# CHAPTER 3 – LINEAR CLASSIFIERS

❖ The Problem: Consider a two class task with $\omega_1$, $\omega_2$

➢ $$g(\underline{x}) = \underline{w}^T \underline{x} + w_0 = 0 = w_1 x_1 + w_2 x_2 + \ldots + w_l x_l + w_0$$

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

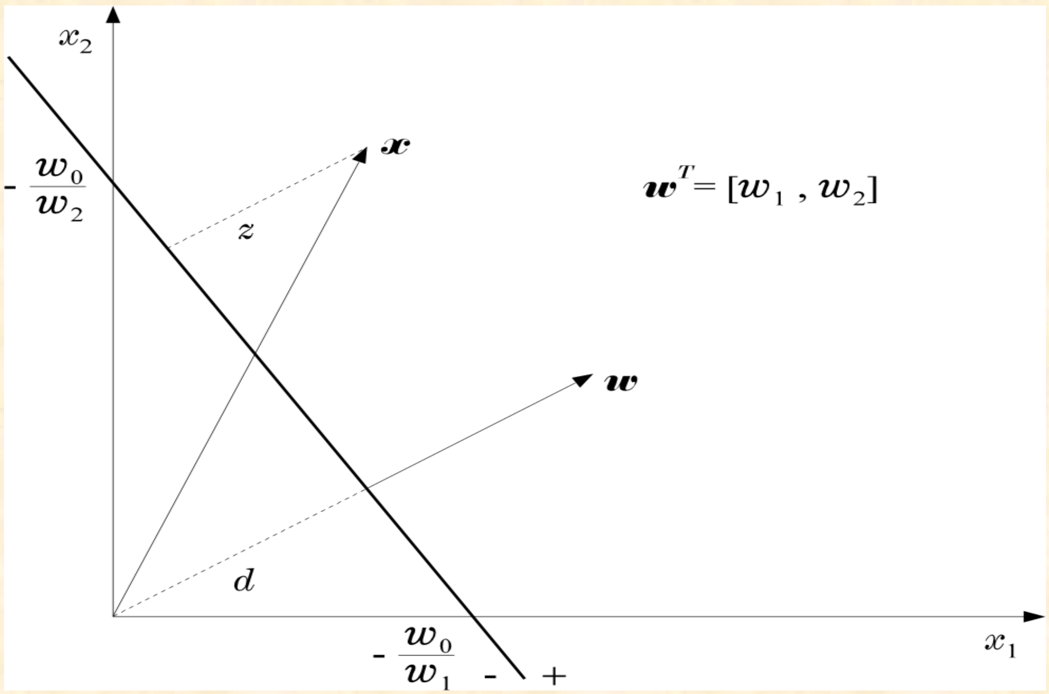
$$0 = \underline{w}^T \underline{x}_1 + w_0 = \underline{w}^T \underline{x}_2 + w_0 \Rightarrow$$

$$\underline{w}^T (\underline{x}_1 - \underline{x}_2) = 0 \quad \forall \underline{x}_1, \underline{x}_2$$

- 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.,

$$\exists \ \underline{w}^*: \ \underline{w}^{*T} \underline{x} > 0 \ \ \forall \underline{x} \in \omega_1$$
$$\underline{w}^{*T} \underline{x} < 0 \ \ \forall \underline{x} \in \omega_2$$

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

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

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

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

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

  - 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

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

- Where $Y$ is the subset of the vectors wrongly classified by $\underline{w}$. When $Y$=O (empty set) a solution is achieved and

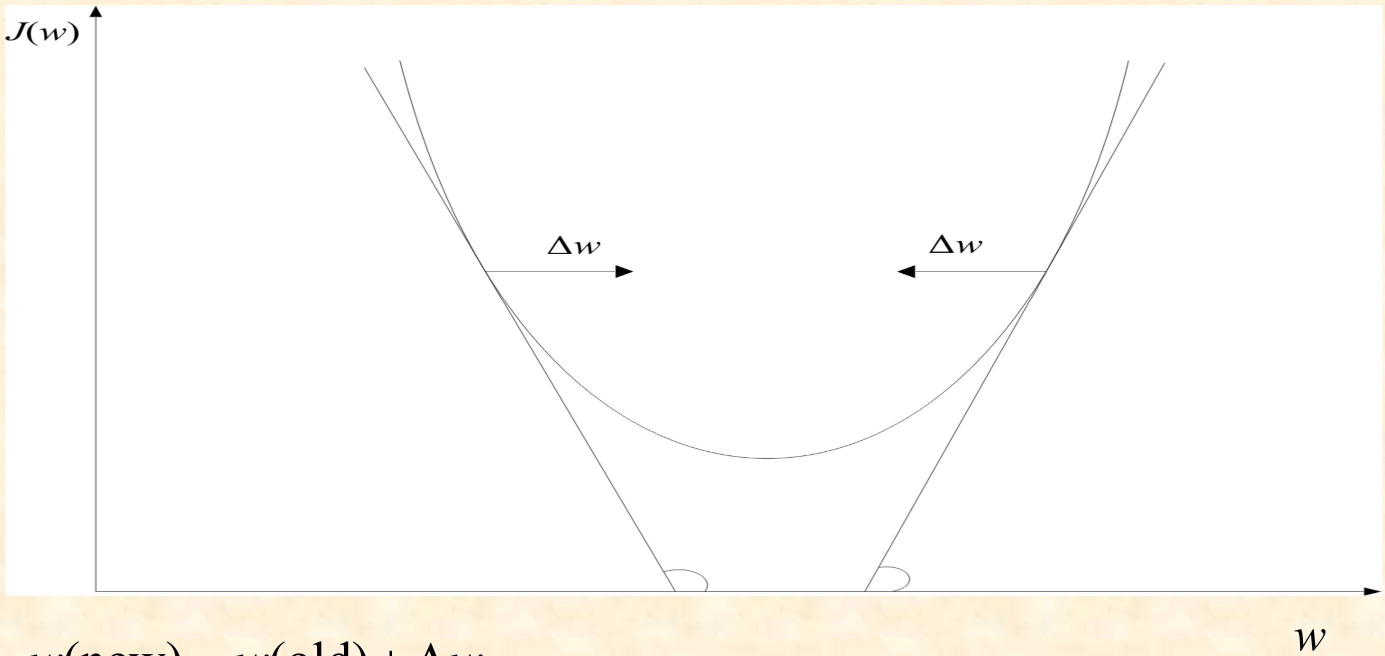
- $J(\underline{w}) = 0$

- $\delta_x = -1 \text{ if } \underline{x} \in Y \text{ and } \underline{x} \in \omega_1$
  $\delta_x = +1 \text{ if } \underline{x} \in Y \text{ and } \underline{x} \in \omega_2$

- $J(\underline{w}) \geq 0$

- $J(\underline{w})$ 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

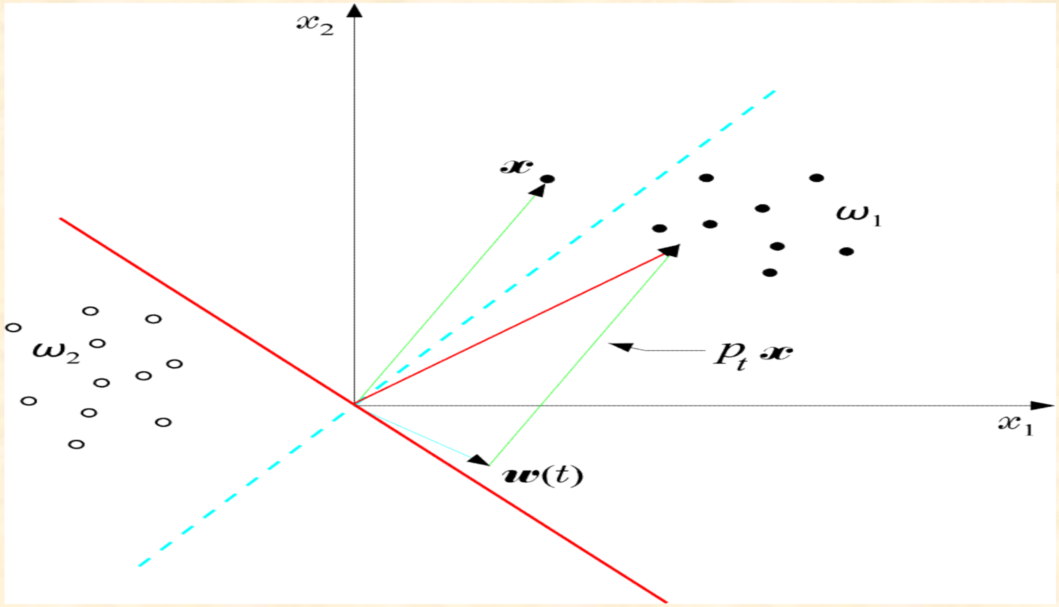
$$\frac{\partial J(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}} \left( \sum_{\underline{x} \in Y} \delta_x \underline{w}^T \underline{x} \right) = \sum_{\underline{x} \in Y} \delta_x \underline{x}$$

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



This is the celebrated Perceptron Algorithm.

- An example:




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

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

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

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

❖ A useful variant of the perceptron algorithm

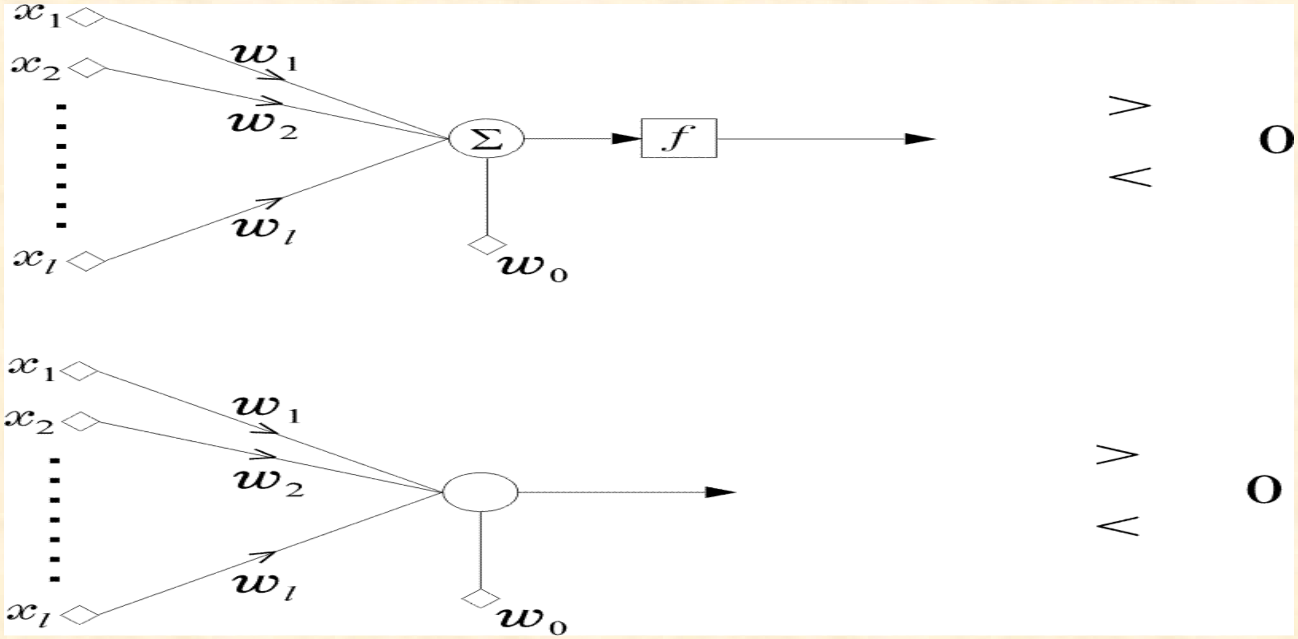
$$\underline{w}(t+1) = \underline{w}(t) + \rho \underline{x}_{(t)}, \quad \begin{array}{c} \underline{w}^T(t)\underline{x}_{(t)} \le 0 \\ \underline{x}_{(t)} \in \omega_1 \end{array}$$

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

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

➢ It is a reward and punishment type of algorithm.

## ❖ The perceptron




$w_i{}'s$ synapses or synaptic weights

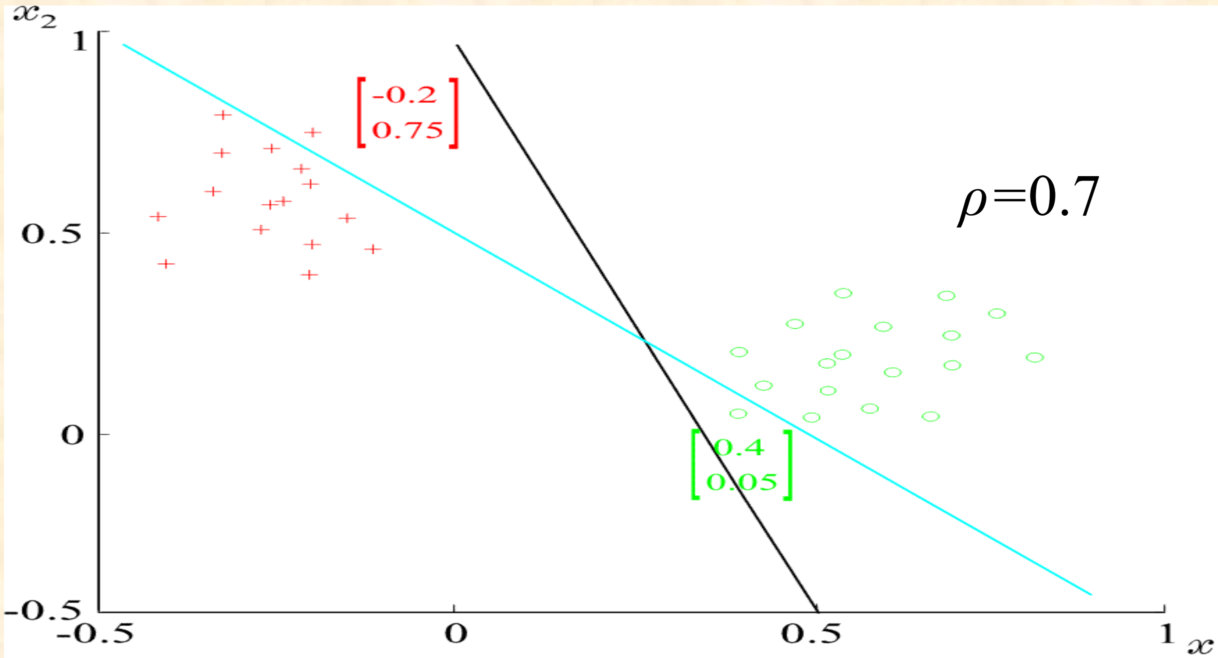
$w_0$ threshold

- 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 $\pm 1$

- If classes are NOT linearly separable, we shall compute the weights, $w_1, w_2, \ldots, w_0$, so that the difference between
  - The actual output of the classifier, $\underline{w}^T \underline{x}$, and
  - The desired outputs, e.g.,

$$+1 \text{ if } \underline{x} \in \omega_1$$
$$-1 \text{ if } \underline{x} \in \omega_2$$

  to be SMALL.

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

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

➢ Minimizing

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

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

$$= 2E[\underline{x}(y - \underline{x}^T \underline{w})] \Rightarrow$$

$$E[\underline{x}\underline{x}^T]\underline{w} = E[\underline{x}y] \Rightarrow$$

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

where $R_x$ is the autocorrelation matrix

$$R_x \equiv E[\underline{x}\underline{x}^T] = \begin{bmatrix} E[x_1 x_1] & E[x_1 x_2]\ldots & E[x_1 x_l] \\ \ldots\ldots\ldots. & \ldots\ldots\ldots\ldots & \ldots\ldots\ldots. \\ E[x_l x_1] & E[x_l x_2]\ldots & E[x_l x_l] \end{bmatrix}$$

and $E[\underline{x}y] = \begin{bmatrix} E[x_1 y] \\ \ldots \\ E[x_l y] \end{bmatrix}$ the crosscorrelation vector.

## ➢ 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 $y_i$:

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

- Let

$$\underline{y} = \left[y_1, y_2, \ldots, y_M\right]^T$$

- And the matrix

$$W = \left[\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_M\right]$$

- 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.

❖ SMALL in the sum of error squares sense means

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

$(y_i, \underline{x}_i)$: training pairs that is, the input $\underline{x}_i$ and its corresponding class label $y_i$ (±1).

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

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

## ❖ Pseudoinverse Matrix

- Define

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

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

- $X^T = [\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N]$ (an $lxN$ matrix)

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

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

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

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

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

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

$$X^{\neq} \equiv (X^T X)^{-1} X^T$$ Pseudoinverse of $X$

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

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

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

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

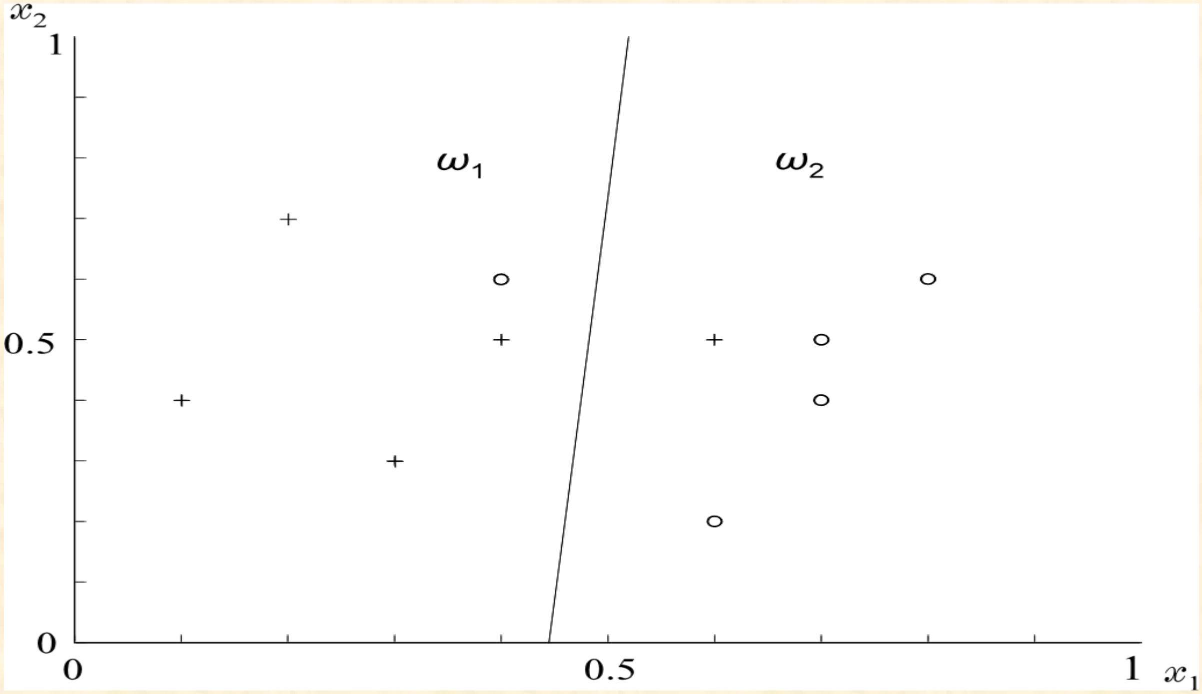
$$X\underline{w}=\underline{y}: \quad \begin{matrix} \underline{x}_1^T\underline{w}=y_1 \\ \underline{x}_2^T\underline{w}=y_2 \\ \ldots. \\ \underline{x}_N^T\underline{w}=y_N \end{matrix} \qquad N \text{ equations} > l \text{ unknowns}$$

- The "solution" $\underline{w}=X^{\neq}\underline{y}$ corresponds to the minimum sum of squares solution.

- Example:

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

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




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

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

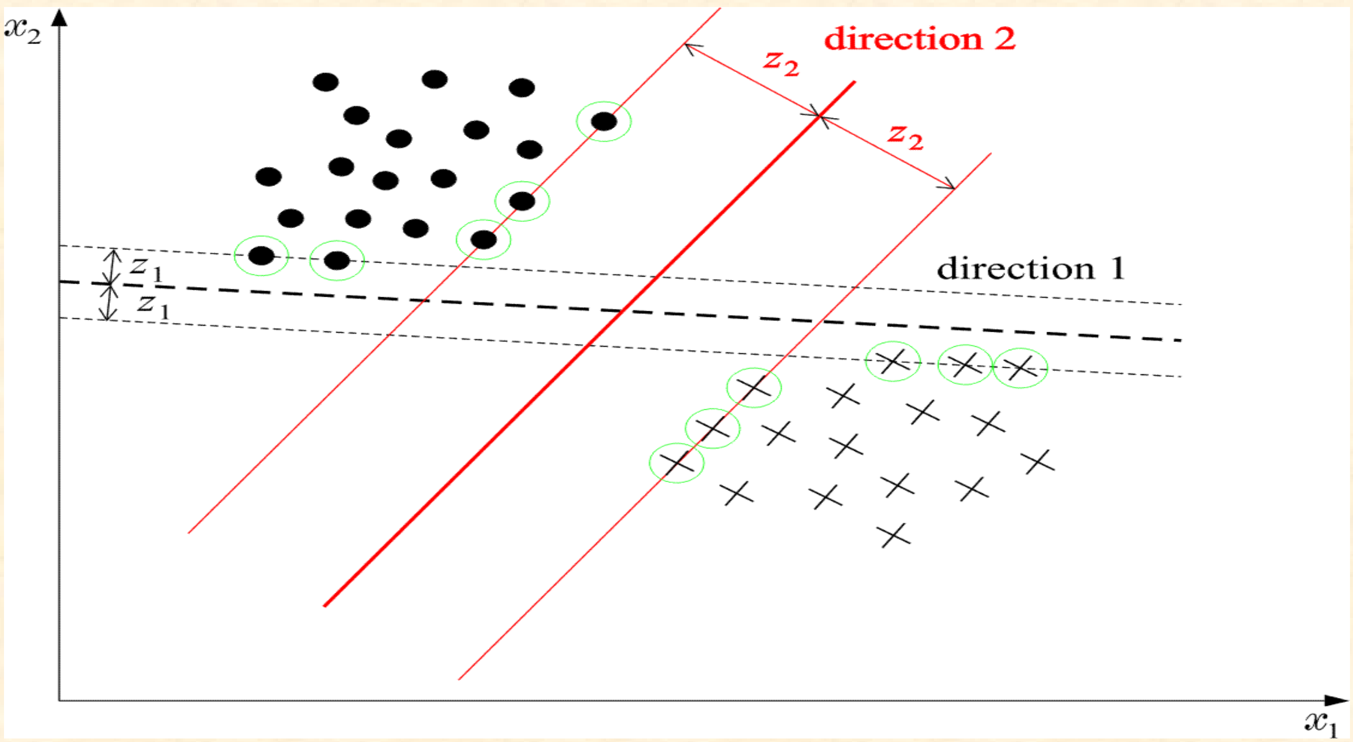
$$\underline{w} = (X^T X)^{-1} X^T \underline{y} = \begin{bmatrix} -3.13 \\ 0.24 \\ 1.34 \end{bmatrix}$$

## ❖ 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:

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

- Scale, $\underline{w}, \underline{w}_0$, so that at the nearest points, from each class, the discriminant function is ±1:

$$|g(\underline{x})| = 1 \ \{g(\underline{\mathrm{x}}) = +1 \text{ for } \omega_1 \text{ and } g(\underline{x}) = -1 \text{ for } \omega_2\}$$

- Thus the margin is given by:

$$\frac{1}{\|\underline{w}\|} + \frac{1}{\|\underline{w}\|} = \frac{2}{\|\underline{w}\|}$$

- Also, the following is valid

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

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

- 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

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

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

$$y_i = -1, \text{ for } \underline{x}_i \in \omega_2$$

- The above is justified since by minimizing $\|\underline{w}\|$

  the margin $\frac{2}{\|w\|}$ is maximised.

- 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) $\frac{\partial}{\partial \underline{w}} \mathrm{L}(\underline{w}, w_0, \underline{\lambda}) = \underline{0}$
  - (2) $\frac{\partial}{\partial w_0} L(\underline{w}, w_0, \underline{\lambda}) = 0$
  - (3) $\lambda_i \geq 0, i = 1,2,\ldots, N$
  - (4) $\lambda_i \left[ y_i (\underline{w}^T \underline{x}_i + w_0) - 1 \right] = 0, i = 1,2,\ldots, N$
  - Where $L(\bullet,\bullet,\bullet)$ is the Lagrangian

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

- 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,

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

  where $N_s \le N_0$, corresponding to positive Lagrange multipliers.

  – From constraint (4) above, i.e.,

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

  the vectors contributing to $\underline{w}$ 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 $\underset{\underline{\lambda}}{} \quad L(\underline{w}, w_0, \underline{\lambda})$
    - subject to

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

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

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

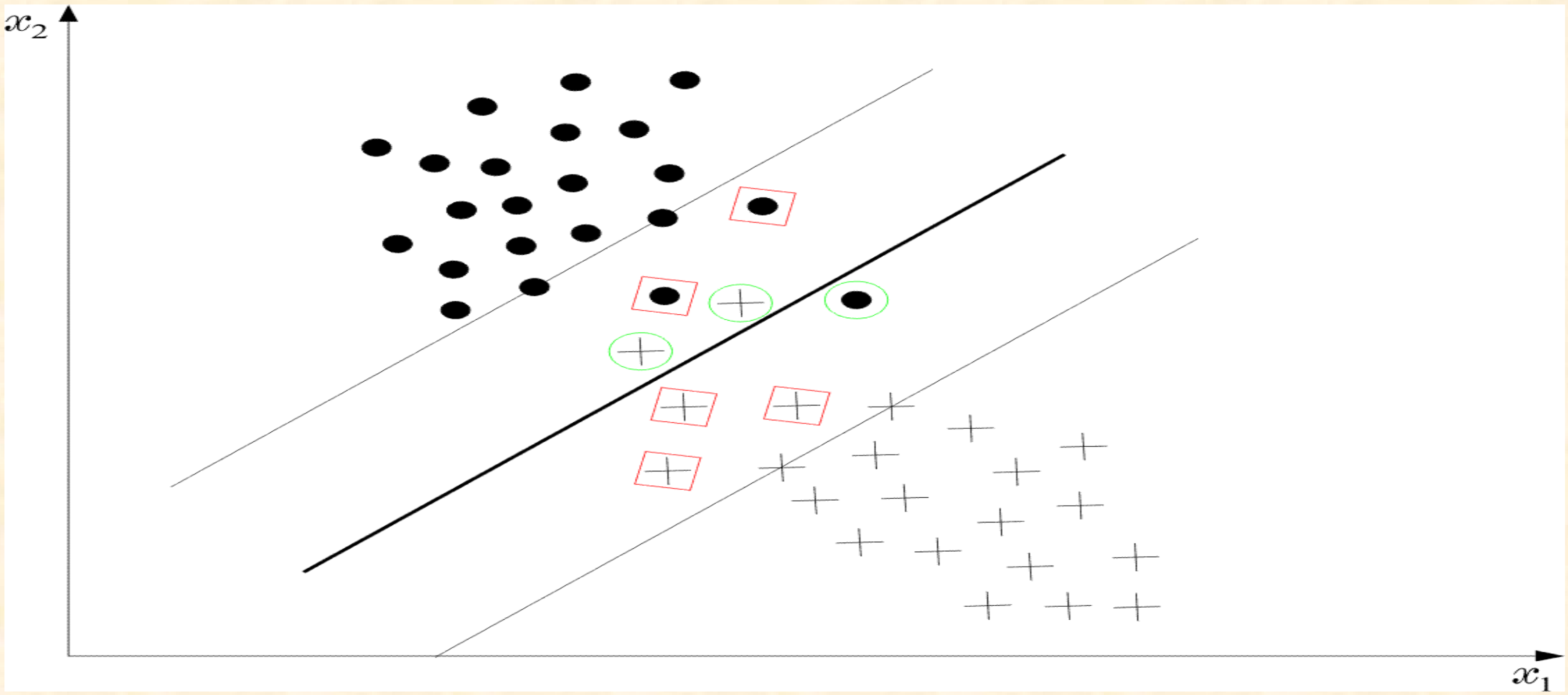
- Combine the above to obtain:
  - maximize $\underset{\underline{\lambda}}{}$ $$\left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{ij} \lambda_i \lambda_j y_i y_j \underline{x}_i^T \underline{x}_j\right)$$
  - subject to

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

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

- Remarks:
  - Support vectors enter via inner products.

- Non-Separable classes



In this case, there is no hyperplane such that:

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

- 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) $\rightarrow 1 < \xi_i$

$\xi_i$ are known as slack variables.

- The goal of the optimization is now two-fold:
  - Maximize margin
  - Minimize the number of patterns with $\xi_i > 0$ .

    One way to achieve this goal is via the cost

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

where $C$ is a constant and

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

  - $I(.)$ is not differentiable. In practice, we use an approximation. A popular choice is:
  - $J(\underline{w}, w_0, \underline{\xi}) = \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{N} \xi_i$
  - Following a similar procedure as before we obtain:

- KKT conditions

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

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

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

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

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

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

- The associated dual problem

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

subject to

$$0 \le \lambda_i \le C,\ i = 1,2,\ldots,N$$

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

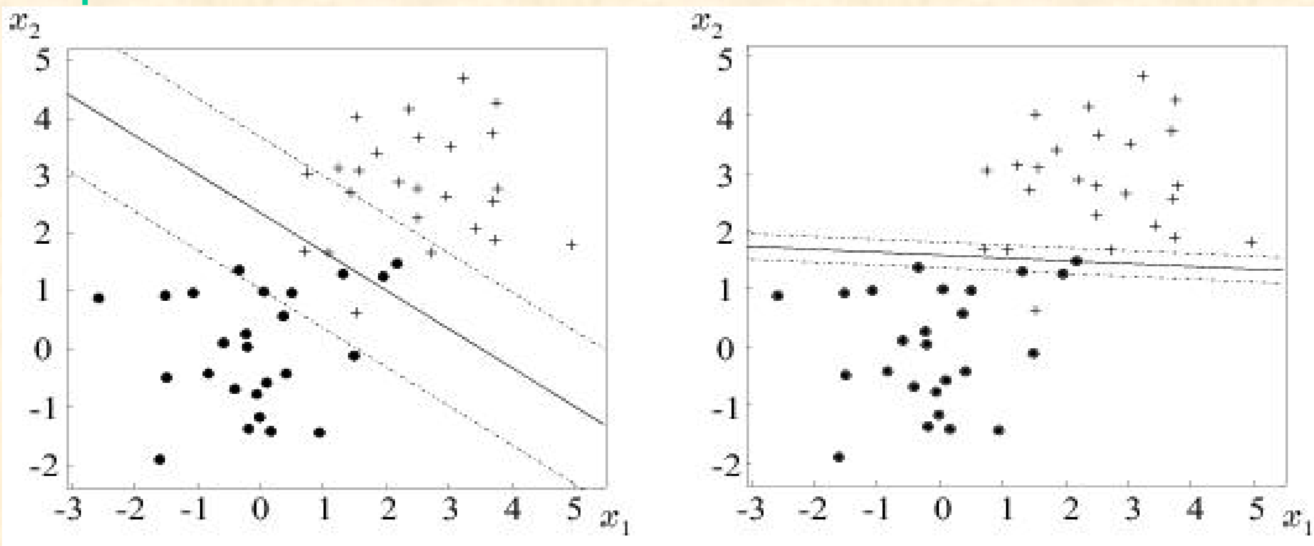
- Remarks: The only difference with the separable class case is the existence of $C$ in the constraints.

- 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:



(a) (b)

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