

Ch3: LINEAR CLASSIFIERS

(Linear Discriminant Functions)

❖ Likelihood vs. Discriminant-based Classification

➤ **Likelihood-based:** Assume a model for $p(\mathbf{x}|\omega_i)$, use Bayes' rule to calculate $P(\omega_i|\mathbf{x})$

$$g_i(\mathbf{x}) = \log P(\omega_i|\mathbf{x})$$

➤ **Discriminant-based:** Assume a model for $g_i(\mathbf{x}|\Phi_i)$; no density estimation.

➤ Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries.

- ❖ In this chapter, we will focus on the design of linear classifiers, regardless of the underlying distributions describing the training data.
- ❖ The Problem: Consider a two class task with ω_1, ω_2

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

$\underline{w} = [w_1, w_2, \dots, w_l]^T$ is known as the *weight vector* and w_0 as the *threshold*.

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

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

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

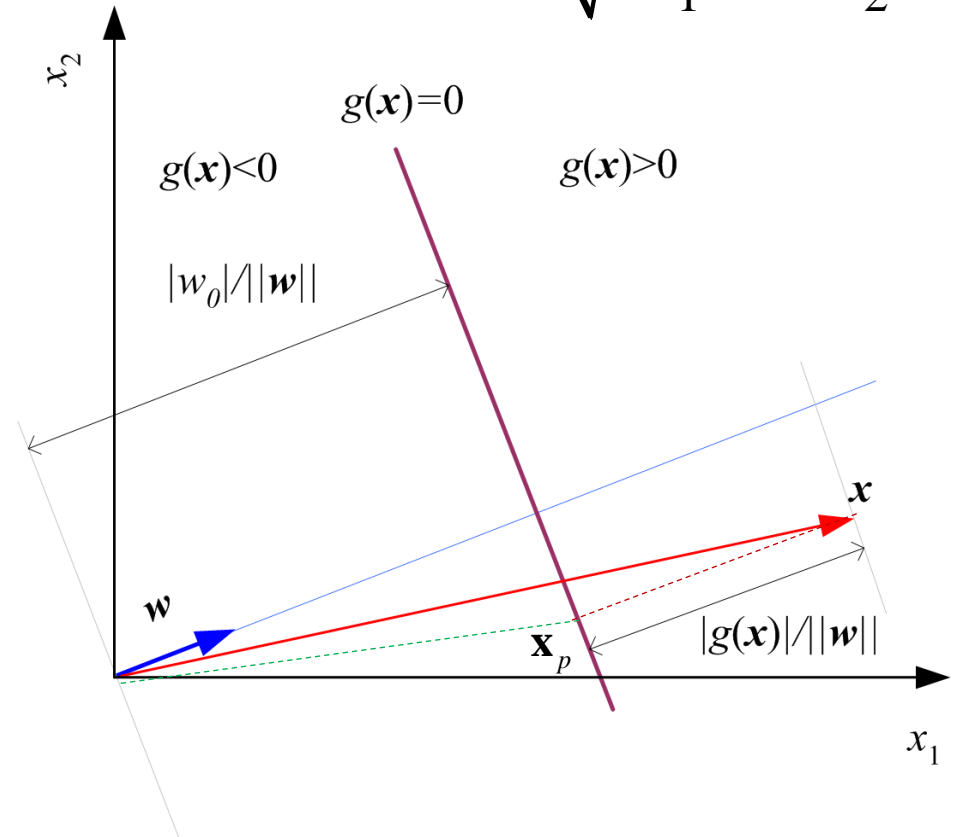
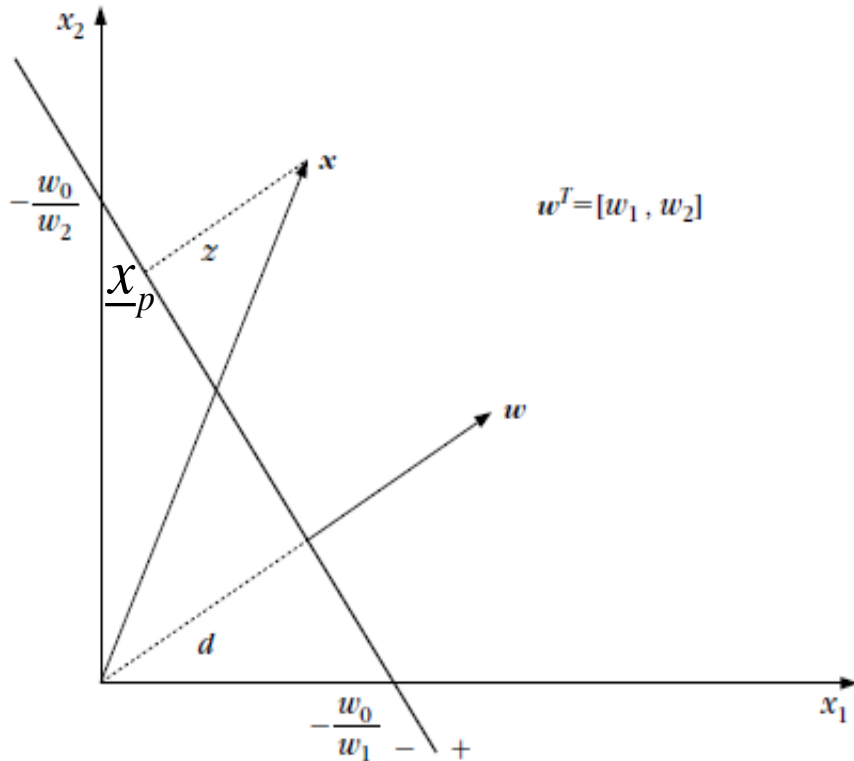
➤ Hence:

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

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

$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}},$$

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



$$\mathbf{x} = \mathbf{x}_p + \frac{z \cdot \mathbf{w}}{\|\mathbf{w}\|} \quad (\text{since } \mathbf{w} \text{ is colinear with } \mathbf{x} - \mathbf{x}_p \text{ and } \frac{\mathbf{w}}{\|\mathbf{w}\|} = 1)$$

$$\text{since } g(\mathbf{x}_p) = 0 \text{ and } \mathbf{w}^t \cdot \mathbf{w} = \|\mathbf{w}\|^2 \Rightarrow g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0 =$$

$$= \mathbf{w}^t \left(\mathbf{x}_p + \frac{z \cdot \mathbf{w}}{\|\mathbf{w}\|} \right) + w_0 = \mathbf{w}^t \mathbf{x}_p + w_0 + \frac{z \mathbf{w}^t \cdot \mathbf{w}}{\|\mathbf{w}\|} = z \|\mathbf{w}\|$$

$$\text{therefore } z = \frac{g(\mathbf{x})}{\|\mathbf{w}\|} \rightarrow \text{in particular } d(0, H) = \frac{g(\mathbf{0})}{\|\mathbf{w}\|} = \frac{w_0}{\|\mathbf{w}\|}$$

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 \mathbf{w} and the location of the surface is determined by the bias

❖ The Perceptron Algorithm

- Assume linearly separable classes, i.e.,

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

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

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

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


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

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

- 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 $Y=(\text{empty set})$ a solution is achieved and
 - $J(\underline{w}) = 0$
 - $\delta_x = -1$ if $\underline{x} \in Y$ and $\underline{x} \in \omega_1$
 - $\delta_x = +1$ if $\underline{x} \in Y$ and $\underline{x} \in \omega_2$
-  $J(\underline{w}) \geq 0$

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

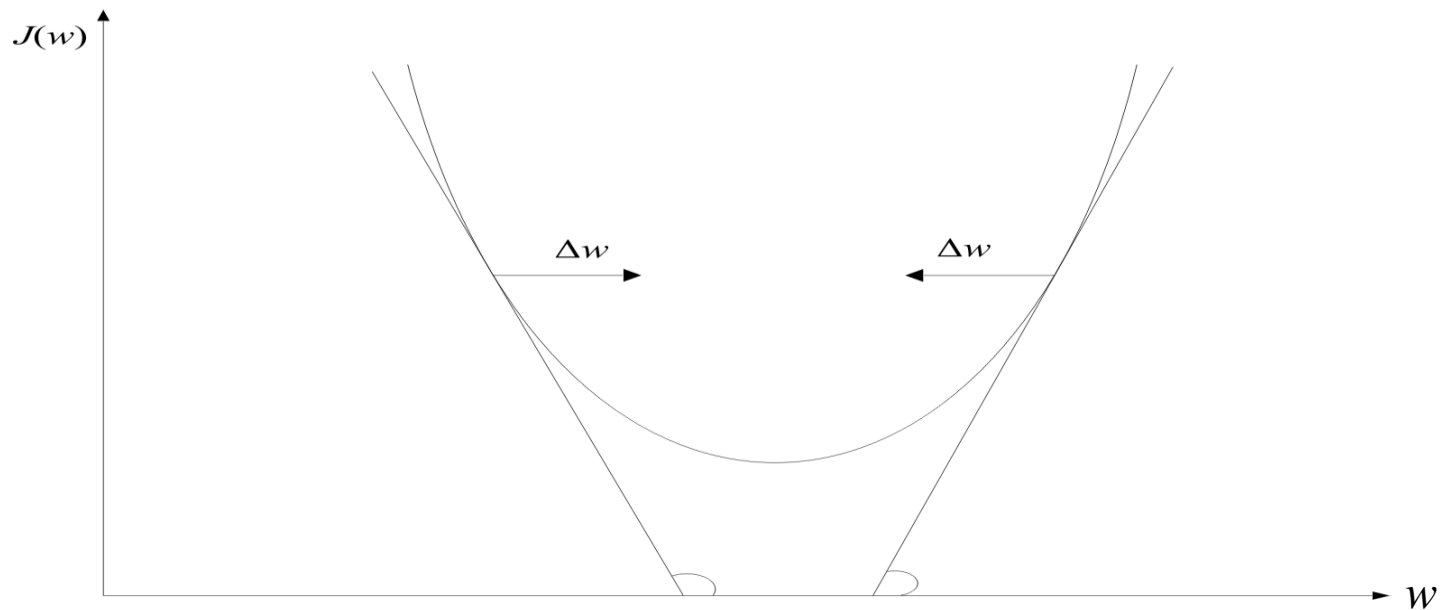


➤ The Algorithm

- The philosophy of the gradient descent is adopted.

$$\underline{w}(t+1) = \underline{w}(t) - \rho_t \left. \frac{\partial J(\underline{w})}{\partial \underline{w}} \right|_{\underline{w} = \underline{w}(t)}$$

ρ_t is a sequence of positive real numbers



$$\underline{w}(\text{new}) = \underline{w}(\text{old}) + \Delta \underline{w} \quad , \quad \Delta \underline{w} = -\rho \left. \frac{\partial J(\underline{w})}{\partial \underline{w}} \right|_{\underline{w}=\underline{w}(\text{old})}$$

- Wherever valid

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

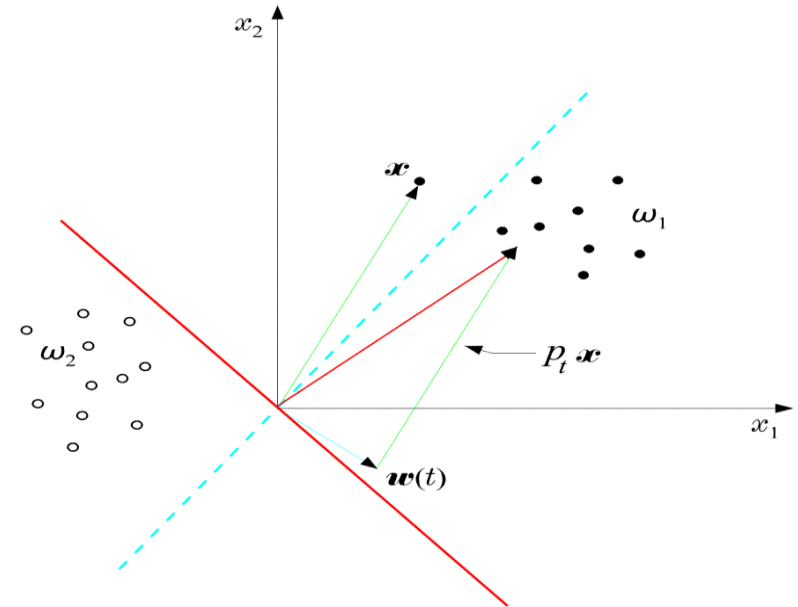
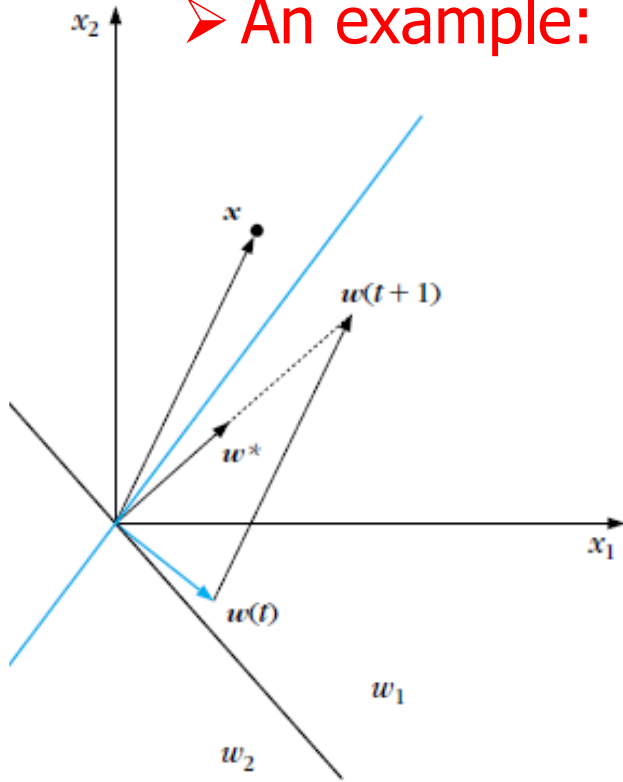
- $$\underline{w}(t+1) = \underline{w}(t) - \rho_t \sum_{\underline{x} \in Y} \delta_x \underline{x}$$

This is the celebrated **Perceptron Algorithm**

The Perceptron Algorithm

- Choose $\mathbf{w}(0)$ randomly
- Choose ρ_0
- $t = 0$
- Repeat
 - $Y = \emptyset$
 - For $i = 1$ to N
 - If $\delta_{x_i} \mathbf{w}(t)^T \mathbf{x}_i \geq 0$ then $Y = Y \cup \{\mathbf{x}_i\}$
 - End {For}
 - $\mathbf{w}(t + 1) = \mathbf{w}(t) - \rho_t \sum_{\mathbf{x} \in Y} \delta_{\mathbf{x}} \mathbf{x}$
 - Adjust ρ_t
 - $t = t + 1$
- Until $Y = \emptyset$

➤ An example:



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

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

$$\lim_{t \rightarrow \infty} \sum_{k=0}^t \rho_k \rightarrow \infty, \quad \lim_{t \rightarrow \infty} \sum_{k=0}^t \rho_k^2 < +\infty \quad \text{e.g.,: } \rho_t = \frac{c}{t}$$

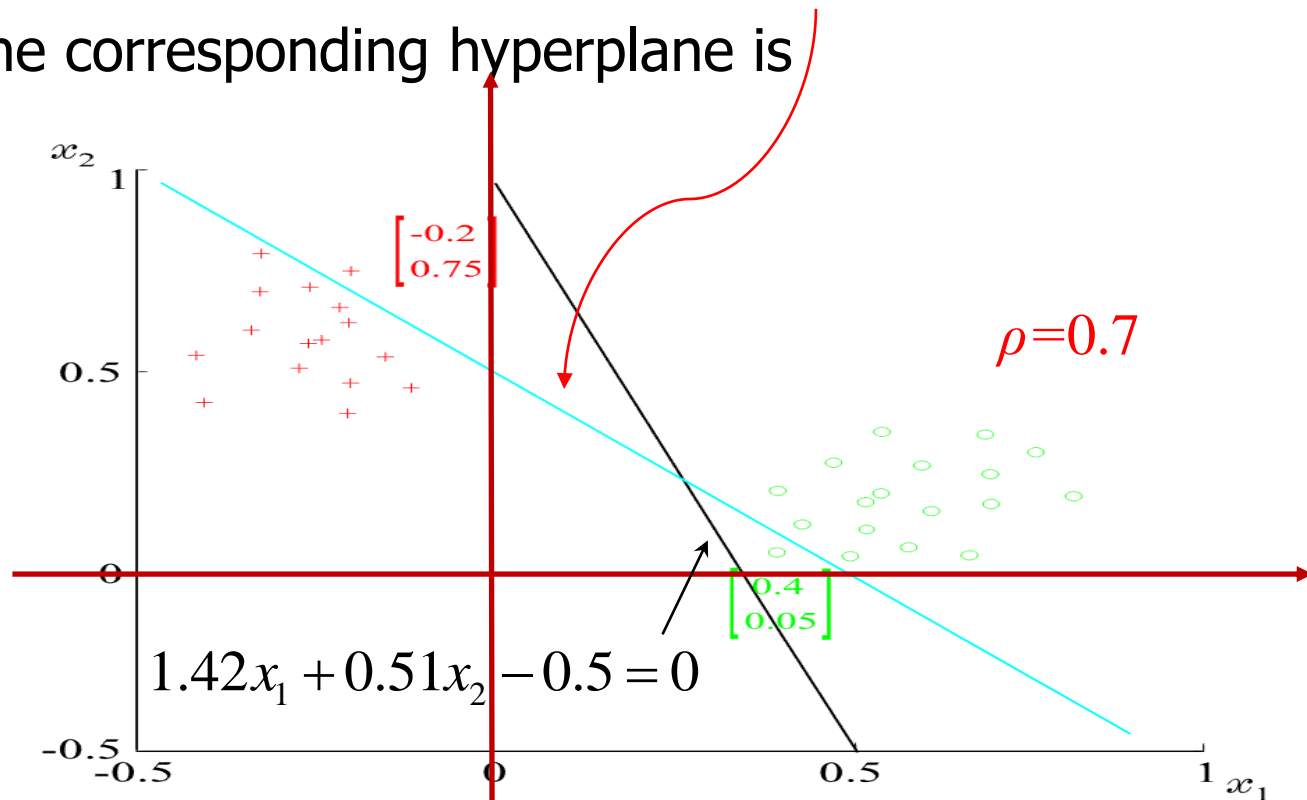
The algorithm also converges for constant $\rho_t = \rho$ provided ρ is properly bounded.

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

$$w_1 = 1, w_2 = 1, w_0 = -0.5 \Rightarrow \underline{w}(t) = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix}$$

$$x_1 + x_2 - 0.5 = 0$$

The corresponding hyperplane is



The next weight Vector will be:

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

❖ A useful variant of the perceptron algorithm

$$\underline{w}(t+1) = \underline{w}(t) + \rho \underline{x}(t), \quad \text{if } \underline{x}(t) \in \omega_1 \text{ and } \underline{w}^T(t) \underline{x}(t) \leq 0$$

$$\underline{w}(t+1) = \underline{w}(t) - \rho \underline{x}(t), \quad \text{if } \underline{x}(t) \in \omega_2 \text{ and } \underline{w}^T(t) \underline{x}(t) \geq 0$$

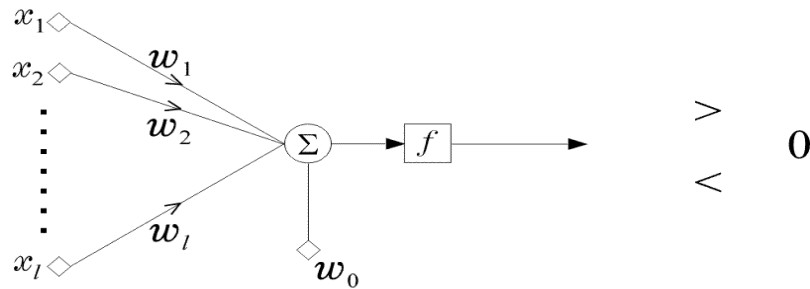
$$\underline{w}(t+1) = \underline{w}(t) \quad \text{otherwise}$$

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

❖ Correct classification → the **reward** is no action.

❖ Incorrect classification → the **punishment** is the cost of correction.

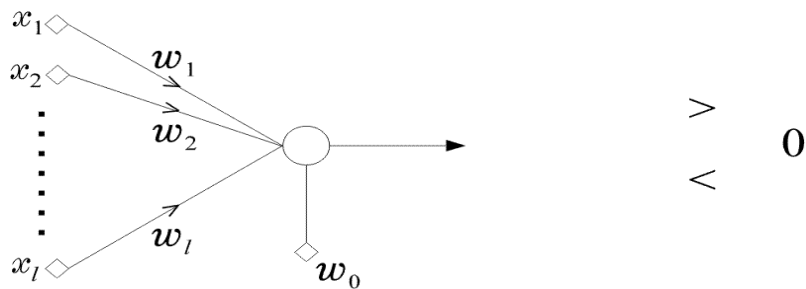
❖ The perceptron



$>$
 $<$

if $\underline{w}^T \underline{x} + w_0 > 0$ assign \underline{x} to ω_1

if $\underline{w}^T \underline{x} + w_0 < 0$ assign \underline{x} to ω_2



$>$
 $<$

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**.

Least Squares Methods

- If classes are linearly separable, the perceptron output (± 1) were correct for all the training feature vectors.
- If classes are NOT linearly separable, we shall compute the weights, 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

- $J(\underline{w}) \equiv E[(y - \underline{w}^T \underline{x})^2]$ is minimum
- $\hat{\underline{w}} = \arg \min_{\underline{w}} J(\underline{w})$
- $y(\underline{x}) = y = \pm 1$ is the corresponding desired responses

➤ It can be shown that $J(\underline{w})$ is equal to:

$$J(\mathbf{w}) = P(\omega_1) \int (1 - \mathbf{x}^T \mathbf{w})^2 p(\mathbf{x} | \omega_1) d\mathbf{x} + P(\omega_2) \int (1 + \mathbf{x}^T \mathbf{w})^2 p(\mathbf{x} | \omega_2) d\mathbf{x}$$

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

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

Orthogonality condition.

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

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

A linear set of equations

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

where R_x is the autocorrelation matrix

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

and $E[\underline{x}y] = \begin{bmatrix} E[x_1y] \\ \dots \\ E[x_ly] \end{bmatrix}$ the cross-correlation 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 desired responses (class labels) y_i as:

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

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

- And the matrix $W = [\underline{w}_1, \underline{w}_2, \dots, \underline{w}_M]$

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

Sum of Error Squares Estimation

- ❖ A criterion closely related to the MSE is the **sum of error squares** or simply the **least squares (LS)** criterion.
- ❖ **SMALL** in the **sum of error squares** sense means

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

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

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

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

❖ Pseudoinverse Matrix

➤ Define

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

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

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

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

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

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

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

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

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

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

➤ 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:

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

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

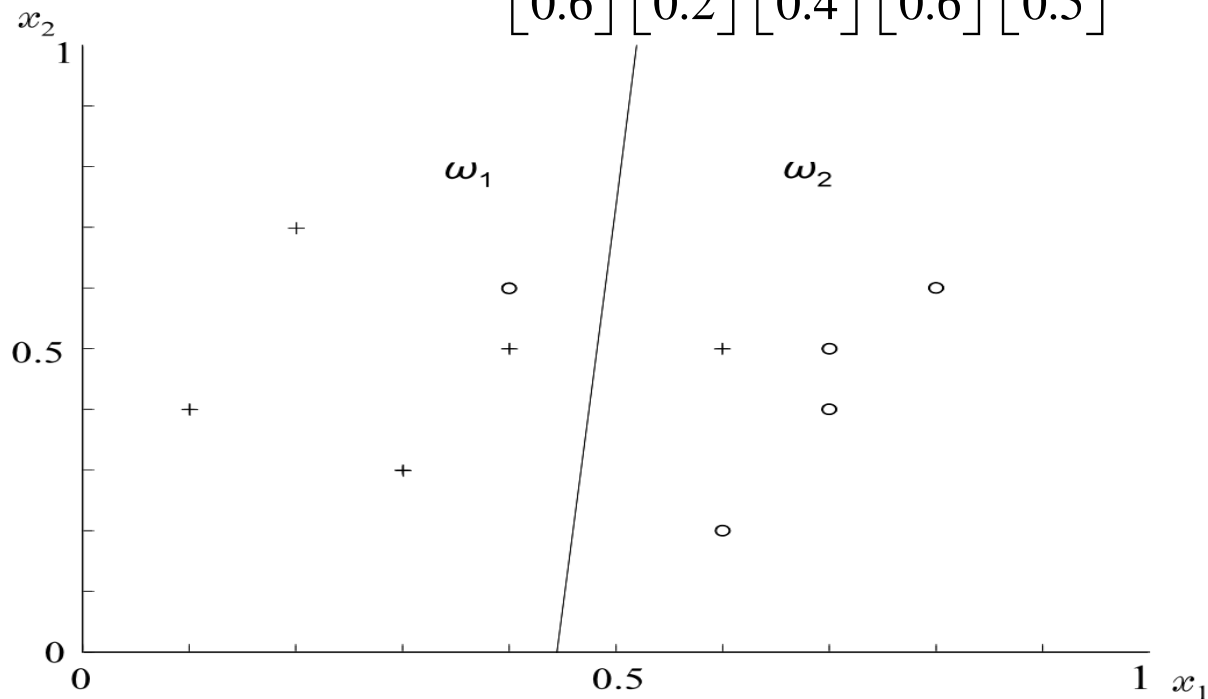
- Assume $N < l$ the undetermined problem

$$\underline{\hat{w}} = X^T (X X^T)^{-1} \underline{y} = X^+ \underline{y} \quad \text{minimum norm solution}$$

➤ Example:

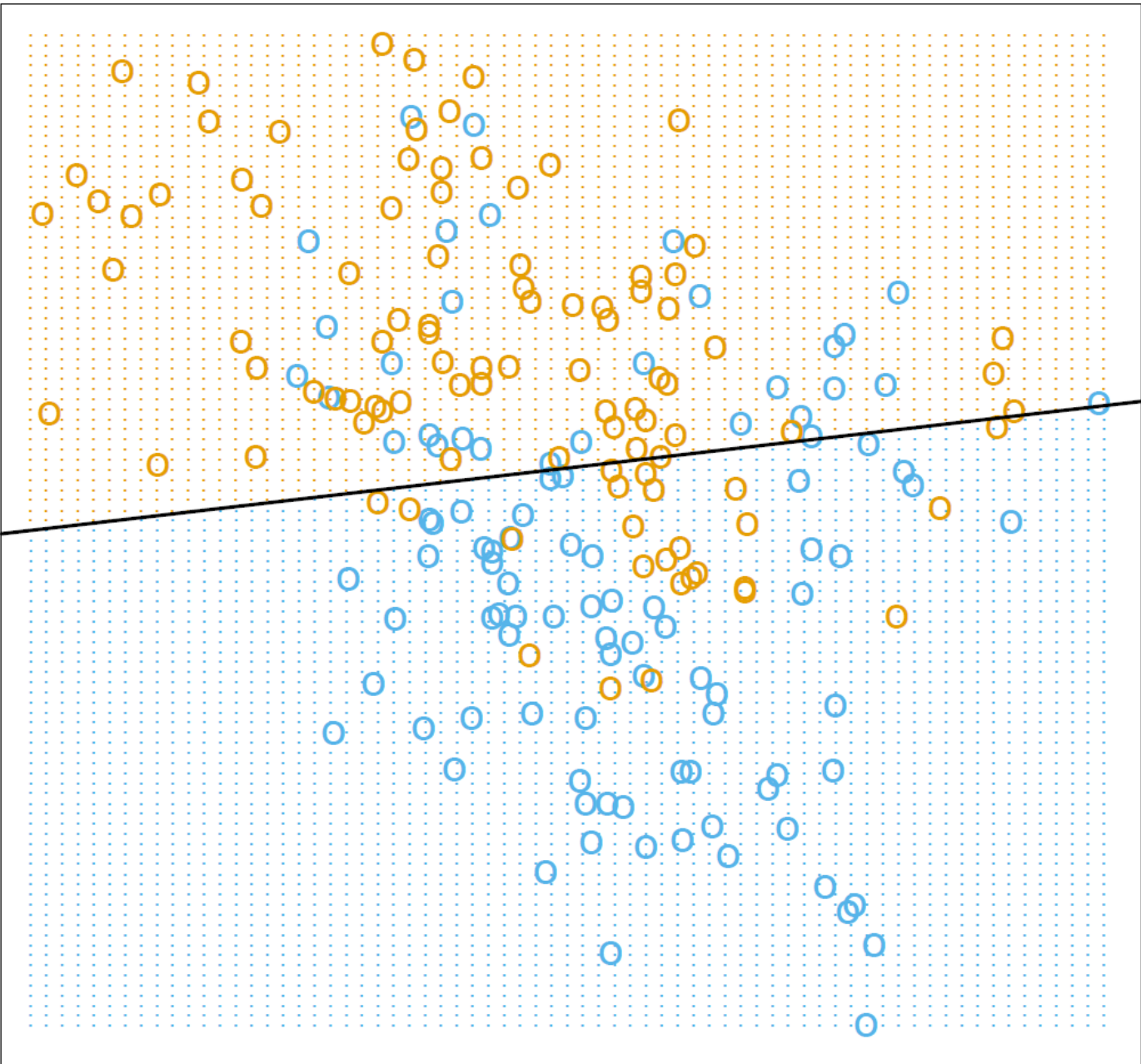
$$\omega_1 : \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0.6 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix}, \begin{bmatrix} 0.2 \\ 0.7 \end{bmatrix}, \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix}$$

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



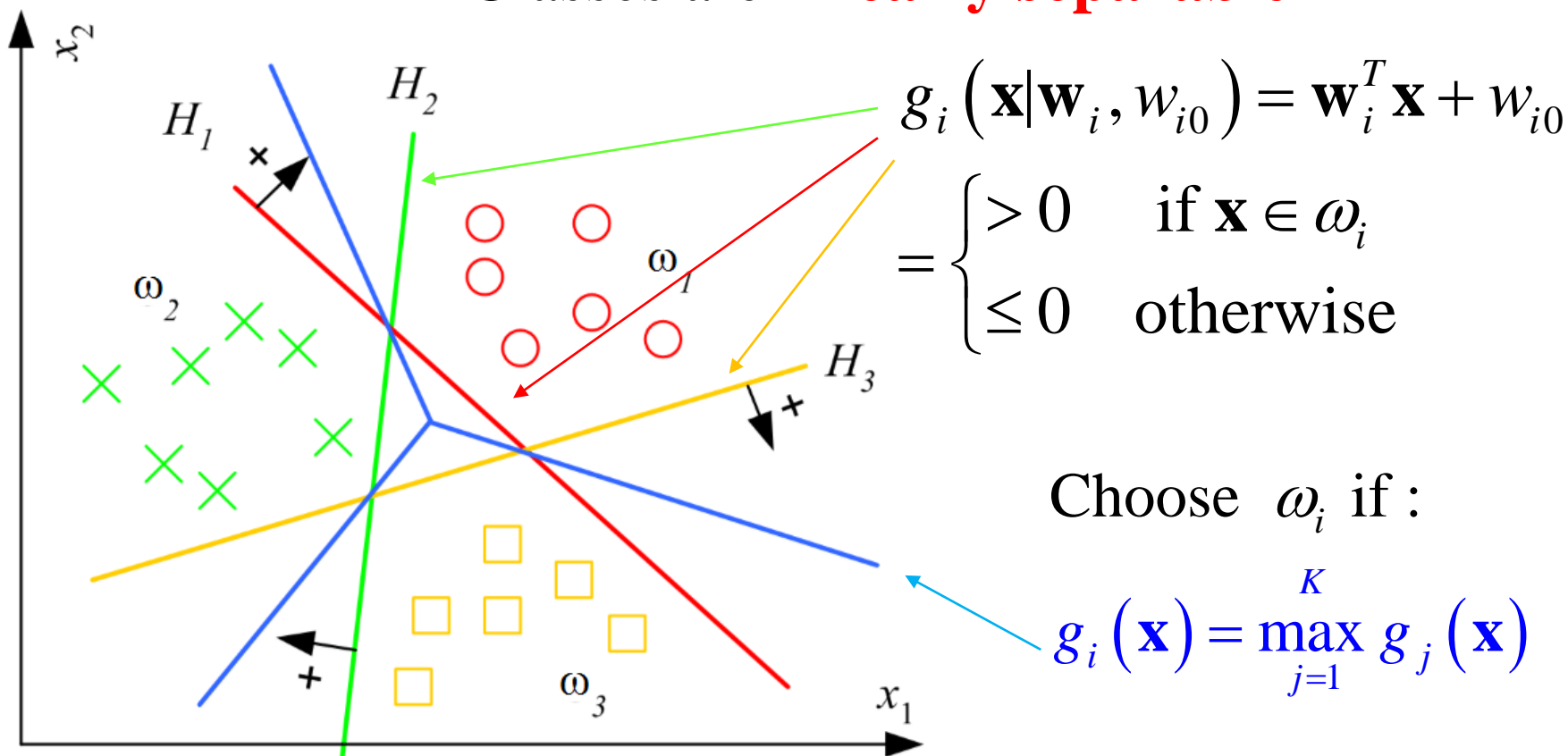
$$X = \begin{bmatrix} 0.4 & 0.5 & 1 \\ 0.6 & 0.5 & 1 \\ 0.1 & 0.4 & 1 \\ 0.2 & 0.7 & 1 \\ 0.3 & 0.3 & 1 \\ 0.4 & 0.6 & 1 \\ 0.6 & 0.2 & 1 \\ 0.7 & 0.4 & 1 \\ 0.8 & 0.6 & 1 \\ 0.7 & 0.5 & 1 \end{bmatrix}, \underline{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 2.8 & 2.24 & 4.8 \\ 2.24 & 2.41 & 4.7 \\ 4.8 & 4.7 & 10 \end{bmatrix}, X^T \underline{y} = \begin{bmatrix} -1.6 \\ 0.1 \\ 0.0 \end{bmatrix}, \underline{w} = (X^T X)^{-1} X^T \underline{y} = \begin{bmatrix} -3.13 \\ 0.24 \\ 1.34 \end{bmatrix}$$



Multiple Classes (Revisited)

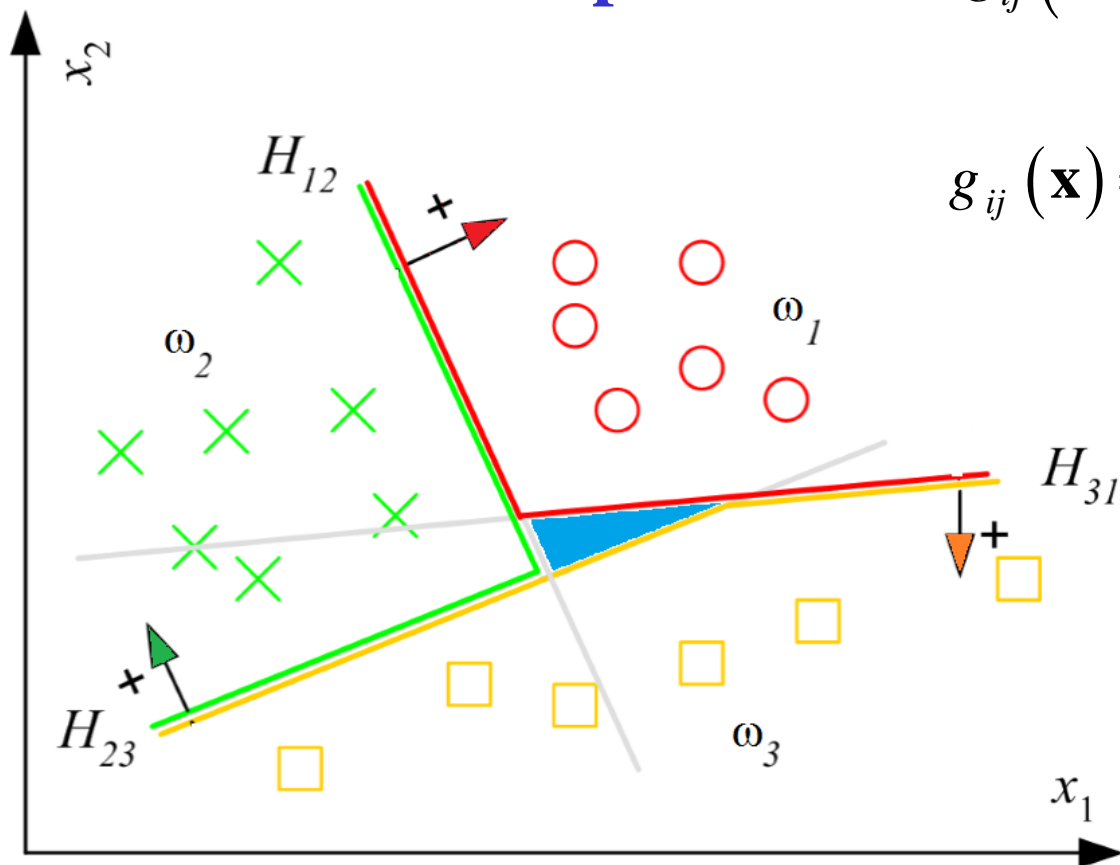
Classes are **linearly separable**



Remembering that $z = |g(\underline{x})| / \|\underline{w}_i\|$ is the distance from the input point to the hyperplane, assuming that all \underline{w}_i have similar length, this assigns the point to the class (among all $g_j(\mathbf{x}) > 0$) to whose hyperplane the point is most distant.

Multiple Classes (Revisited)

Pairwise Separation



$$g_{ij}(\mathbf{x} | \mathbf{w}_{ij}, w_{ij0}) = \mathbf{w}_{ij}^T \mathbf{x} + w_{ij0}$$

$$g_{ij}(\mathbf{x}) = \begin{cases} > 0 & \text{if } \mathbf{x} \in \omega_i \\ \leq 0 & \text{if } \mathbf{x} \in \omega_j \\ \text{don't care} & \text{otherwise} \end{cases}$$

choose ω_i if
 $\forall j \neq i, g_{ij}(\mathbf{x}) > 0$?


reject

If we do not want to **reject** such cases, we can relax the conjunction by using a summation and choosing the maximum of $g_i(\mathbf{x}) = \sum_{j \neq i} g_{ij}(\mathbf{x})$

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 .

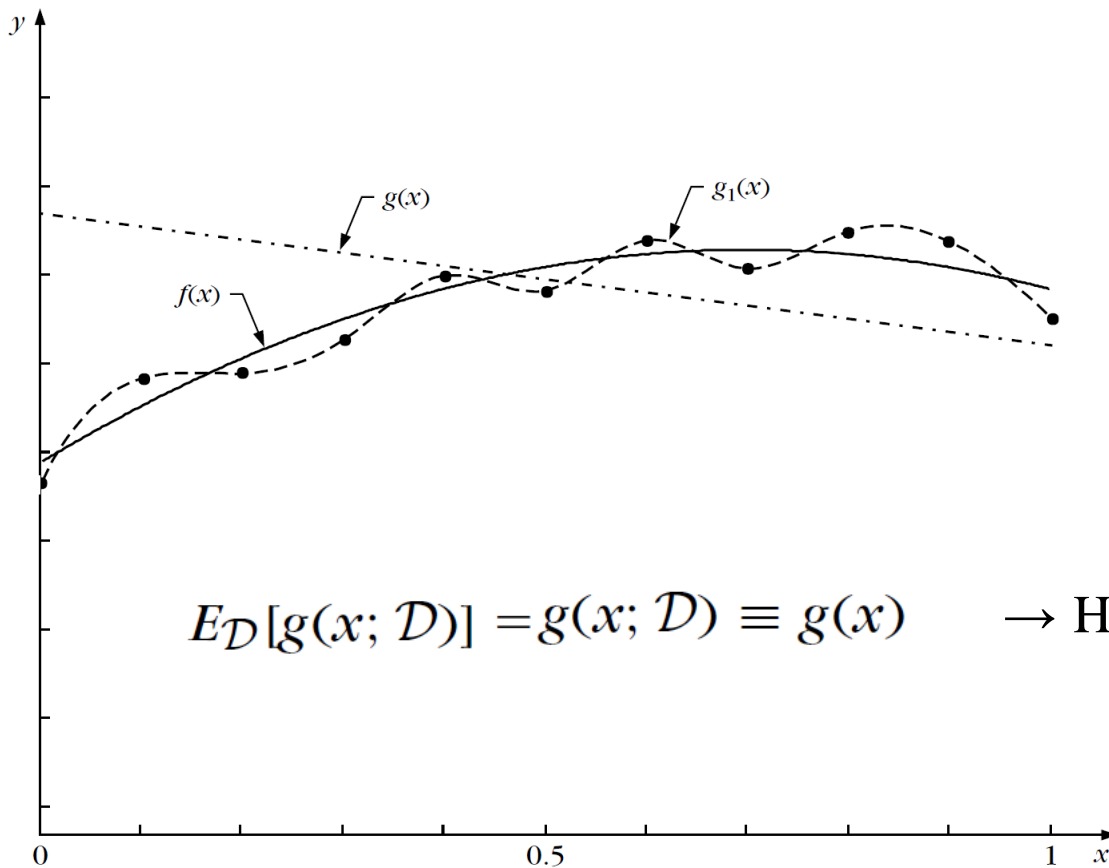
❖ The above is written as:

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

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

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

- ❖ 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 the bias, decreases the 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**.



$$y = f(x) + \epsilon$$

$$g(x) = w_1x + w_0$$

$g_1(x)$ is a polynomial
of high degree

$$E_{\mathcal{D}}[g(x; \mathcal{D})] = g(x; \mathcal{D}) \equiv g(x) \quad \rightarrow \text{High bias, zero variance}$$

$$\begin{aligned} E_{\mathcal{D}} \left[(g_1(x; \mathcal{D}) - E_{\mathcal{D}}[g_1(x; \mathcal{D})])^2 \right] &= E_{\mathcal{D}} \left[(f(x) + \epsilon - f(x))^2 \right] \\ &= \sigma_{\epsilon}^2, \text{ for } x = x_i, i = 1, 2, \dots, N \end{aligned}$$

\rightarrow zero bias, variance is equal to the variance of the noise

From Discriminants to Posteriors

When $p(\mathbf{x} | \omega_i) \sim N(\boldsymbol{\mu}_i, \Sigma)$

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

$$\mathbf{w}_i = \Sigma^{-1} \boldsymbol{\mu}_i, \quad w_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i^T \Sigma^{-1} \boldsymbol{\mu}_i + \log P(\omega_i)$$

$$z \equiv P(\omega_1 | \mathbf{x}) \quad \text{and} \quad P(\omega_2 | \mathbf{x}) = 1 - z$$

$$\text{choose } \omega_1 \text{ if } \begin{cases} z > 0.5 \\ \frac{z}{1-z} > 1 \quad \text{and } \omega_2 \text{ otherwise} \\ \log \frac{z}{1-z} > 0 \end{cases}$$

← the **logit** transformation or **log odds** of z

The **logit** maps the $[0, 1]$ interval onto the real line. In logistic classification, the posterior-probability function is **linear** in logit space. Why? $\rightarrow \dots$

$$\text{logit} \left(P(\omega_1 | \mathbf{x}) \right) = \log \frac{P(\omega_1 | \mathbf{x})}{1 - P(\omega_1 | \mathbf{x})} = \log \frac{P(\omega_1 | \mathbf{x})}{P(\omega_2 | \mathbf{x})}$$

$$= \log \frac{p(\mathbf{x} | \omega_1)}{p(\mathbf{x} | \omega_2)} + \log \frac{P(\omega_1)}{P(\omega_2)}$$

$$= \log \frac{(2\pi)^{-d/2} |\Sigma|^{-1/2} \exp \left[-(1/2) (\mathbf{x} - \mu_1)^T \Sigma^{-1} (\mathbf{x} - \mu_1) \right]}{(2\pi)^{-d/2} |\Sigma|^{-1/2} \exp \left[-(1/2) (\mathbf{x} - \mu_2)^T \Sigma^{-1} (\mathbf{x} - \mu_2) \right]} + \log \frac{P(\omega_1)}{P(\omega_2)}$$

$$= \mathbf{w}^T \mathbf{x} + w_0$$

$$\text{where } \mathbf{w} = \Sigma^{-1} (\mu_1 - \mu_2), \quad w_0 = -\frac{1}{2} (\mu_1 + \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) + \log \frac{P(\omega_1)}{P(\omega_2)}$$

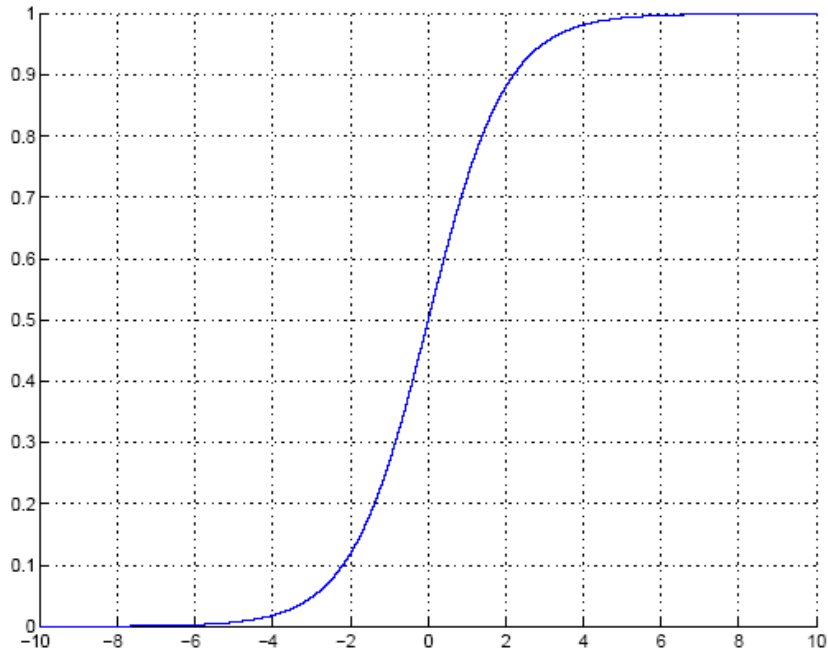
The inverse of logit, $\log \frac{P(\omega_1 | \mathbf{x})}{1 - P(\omega_1 | \mathbf{x})} = \mathbf{w}^T \mathbf{x} + w_0$ is the logistic function,

also called the sigmoid function

$$P(\omega_1 | \mathbf{x}) = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0) = \frac{1}{1 + \exp \left[-(\mathbf{w}^T \mathbf{x} + w_0) \right]}$$

In the case of two normal classes sharing a common covariance matrix, the log odds is linear

Sigmoid (Logistic) Function



Testing phase:

1. Calculate $g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$ and choose ω_1 if $g(\mathbf{x}) > 0$, or
2. Calculate $z = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0)$ and choose ω_1 if $z > 0.5$

LOGISTIC DISCRIMINATION (M = 2)

Two classes: Assume log likelihood ratio is linear (we do not model the class-conditional densities but rather their ratio.)

$$\log \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} = \mathbf{w}^T \mathbf{x} + w_0^o$$

$$\begin{aligned} \text{logit}(P(\omega_1|\mathbf{x})) &= \log \frac{P(\omega_1|\mathbf{x})}{1 - P(\omega_1|\mathbf{x})} = \log \frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} + \log \frac{P(\omega_1)}{P(\omega_2)} \\ &= \mathbf{w}^T \mathbf{x} + w_0 \end{aligned}$$

$$\text{where } w_0 = w_0^o + \log \frac{P(\omega_1)}{P(\omega_2)}$$

$$z = \hat{P}(\omega_1|\mathbf{x}) = \frac{1}{1 + \exp\left[-(\mathbf{w}^T \mathbf{x} + w_0)\right]} = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0)$$

Training: Two Classes

We are given samples of two classes, $\mathbf{x} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, $y_i \in \{1, 0\}$

We assume y_i , given \mathbf{x}_i is Bernoulli i.e. $y_i|\mathbf{x}_i \sim \text{Bernoulli}(z_i)$

$$z = P(\omega_1|\mathbf{x}) = \frac{1}{1 + \exp\left[-(\mathbf{w}^T \mathbf{x} + w_0)\right]} = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0)$$

$$l(\mathbf{w}, w_0|\mathbf{X}) = \prod_{i=1}^N (z_i)^{(y_i)} (1 - z_i)^{(1-y_i)}$$

$$E = -\log l$$

cross-entropy

Maximize $l \equiv$ Minimize E

$$E(\mathbf{w}, w_0|\mathbf{X}) = -\sum_{i=1}^N y_i \log z_i + (1 - y_i) \log (1 - z_i)$$

$$\frac{\partial E}{\partial w_i} = 0, \forall i \quad \frac{\partial E}{\partial w_0} = 0$$

This is a system of $l + 1$ highly nonlinear equations, which must be solved by iterative numerical methods.

Gradient-Descent

❖ $E(\mathbf{w}|X)$ is error with parameters \mathbf{w} on sample X

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} E(\mathbf{w} | X)$$

❖ Gradient $\nabla_{\mathbf{w}} E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_l} \right]^T$

❖ Gradient-descent:

Starts from random \mathbf{w} and updates \mathbf{w} iteratively in the negative direction of gradient

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i \quad w_i = w_i + \Delta w_i$$

Training: Gradient-Descent

$$E(\mathbf{w}, w_0 | X) = -\sum_{i=1}^N y_i \log z_i + (1 - y_i) \log (1 - z_i)$$

If $\varphi = \text{sigmoid}(\alpha)$ $\frac{d\varphi}{d\alpha} = \varphi(1 - \varphi)$

$$\begin{aligned} \Delta w_j &= -\eta \frac{\partial E}{\partial w_j} = \eta \sum_i \left(\frac{y_i}{z_i} - \frac{1 - y_i}{1 - z_i} \right) z_i (1 - z_i) x_{ij} \\ &= \eta \sum_i (y_i - z_i) x_{ij}, \quad j = 1, \dots, l \end{aligned}$$

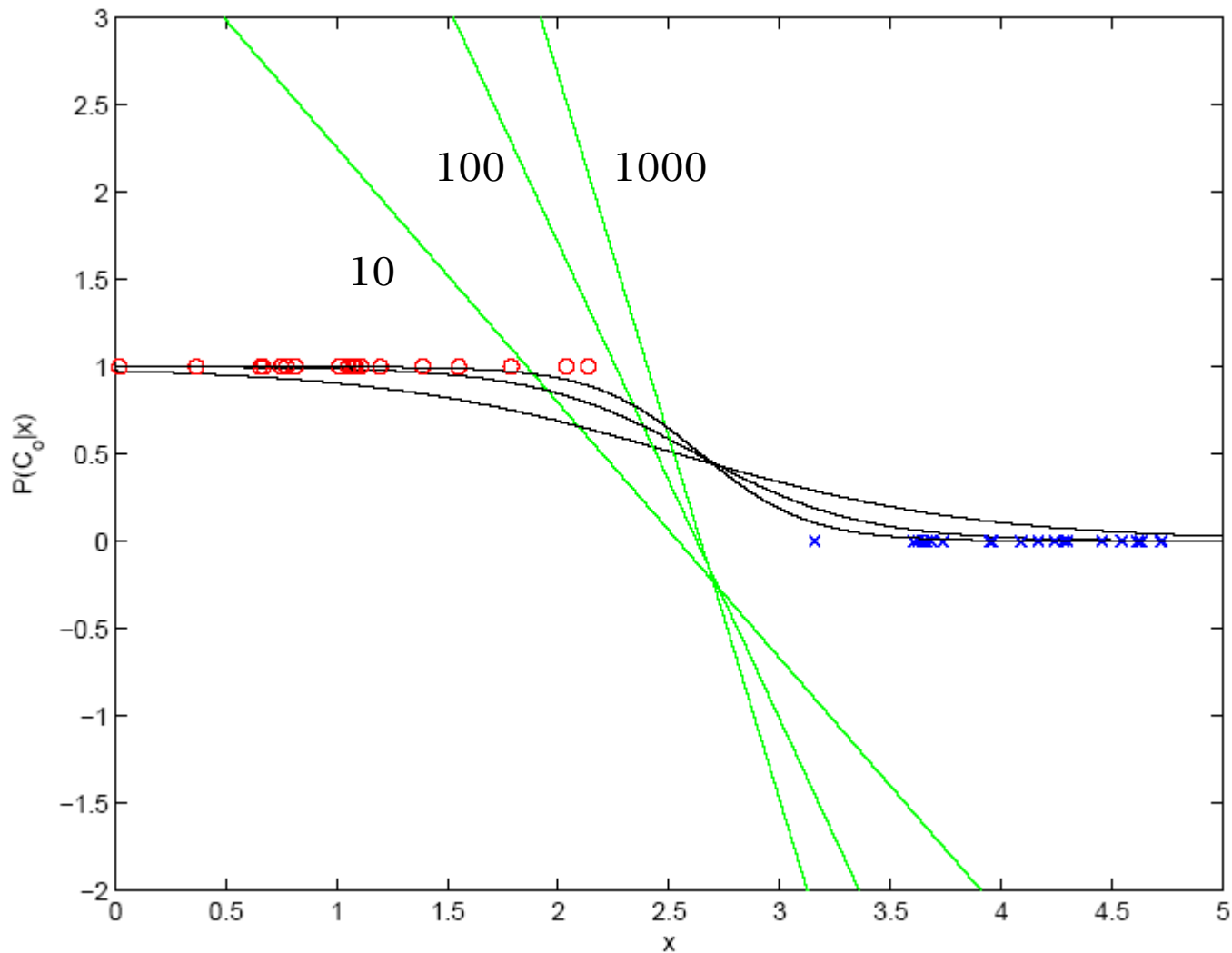
$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_i (y_i - z_i)$$

```

For  $j = 0, \dots, l$ 
     $w_j \leftarrow \text{rand}(-0.01, 0.01)$ 
Repeat
    For  $j = 0, \dots, l$ 
         $\Delta w_j \leftarrow 0$ 
    For  $i = 1, \dots, N$ 
         $o \leftarrow 0$ 
        For  $j = 0, \dots, l$ 
             $o \leftarrow o + w_j \cdot x_{ij}$ 
         $z \leftarrow \text{sigmoid}(o)$ 
         $\Delta w_j \leftarrow \Delta w_j + (y_i - z) x_{ij}$ 
    For  $j = 0, \dots, l$ 
         $w_j \leftarrow w_j + \eta \Delta w_j$ 
Until convergence

```

note: $x_{i0} = +1$ for all i



To get outputs of 0 and 1, the sigmoid hardens, which is achieved by increasing the magnitude of w , or $\|w\|$ in the multivariate case.

To minimize the number of misclassifications, we do not need to continue learning until all y_i are 0 or 1, but only until y_i are less than or greater than 0.5.

STOPPING EARLY, REGULARIZATION, GENERALIZATION.

LOGISTIC DISCRIMINATION ($M > 2$)

- ❖ 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:

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

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

❖ 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.

Training: Multiple Classes ($M > 2$)

We are given samples of M classes, $X = \{\mathbf{x}_j, \mathbf{y}_j\}_{j=1}^N$, $y_{ji} \in \{1, 0\}$

We assume \mathbf{y}_j , given \mathbf{x}_j is Multinomial i.e. $\mathbf{y}_j | \mathbf{x}_j \sim \text{Mult}_M(1, \mathbf{z})$

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

$$l(\{\mathbf{w}_i, w_{i,0}\}_i | \mathbf{X}) = \prod_{j=1}^N \prod_{i=1}^M (z_{ji})^{(y_{ji})}$$

$E = -\log l$ *cross-entropy* **Maximize $l \equiv$ Minimize E**

$$E(\{\mathbf{w}_i, w_{i,0}\}_i | \mathbf{X}) = -\sum_{j=1}^N \sum_{i=1}^M y_{ji} \log z_{ji}$$

Estimates \mathbf{w}_i and $w_{i,0}$ of the logistic curve coefficients are typically obtained by maximizing the conditional log-likelihood l (minimizing E) which lead to a system of highly nonlinear equations.

Training: Gradient-Descent

$$E \left(\left\{ \mathbf{w}_i, w_{i,0} \right\}_i \mid \mathbf{X} \right) = - \sum_{j=1}^N \sum_{i=1}^M y_{ji} \log z_{ji}$$

احتمال تعلق
داده زام به کلاس زام

If $z_i = \text{sigmoid}(\alpha_i) = \frac{\exp(\alpha_i)}{\sum_j \exp(\alpha_j)}$, $\frac{\partial z_{ji}}{\partial \alpha_j} = z_i (\delta_{ij} - z_j)$

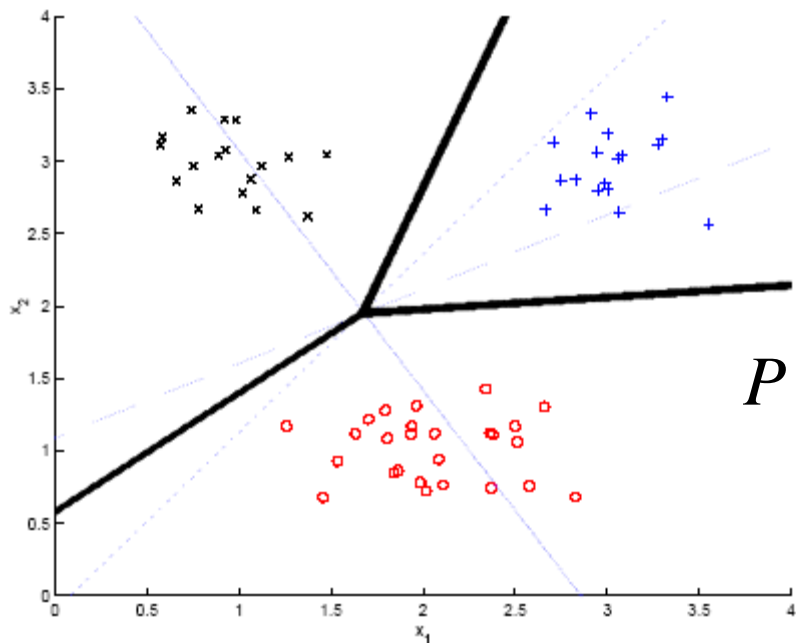
where δ_{ij} is Kronecker delta, which 1 if $i = j$ and 0 if $i \neq j$

$$\begin{aligned} \Delta \mathbf{w}_k &= -\eta \frac{\partial E}{\partial \mathbf{w}_k} = \eta \sum_{j=1}^N \sum_{i=1}^M \left(\frac{y_{ji}}{z_{ji}} \right) z_{ji} (\delta_{jk} - z_{jk}) \mathbf{x}_j = \eta \sum_{j=1}^N \sum_{i=1}^M y_{ji} (\delta_{jk} - z_{jk}) \mathbf{x}_j \\ &= \eta \sum_{j=1}^N \left[\sum_{i=1}^M y_{ji} \delta_{jk} - z_{jk} \sum_{i=1}^M y_{ji} \right] \mathbf{x}_j = \eta \sum_{j=1}^N (y_{jk} - z_{jk}) \mathbf{x}_j, \text{ Note: } \sum_{i=1}^M y_{ji} = 1 \end{aligned}$$

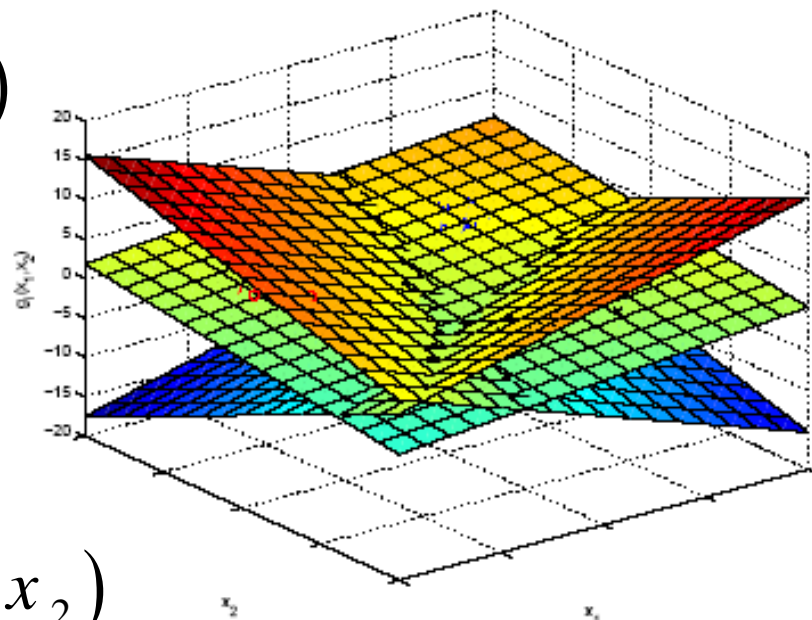
$$\Delta w_{k,0} = -\eta \frac{\partial E}{\partial w_{j,0}} = \eta \sum_{j=1}^N (y_{jk} - z_{jk})$$

The discriminants are updated so that the correct class has the highest weighted sum after softmax, and the other classes have their weighted sums as low as possible. 44

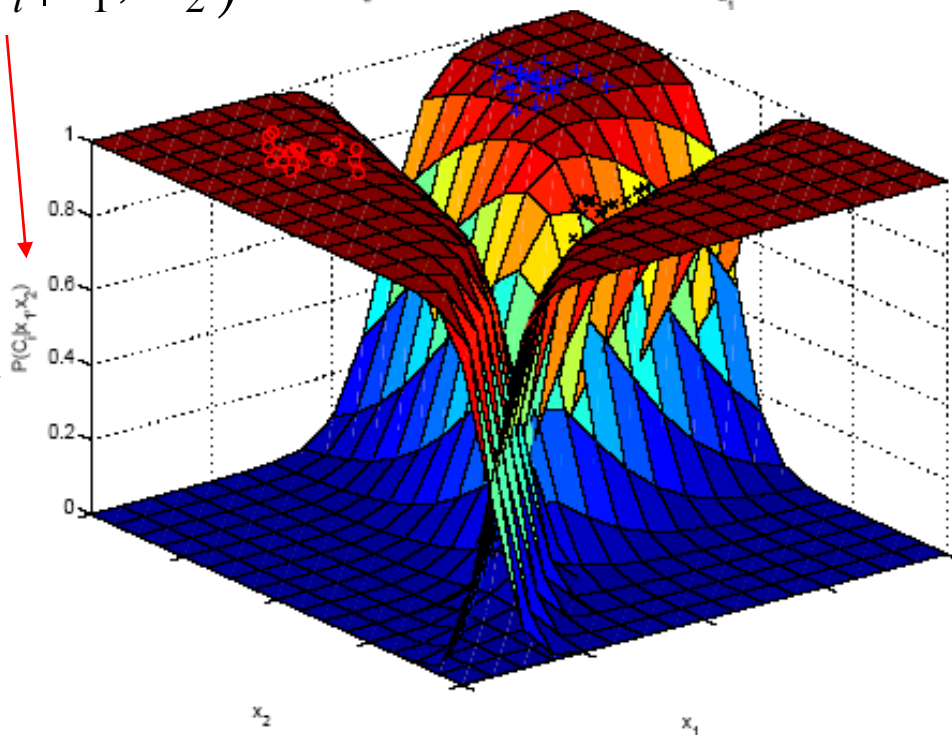
Example:



$$g_i(x_1, x_2)$$



$$P(\omega_i | x_1, x_2)$$



Thin lines are where $g_i(x) = 0$, and the thick line is the boundary induced by the linear classifier choosing the maximum.

- ❖ Logistic discrimination is a useful tool, since it allows linear modeling and at the same time **ensures** posterior probabilities to add to one. ($\mathbf{x}_k^{(m)}$, $k=1,2,\dots,N_m$ are from class m .)

$$L(\boldsymbol{\theta}) = \ln \left\{ \prod_{k=1}^{N_1} p(\mathbf{x}_k^{(1)} | \omega_1; \boldsymbol{\theta}) \prod_{k=1}^{N_2} p(\mathbf{x}_k^{(2)} | \omega_2; \boldsymbol{\theta}) \dots \prod_{k=1}^{N_M} p(\mathbf{x}_k^{(M)} | \omega_M; \boldsymbol{\theta}) \right\}$$

$$p(\mathbf{x}_k^{(m)} | \omega_m; \boldsymbol{\theta}) = \frac{p(\mathbf{x}_k^{(m)}) P(\omega_m | \mathbf{x}_k^{(m)}; \boldsymbol{\theta})}{P(\omega_m)}$$

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

$$L(\boldsymbol{\theta}) = \sum_{k=1}^{N_1} \ln P(\omega_1 | \mathbf{x}_k^{(1)}) + \sum_{k=1}^{N_2} \ln P(\omega_2 | \mathbf{x}_k^{(2)}) + \dots + \sum_{k=1}^{N_M} \ln P(\omega_M | \mathbf{x}_k^{(M)}) + C$$

$$C = \ln \frac{\prod_{k=1}^N p(\mathbf{x}_k)}{\prod_{m=1}^M P(\omega_m)^{N_m}} \quad \text{Any optimization algorithm can then be used to perform the required maximization.}$$

Note: Under the Gaussian assumption and for equal covariance matrices across all classes the following holds true

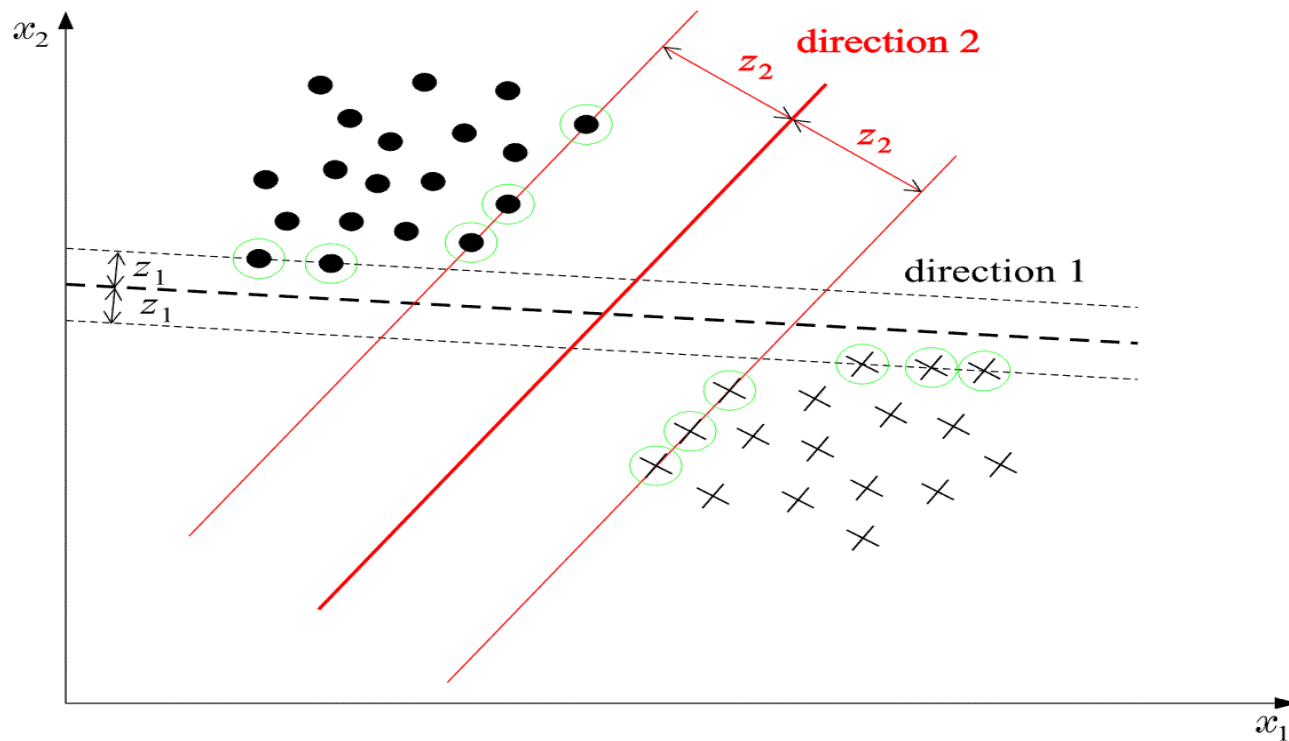
$$\ln \frac{P(\omega_1 | \mathbf{x})}{P(\omega_2 | \mathbf{x})} = \frac{1}{2} (\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} \mathbf{x} \equiv w_0 + \mathbf{w}^T \mathbf{x}$$

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.

- Our goal is to search for the direction that gives the maximum possible margin.

- 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, \underline{w} , w_0 , so that at the **nearest points** from each class the discriminant function is ± 1 :

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

- Thus the **margin** is given by

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

- Also, the following is valid

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

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

➤ **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

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

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

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

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

the margin $\frac{2}{\|\underline{w}\|}$ is maximized

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

$$(1) \quad \frac{\partial}{\partial \underline{w}} L(\underline{w}, w_0, \underline{\lambda}) = \underline{0} \quad (2) \quad \frac{\partial}{\partial w_0} L(\underline{w}, w_0, \underline{\lambda}) = 0$$

$$(3) \quad \lambda_i \geq 0, \quad i = 1, 2, \dots, N$$

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

Where $L(\cdot)$ is the **Lagrangian function**

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

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

- The solution to the constrained-optimization problem is determined by the saddle point of the Lagrangian function $L(\mathbf{w}, w_0, \boldsymbol{\lambda})$.
- A saddle point of a Lagrangian is a point where the roots are real, but of opposite signs; such a singularity is always unstable.
- The saddle point has to be minimized with respect to \mathbf{w} and w_0 ; it also has to be maximized with respect to $\boldsymbol{\lambda}$.
- Application of optimality condition 1 to the Lagrangian function yields the following

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

- Application of optimality condition 2 to the Lagrangian function yields

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

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 $N_s \leq N_0$, corresponding to **positive** Lagrange multipliers

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

- It is also important to note that for all the constraints that are not satisfied as equalities, the corresponding multiplier λ_i must be **zero**.
- 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).

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

❖ The optimal hyperplane classifier of a SVM is **UNIQUE**.

❖ Although the solution is unique, the resulting Lagrange multipliers are **not** unique.

❖ Feature vectors corresponding to $\lambda_i=0$ can either lie outside the “class separation band,” defined as the region between the two hyperplanes, or they can also lie on one of these hyperplanes.

❖ Although \underline{w} is explicitly given, w_0 can be implicitly obtained by any of the (complementary slackness) conditions.

❖ 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 $L(\underline{w}, w_0, \underline{\lambda})$ respect to $\underline{\lambda}$

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

- subject to

$$(1) \quad \underline{w} = \sum_{i=1}^N \lambda_i y_i \underline{x}_i \quad (2) \quad \sum_{i=1}^N \lambda_i y_i = 0, \quad (3) \quad \underline{\lambda} \geq \underline{0},$$

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

$$\underline{w}^T \underline{w} = \sum_{i=1}^N \lambda_i y_i \underline{w}^T \underline{x}_i = \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \underline{x}_i^T \underline{x}_j$$

❖ Combine the above to obtain

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

❖ We may now state the dual problem as follows:

❖ Given the training sample $\mathbf{x} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, $y_i \in \{1, -1\}$ find the Lagrange multipliers $\underline{\lambda} = \{\lambda_i\}_{i=1}^N$, that maximize the objective function

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

➤ subject to

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

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

➤ Remarks:

- Support vectors enter into the game in pairs, in the form of **inner products**

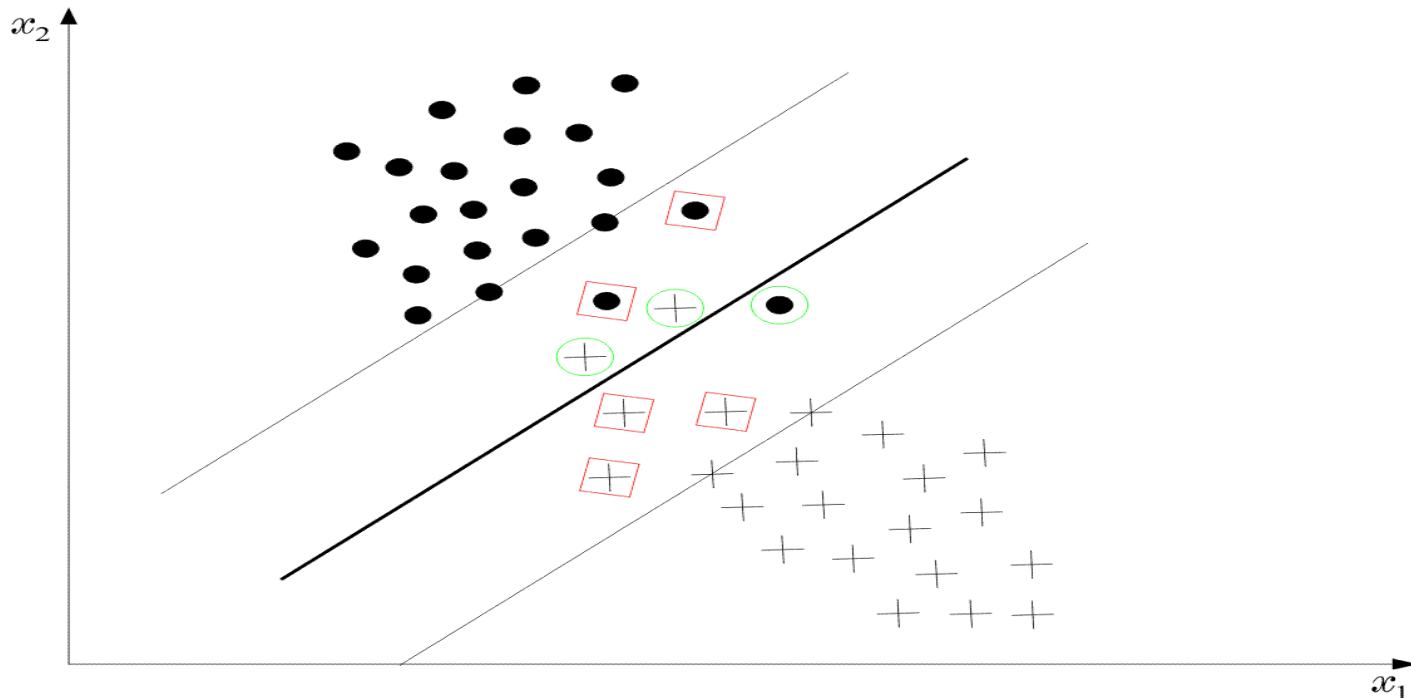
❖ Non-Separable classes

In this case, there is no hyperplane such that

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

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

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



❖ The training vectors belong to **one** of **three** possible categories

1) Vectors **outside** the band which are **correctly** classified, i.e.,
$$y_i (\underline{w}^T \underline{x} + w_0) \geq 1$$

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

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

$$(1) \rightarrow \xi_i = 0$$

$$(2) \rightarrow 0 < \xi_i \leq 1 \quad \xi_i \text{ are known as } \mathbf{slack\ variables}$$

$$(3) \rightarrow 1 < \xi_i$$

❖ 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

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

➤ $I(\cdot)$ is not differentiable. In practice, we use an approximation

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

➤ Following a similar procedure as before we obtain

➤ The corresponding Lagrangian is given by

to guarantee the positivity of ξ_i

$$L(\underline{w}, w_0, \underline{\xi}, \underline{\lambda}, \underline{\mu}) = \frac{1}{2} \|\underline{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \mu_i \xi_i$$

➤ KKT conditions are: $-\sum_{i=1}^N \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - 1 + \xi_i]$

$$\frac{\partial L}{\partial \underline{w}} = \underline{0} \quad (1) \quad \underline{w} = \sum_{i=1}^N \lambda_i y_i \underline{x}_i$$

$$\frac{\partial L}{\partial w_0} = 0 \quad (2) \quad \sum_{i=1}^N \lambda_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \quad (3) \quad C - \mu_i - \lambda_i = 0, \quad i = 1, 2, \dots, N$$

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

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

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

The associated Wolfe dual representation now becomes

$$\text{Maximize } L(\underline{w}, w_0, \underline{\lambda}, \underline{\xi}, \underline{\mu})$$

Subject to:

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

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

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

$$(4) \quad \mu_i \geq 0, \quad \lambda_i \geq 0, \quad i = 1, 2, \dots, N$$

➤ The associated dual problem

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

subject to

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

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

➤ Remarks:

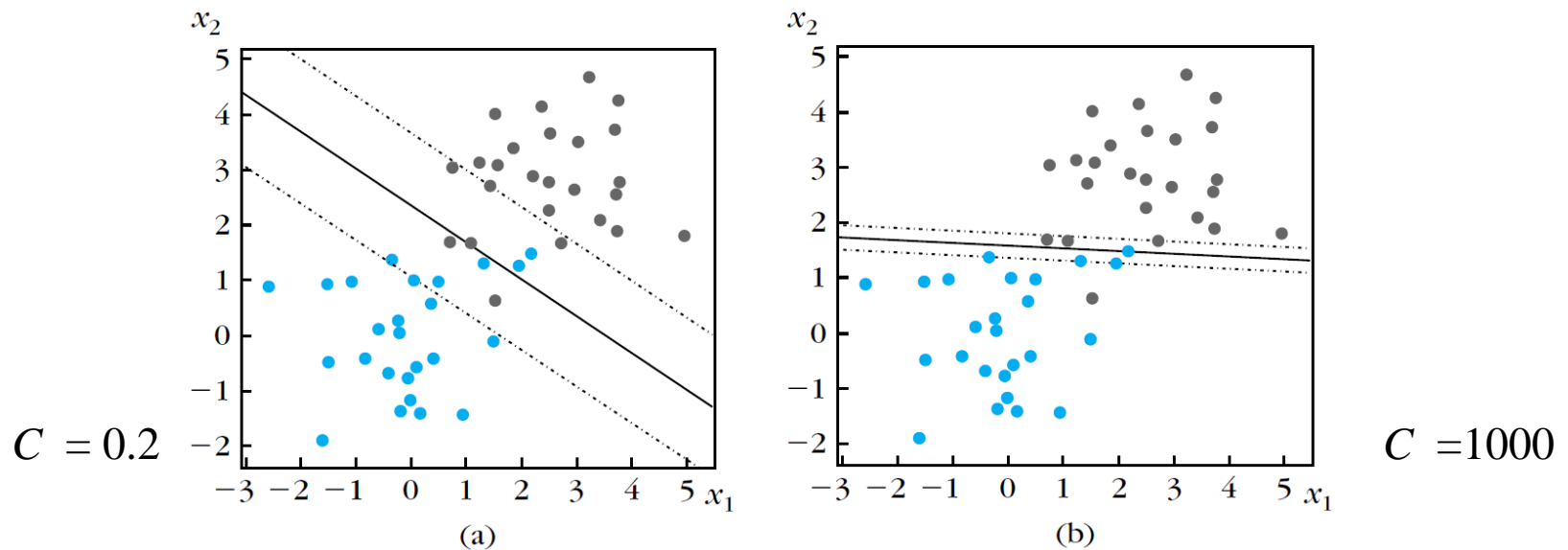
The only difference with the separable class case is the existence of C in the constraints

➤ Training the SVM

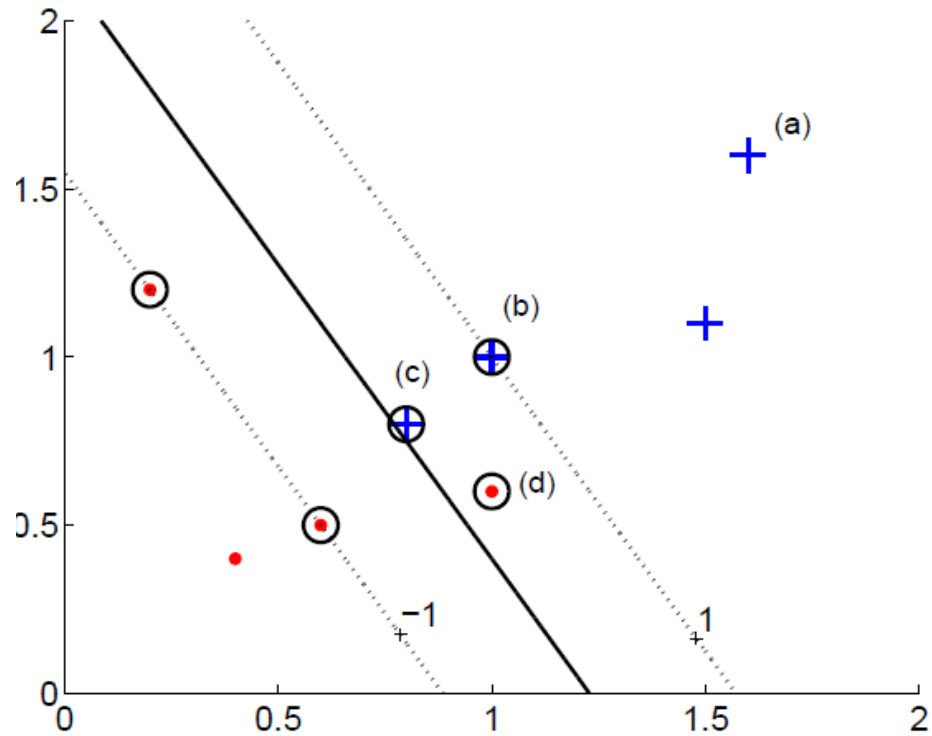
A major problem is the high computational cost. To this end, decomposition techniques are used. The rationale behind them consists of the following:

- 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.

❖ **Example:** Two nonseparable classes and the resulting SVM linear classifier (full line) with the associated margin (dotted lines) for the values (a) $C = 0.2$ and (b) $C = 1000$. In the latter case, the location and direction of the classifier as well as the width of the margin have changed in order to include a smaller number of points inside the margin.

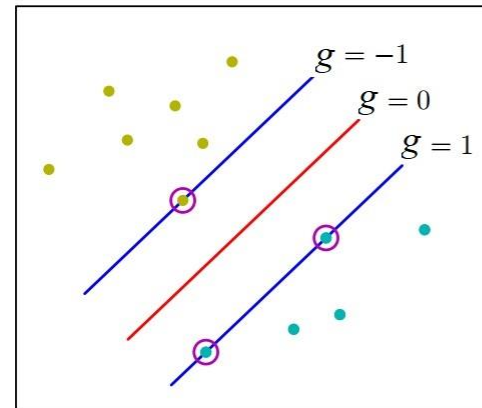
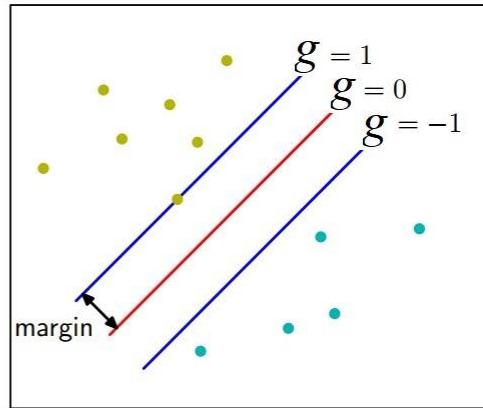


➤ Observe the effect of different values of C in the case of non-separable classes.



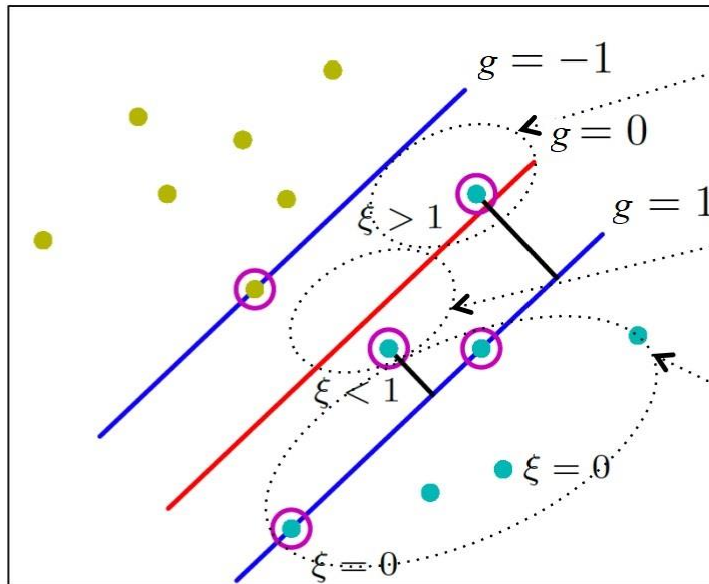
In classifying an instance, there are four possible cases: In (a), the instance is on the correct side and far away from the margin; $y_i g(\mathbf{x}_i) > 1$, $\xi_i = 0$. In (b), $y_i g(\mathbf{x}_i) = 1$, $\xi_i = 0$; it is on the right side and on the margin. In (c), $\xi_i = 1 - y_i g(\mathbf{x}_i)$, $0 < \xi_i < 1$; it is on the right side but is in the margin and not sufficiently away. In (d), $\xi_i = 1 - y_i g(\mathbf{x}_i) > 1$; it is on the wrong side—this is a misclassification. All cases except (a) are support vectors. In terms of the dual variable, in (a), $\lambda_i = 0$; in (b), $\lambda_i < C$; in (c) and (d), $\lambda_i = C$.

Hard Margin



Data points with circles are support vectors

Soft Margin



Data points on wrong side of decision boundary have $\xi > 1$

Correctly classified points inside margin but on correct side of decision boundary have $0 < \xi \leq 1$

Correctly classified points on correct side of margin have $\xi = 0$

❖ 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).

- For each one of the classes, we seek to design an optimal discriminant function,

$$g_i(\underline{x}), \quad i = 1, 2, \dots, M \text{ so that } g_i(\underline{x}) > g_j(\underline{x}), \forall j \neq i, \text{ if } \underline{x} \in \omega_i$$

- Classification rule:

$$\text{assign } \underline{x} \text{ in } \omega_i \text{ if } i = \arg \max_k g_k(\underline{x})$$

- This technique, however, may lead to indeterminate regions
→ one-against-one approach, error correcting coding approach and extending the two class SVM mathematical formulation to the M -class problem.

- ❖ Error Correcting Coding method: M -class, L binary classifiers.
The HD of the code word is measured against the M code words⁶⁸

v-SVM

- ❖ Involve margin in a more direct way in the cost function, instead of leaving its control to a parameter (i.e., C).
- ❖ The margin is defined by the pair of hyperplanes

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

and $\rho \geq 0$ is left as a free variable to be optimized.

$$\text{Minimize } J(\mathbf{w}, w_0, \boldsymbol{\xi}, \rho) = \frac{1}{2} \|\mathbf{w}\|^2 - v\rho + \frac{1}{N} \sum_{i=1}^N \xi_i$$

$$\text{subject to: } y_i (\underline{w}^T \underline{x}_i + w_0) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \rho \geq 0, \quad i = 1, \dots, N$$

- ❖ We simply count and average the number of points with $\xi_i > 0$, whose number is now controlled by the margin variable ρ . The parameter v ($0 \leq v \leq 1$) controls the influence of ρ .

The corresponding Lagrangian is given by

$$L(\underline{w}, w_0, \underline{\lambda}, \underline{\xi}, \underline{\mu}, \rho, \delta) = \frac{1}{2} \|\underline{w}\|^2 - v\rho + \frac{1}{N} \sum_{i=1}^N \xi_i - \frac{1}{N} \sum_{i=1}^N \mu_i \xi_i - \sum_{i=1}^N \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - \rho + \xi_i] - \delta\rho$$

Adopting similar steps the following KKT conditions result:

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

$$(3) \quad \mu_i + \lambda_i = \frac{1}{N}, \quad i = 1, 2, \dots, N, \quad (4) \quad \sum_{i=1}^N \lambda_i - \delta = v$$

$$(5) \quad \lambda_i [y_i (\underline{w}^T \underline{x}_i + w_0) - \rho + \xi_i] = 0, \quad i = 1, 2, \dots, N$$

$$(6) \quad \mu_i \xi_i = 0, \quad i = 1, 2, \dots, N, \quad (7) \quad \delta\rho = 0$$

$$(8) \quad \mu_i \geq 0, \lambda_i \geq 0, \delta \geq 0 \quad i = 1, 2, \dots, N$$

The associated Wolfe dual representation now becomes

$$\text{Maximize } L(\underline{w}, w_0, \underline{\lambda}, \underline{\xi}, \underline{\mu}, \rho, \delta)$$

Subject to:

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

$$(3) \quad \mu_i + \lambda_i = \frac{1}{N}, \quad i = 1, 2, \dots, N, \quad (4) \quad \sum_{i=1}^N \lambda_i - \delta = \nu$$

$$(5) \quad \mu_i \geq 0, \quad \lambda_i \geq 0, \quad \delta \geq 0 \quad i = 1, 2, \dots, N$$

If we substitute the equality constraints in the Lagrangian, the dual problem becomes equivalent to:

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

$$\text{Subject to:} \quad (1) \quad 0 \leq \lambda_i \leq \frac{1}{N}, \quad i = 1, 2, \dots, N,$$

$$(2) \quad \sum_{i=1}^N \lambda_i y_i = 0, \quad (3) \quad \sum_{i=1}^N \lambda_i \geq \nu$$

Once more, only the Lagrange multipliers $\underline{\lambda}$ enter into the problem explicitly, and ρ and the slack variables, ξ_i , make their presence felt through the bounds appearing in the constraints. ν has been shown to be a lower bound on the fraction of support vectors and an upper bound on the fraction of instances having margin errors.

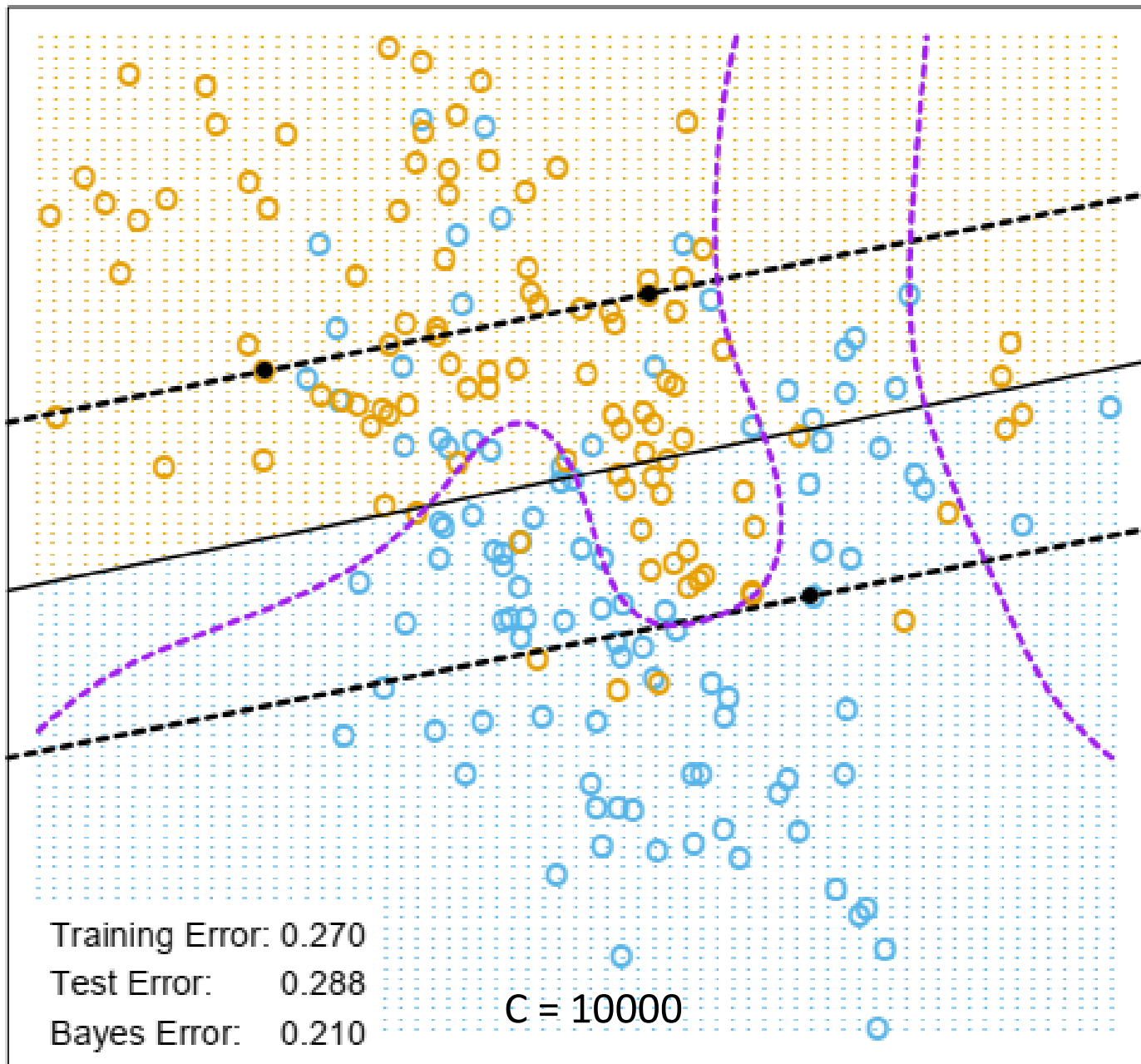
Remarks

- ❖ [Chan 01] shows that the ν -SVM and the more standard SVM formulation (C -SVM), lead to the same solution for appropriate values of C and ν . However, ν offers itself to serve two important bounds concerning (a) the error rate and (b) the number of the resulting support vectors.
- ❖ At the solution, the points lying either within the margin or outside it but on the wrong side of the separating hyperplane correspond to $\xi_i > 0$ and hence to $\mu_i = 0$ forcing the respective Lagrange multipliers to be $\lambda_i = 1/N$.
- ❖ Also, since at the solution, for $\rho > 0$, $\delta = 0$ it turns out that $\sum_{i=1}^N \lambda_i = \nu$.
- ❖ Therefore the total number of errors can, at most, be equal to $N\nu$. the error rate, P_e on the training set is upper-bounded as $P_e \leq \nu$.

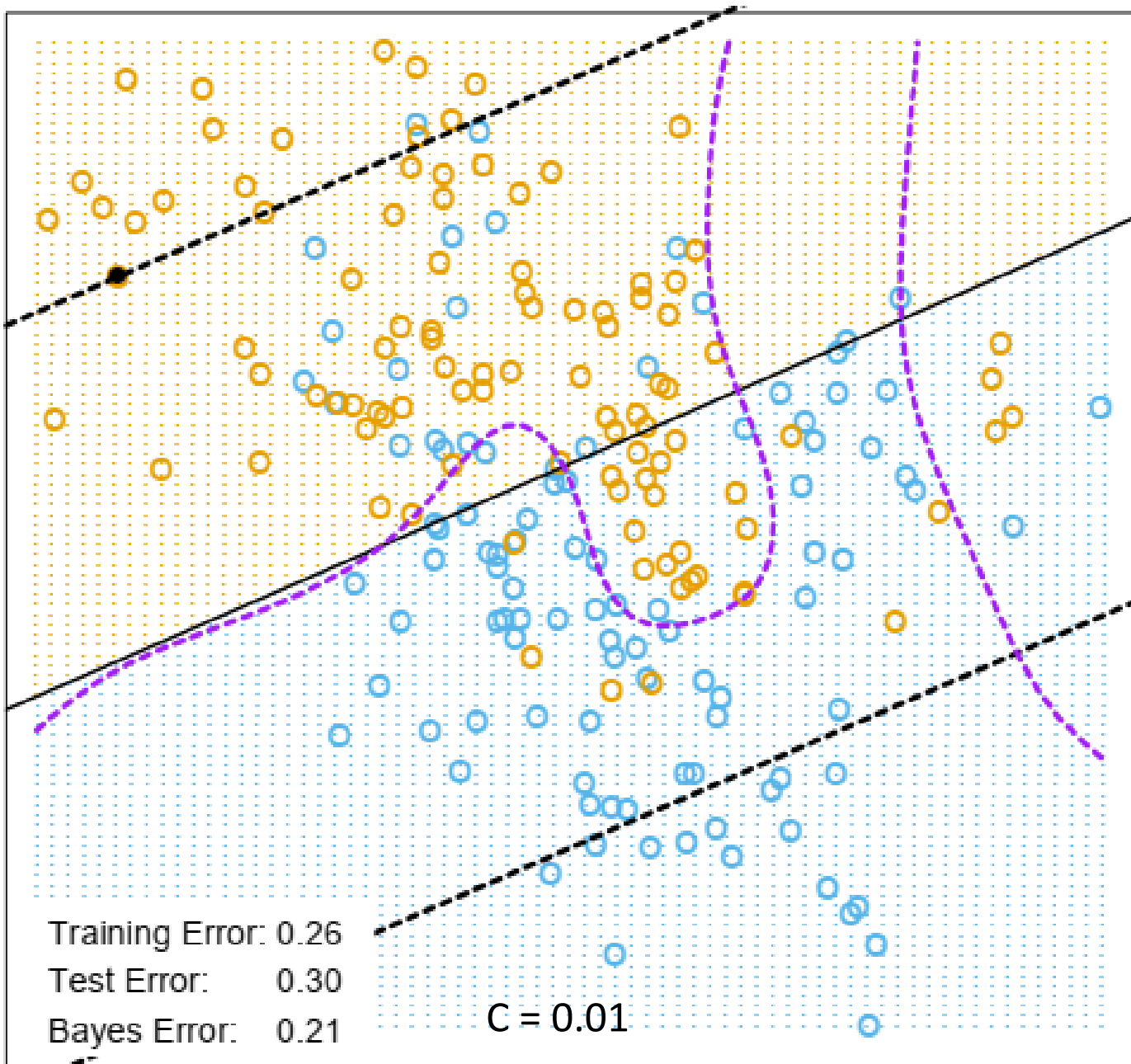
- ❖ Also, at the solution, from the constraints (3.127) and (3.126) we have that

$$\nu = \sum_{i=1}^N \lambda_i = \sum_{i=1}^{N_s} \lambda_i \leq \sum_{i=1}^{N_s} 1/N \text{ or } N\nu \leq N_s.$$

- ❖ Thus, the designer, by controlling the value of ν , may have a feeling for both the error rate on the training set and the number of the support vectors to result from the optimization process.
- ❖ The number of the support vectors, N_s , is very important for the performance of the classifier in practice.
- ❖ It directly affects the computational load, since large N_s means that a large number of inner products are to be computed for classifying an unknown pattern.
- ❖ A large number of support vectors can limit the error performance of the SVM classifier when it is fed with test set (the generalization performance of the classifier).



The broken purple curve in the background is the Bayes decision boundary.



The broken purple curve in the background is the Bayes decision boundary.