J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2} \|\underline{w}\|^2 + C \sum_{i=1}^N \xi_i$$

➤ KKT conditions

$$(1) \quad \underline{w} = \sum_{i=1}^N \lambda_i y_i \underline{x}_i$$

$$(2) \quad \sum_{i=1}^N \lambda_i y_i = 0$$

$$(3) \quad C - \mu_i - \lambda_i = 0, i = 1, 2, \dots, N$$

$$(4) \quad \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1 + \xi_i] = 0, \quad i = 1, 2, \dots, N$$

$$(5) \quad \mu_i \xi_i = 0, \quad i = 1, 2, \dots, N$$

$$(6) \quad \mu_i, \lambda_i \geq 0, \quad i = 1, 2, \dots, N$$

- The associated dual problem

Maximize $\lambda(\sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \underline{x}_i^T \underline{x}_j)$

subject to

$$0 \leq \lambda_i \leq C, \quad i = 1, 2, \dots, N$$

$$\sum_{i=1}^N \lambda_i y_i = 0$$

- Remarks: The only difference with the separable class case is the existence of constraints in the

➤ Training the SVM

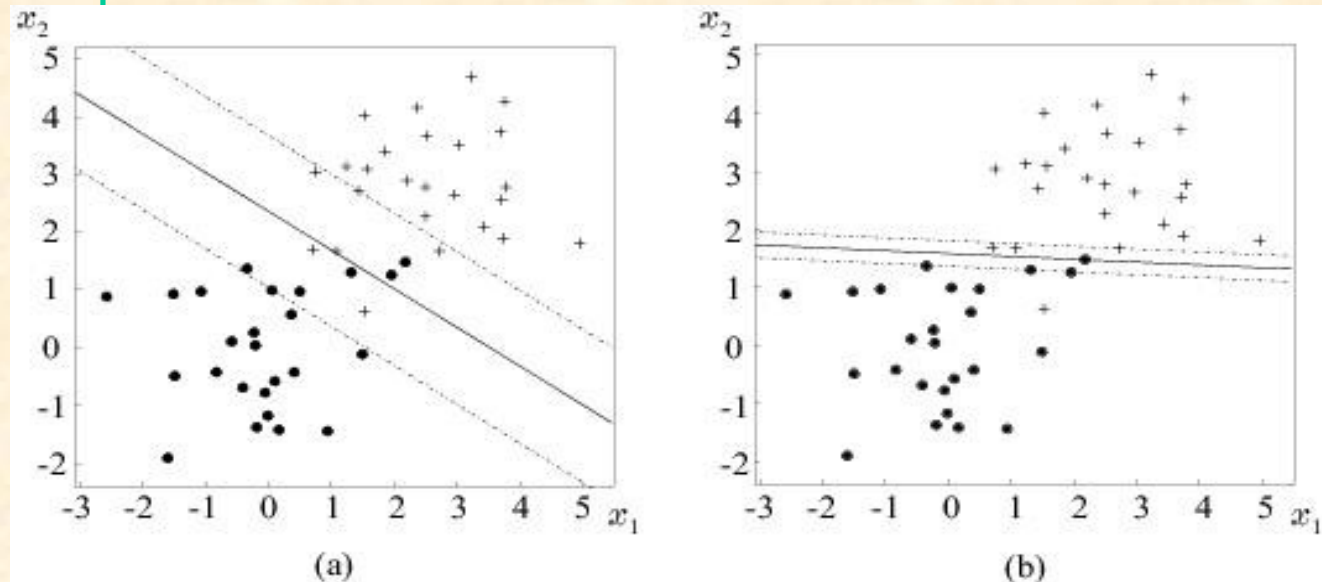
A major problem is the high computational cost. To this end, decomposition techniques are used. The rationale behind them consists of the following:

- Start with an arbitrary data subset (**working set**) that can fit in the memory. Perform optimization, via a general purpose optimizer.
- Resulting support vectors **remain** in the working set, while others are replaced by new ones (outside the set) that violate severely the KKT conditions.
- Repeat the procedure.
- The above procedure guarantees that the cost function decreases.
- Platt's **SMO algorithm** chooses a working set of two samples, thus **analytic** optimization solution can be obtained.

➤ Multi-class generalization

Although theoretical generalizations exist, the most popular in practice is to look at the problem as M two-class problems (one against all).

➤ Example:



➤ Observe the effect of different values of C in the case of non-separable classes.