Ch3: LINEAR CLASSIFIERS

- 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

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

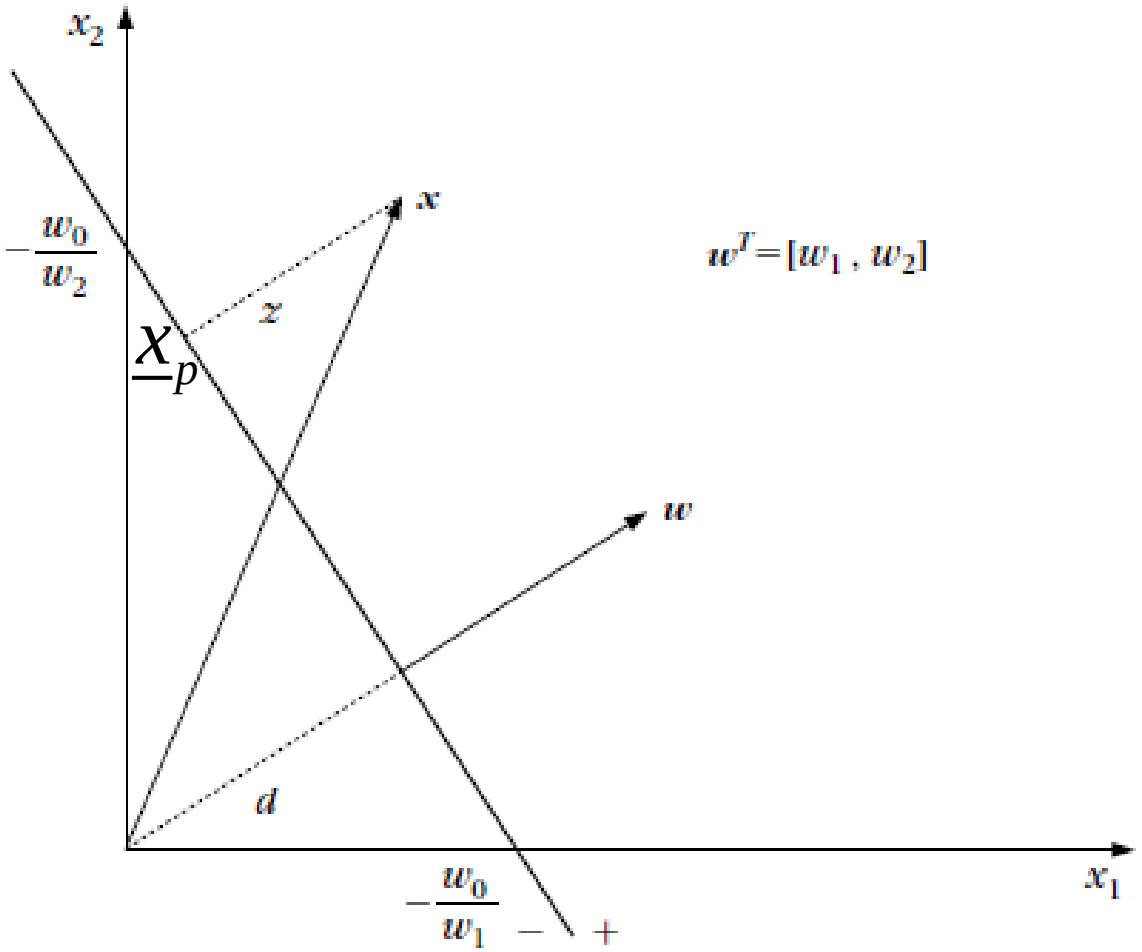
    $\boldsymbol{w} = [w_1, w_2, \ldots, w_l]^T$ is known as the *weight vector* and w_0 as the *threshold*.

  - $$\text{Assume } \underline{x}_1, \underline{x}_2 \text{ on the decision hyperplane:}$$
    $$0 = \underline{w}^T \underline{x}_1 + w_0 = \underline{w}^T \underline{x}_2 + w_0 \Rightarrow$$
    $$\underline{w}^T (\underline{x}_1 - \underline{x}_2) = 0 \quad \forall \underline{x}_1, \underline{x}_2$$

- Hence:

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

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



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

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

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

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

$$= \mathbf{w}^t(\mathbf{x}_p + \frac{z.\mathbf{w}}{\|\mathbf{w}\|}) + w_0 = \mathbf{w}^t\mathbf{x}_p + w_0 + \frac{z\mathbf{w}^t.\mathbf{w}}{\|\mathbf{w}\|} = z \,\|\, \mathbf{w} \,\|$$

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

❖ The Perceptron Algorithm

- Assume linearly separable classes, i.e.,

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

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

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

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

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

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

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

  - 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=(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 $\underline{x} \in \omega_2$

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

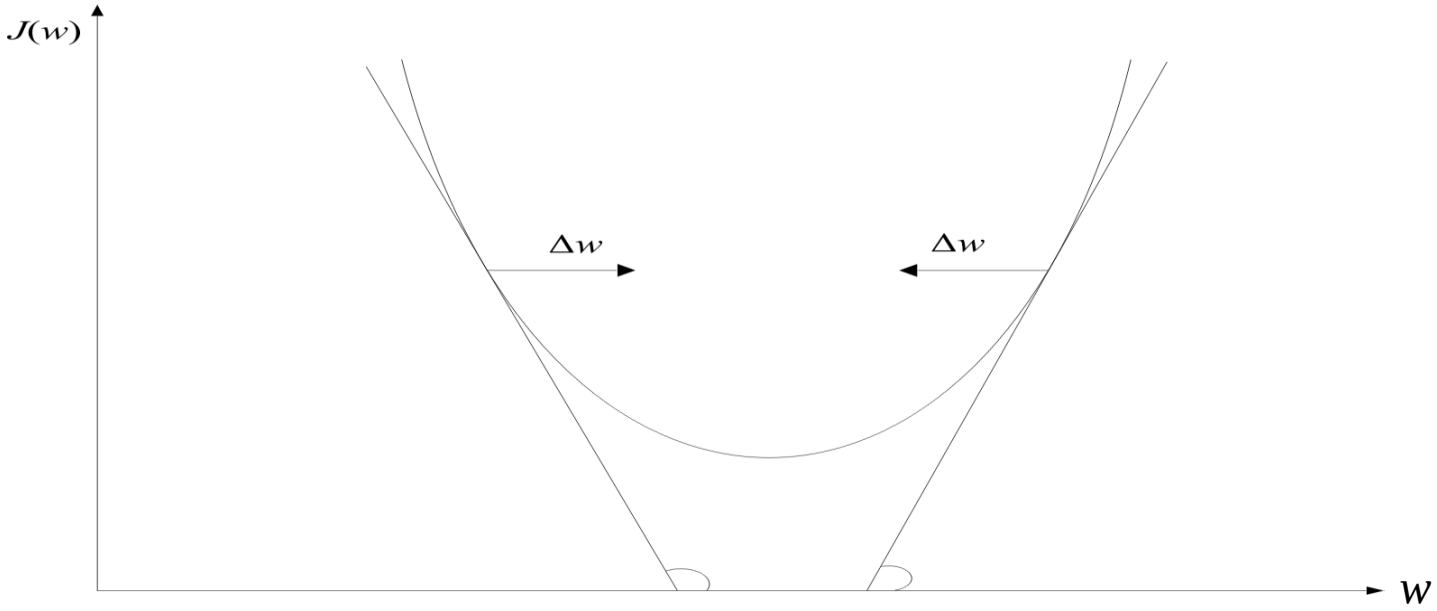
- $J(\underline{w})$ is piecewise linear (WHY?)

➢ 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



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

- Wherever valid

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

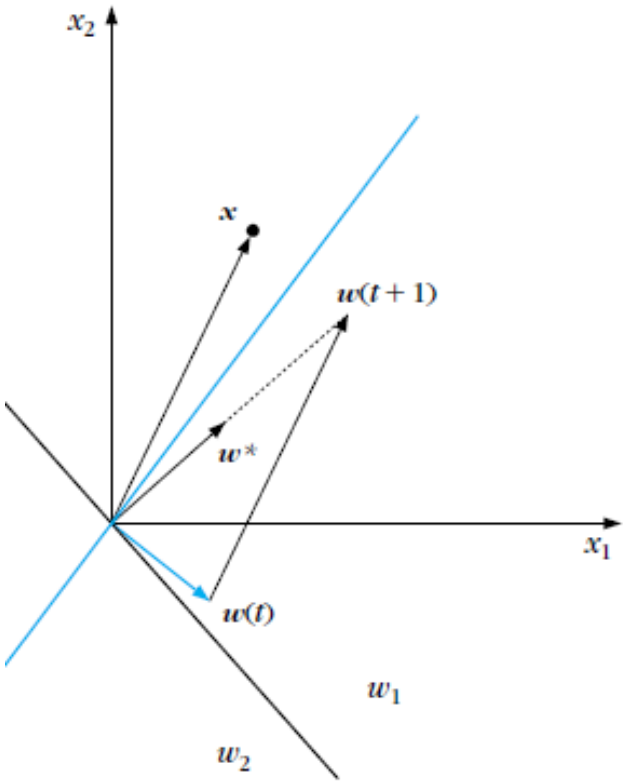
- $$\boxed{\underline{w}(t+1) = \underline{w}(t) - \rho_t \sum_{\underline{x}\in Y} \delta_x \underline{x}}$$

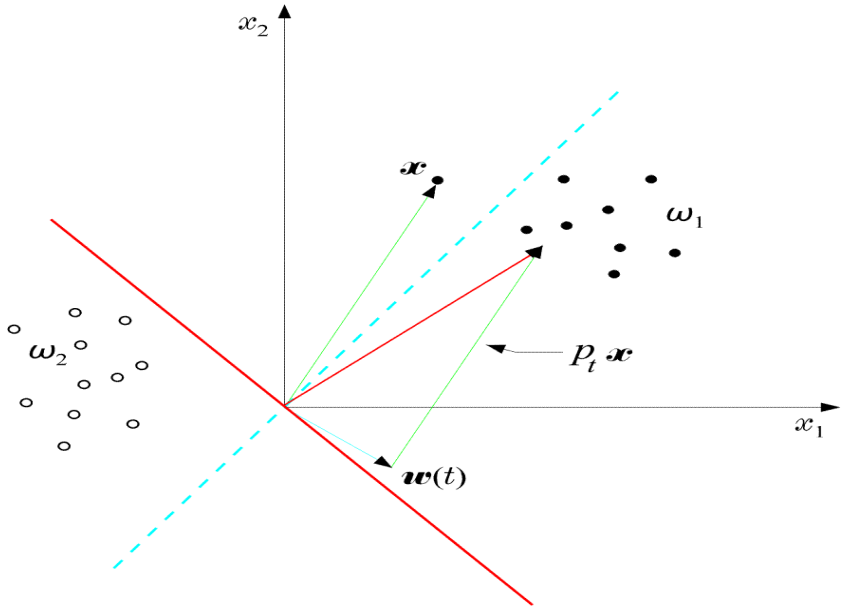
This is the celebrated Perceptron Algorithm

The Perceptron Algorithm

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

- 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

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

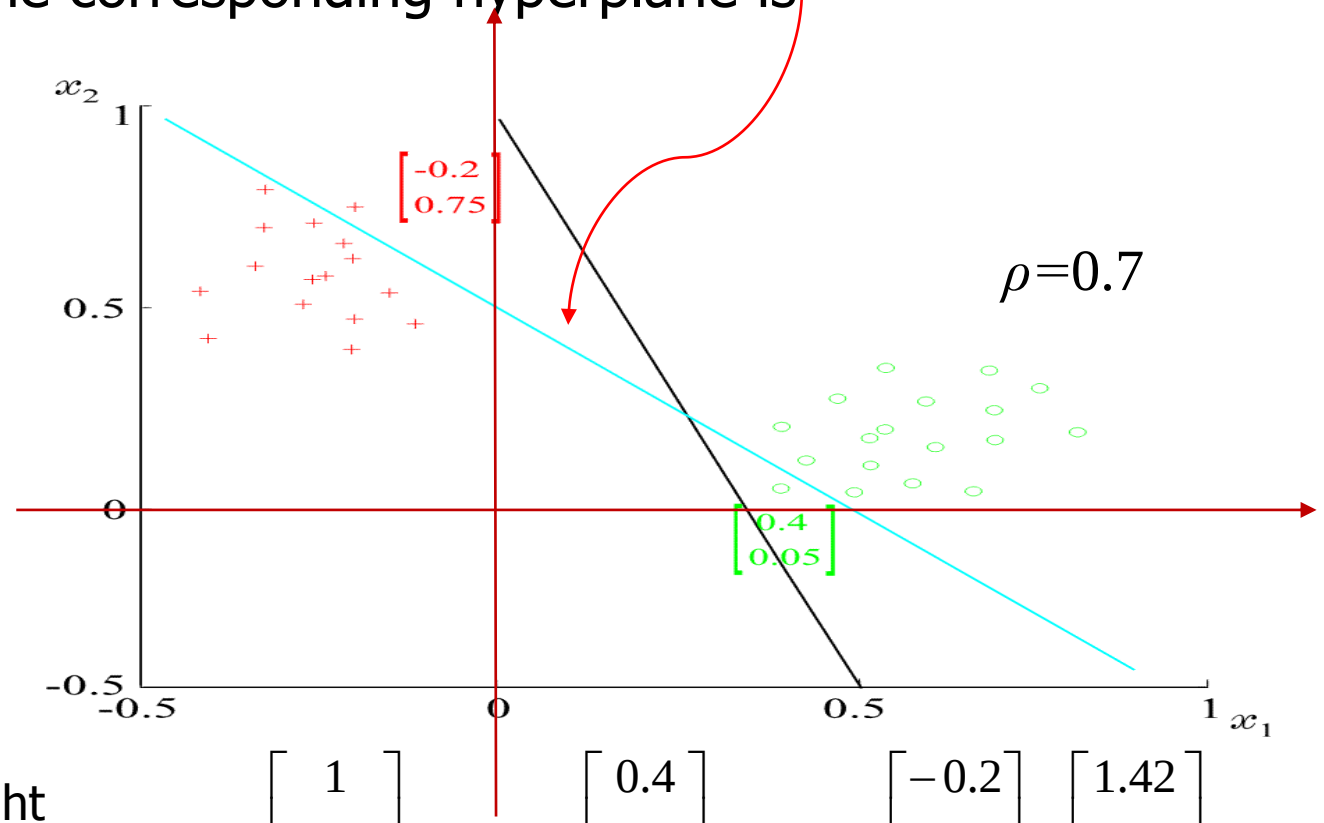
$$\text{e.g.,:}\ \ \rho_t = \frac{c}{t}$$

➢ 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

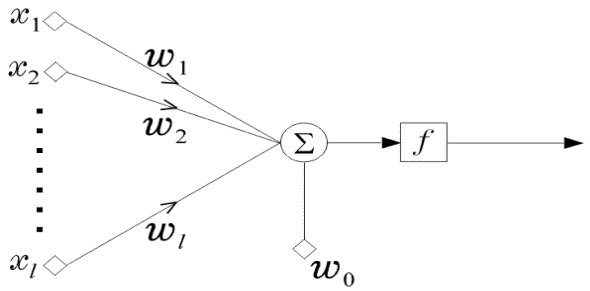
$$\underline{w}(t+1) = \underline{w}(t) + \rho \underline{x}(t), \quad \text{if } \underline{x}(t) \in \omega_1 \text{ and } \underline{w}^T(t)\underline{x}(t) \leq 0$$

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

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

- It is a reward and punishment type of algorithm

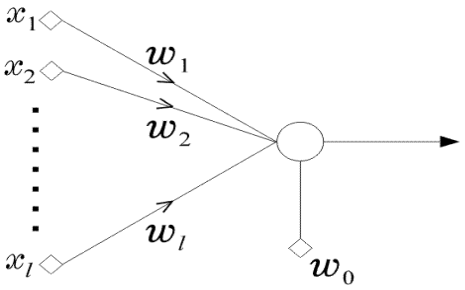
❖ The perceptron



$$> \quad 0$$
$$<$$

$$\text{if } \underline{w}^T \underline{x} + w_0 > 0 \text{ assign } \underline{x} \text{ to } \omega_1$$

$$\text{if } \underline{w}^T \underline{x} + w_0 < 0 \text{ assign } \underline{x} \text{ to } \omega_2$$



$$> \quad 0$$
$$<$$

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 results in ± 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

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

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

- Minimizing

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

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

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

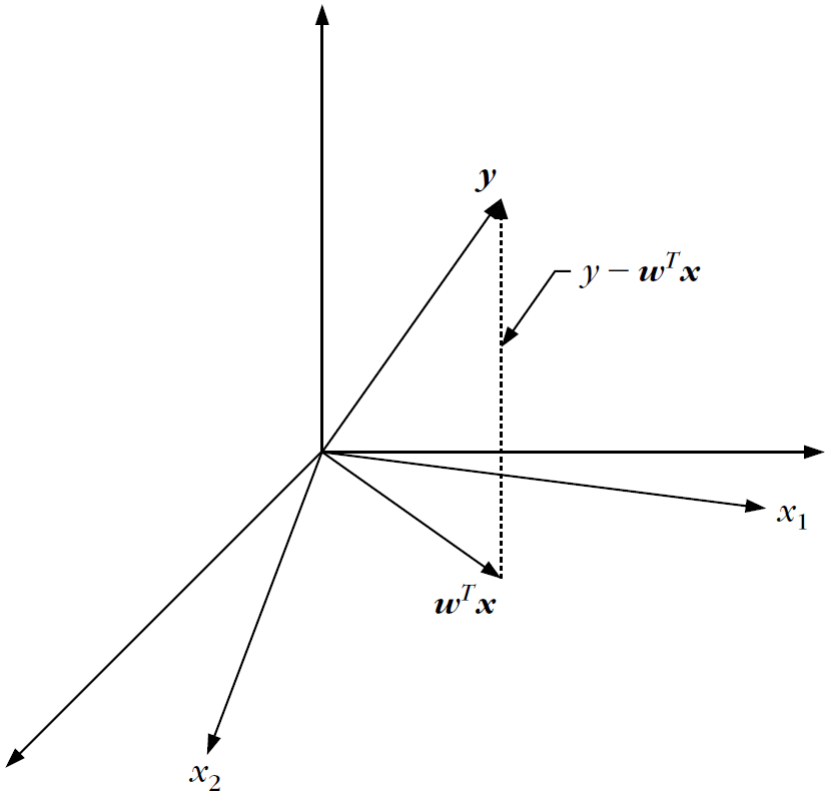
$$E[\underline{x}\underline{x}^T]\underline{w} = E[\underline{x}y] \Rightarrow$$

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

where R_x is the autocorrelation matrix

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

and $E[\underline{x}y] = \begin{bmatrix} E[x_1y] \\ ... \\ E[x_ly] \end{bmatrix}$ the crosscorrelation vector



Interpretation of the MSE estimate as an orthogonal projection on the input vector elements' subspace.

The minimum mean square error solution results if the error is orthogonal to each x_i ; thus it is orthogonal to the vector subspace spanned by x_i ,i = 1, 2, ..., l ; in other words, if y is approximated by its orthogonal projection on the subspace.

- 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

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

  - And the matrix

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

- 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 $P(\omega_i \mid \underline{x})$, provided that the desired responses used during training are $y_i = 1, \underline{x} \in \omega_i$ and 0 otherwise.

- Mean square error regression: Let $\underline{y} \in \Re^M$, $\underline{x} \in \Re^\ell$ 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:

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

  - It turns out that:

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

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.

- 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

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

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

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

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

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

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

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

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

$$X^+ \equiv (X^T X)^{-1} X^T$$ 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:

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

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

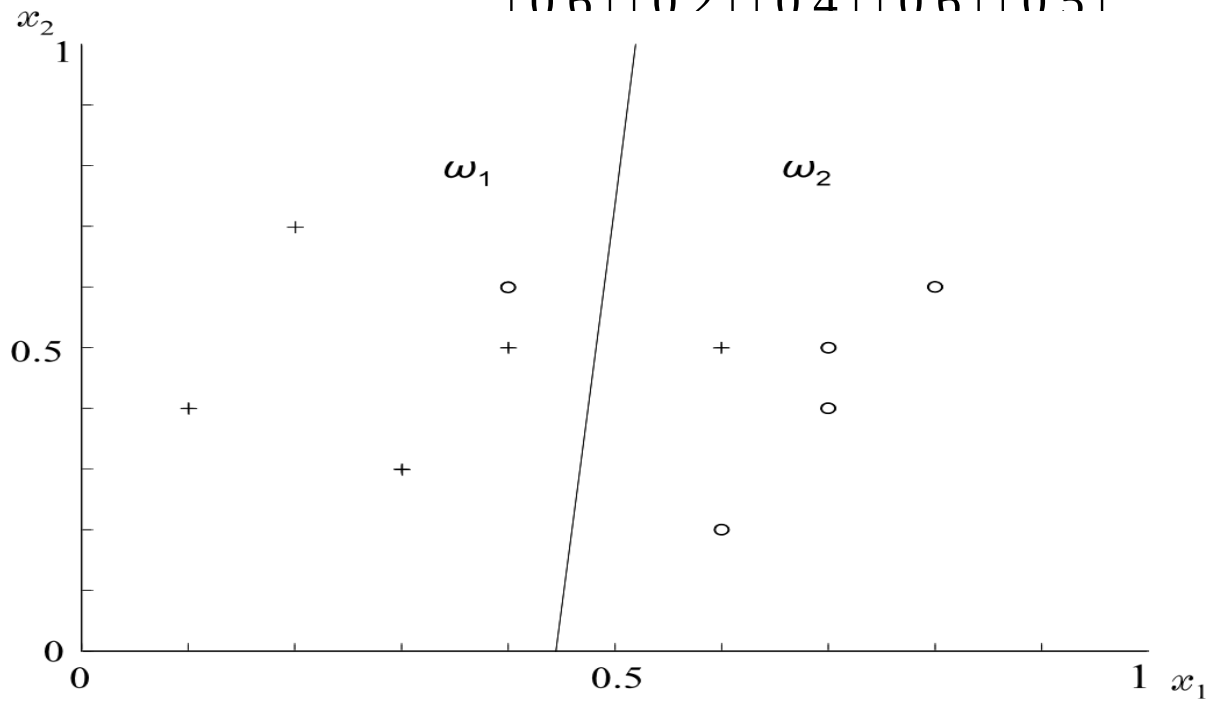
- Assume $N < l$

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

- Example:

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

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



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

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

❖ 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, $D=\{(y_i,\underline{x}_i),i=1,2,...,N\}$, 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 verse. 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, $\omega_1, \omega_2, ..., \omega_M$. In logistic discrimination, the logarithm of the likelihood ratios are modeled via linear functions, i.e.,

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

- Taking into account that

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

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

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

$$P\left(\omega_i \mid \underline{x}\right) = \frac{\exp\left(w_{i,0} + \underline{w}_i^T \underline{x}\right)}{1 + \sum_{i=1}^{M-1} \exp\left(w_{i,0} + \underline{w}_i^T \underline{x}\right)}, i = 1, 2, \ldots \mathrm{M} - 1$$

- For the two-class case it turns out that

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

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

- The unknown parameters $\underline{w}_i$, $w_{i,0}$, $i = 1, 2, \ldots, M - 1$ 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.

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

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

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

$$C = \ln \frac{\prod_{k=1}^{N} p(\boldsymbol{x}_k)}{\prod_{m=1}^{M} P(\omega_m)^{N_m}}$$

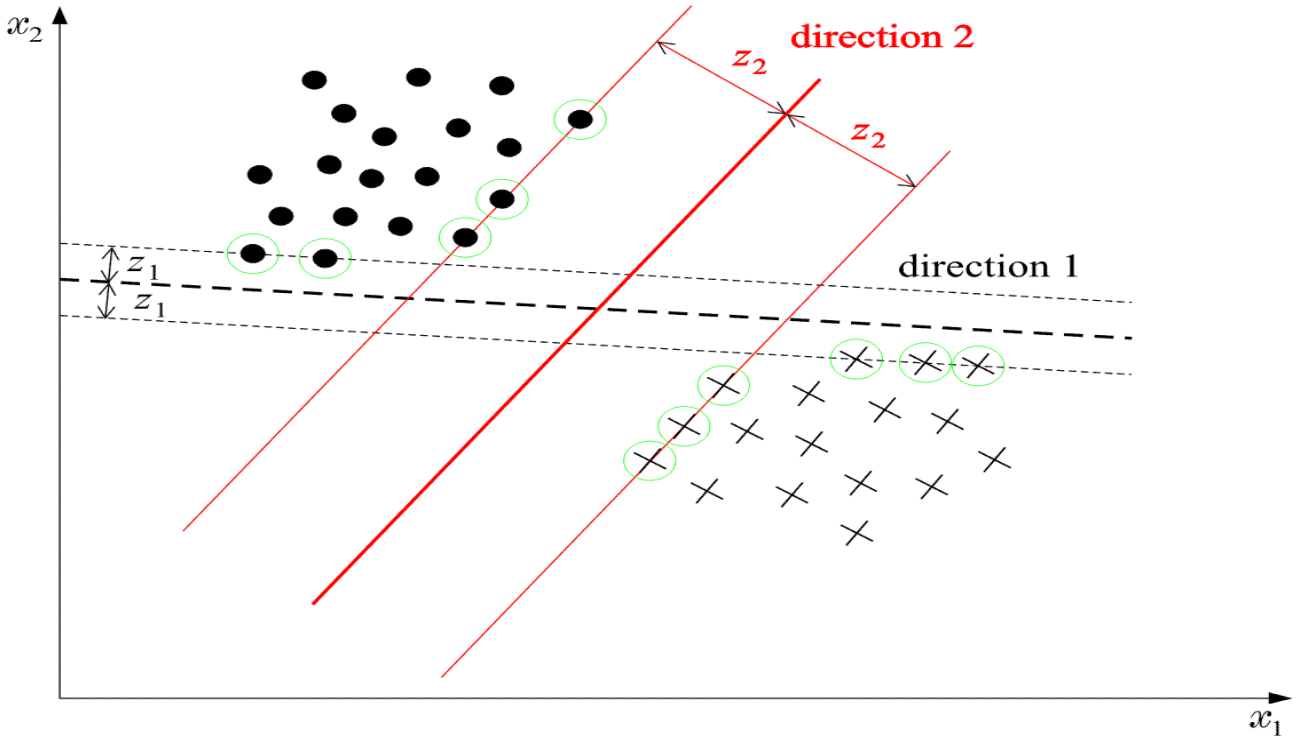
Any optimization algorithm can then be used to perform the required maximization.

❖ 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

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

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

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

- 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

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

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

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

- The above is justified since by minimizing $\|\underline{w}\|$ the margin $\frac{2}{\|w\|}$ is maximized

- The above is a nonlinear (quadratic) optimization task, subject to a set of linear inequality constraints. The Karush-Kuhh-Tucker (KKT) 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 $L(.)$ is the Lagrangian

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

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

$$\underline{w} = \sum_{i=1}^{N} \lambda_i y_i \underline{x}_i \qquad \sum_{i=1}^{N} \lambda_i y_i = 0$$

- 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

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

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).

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

- The optimal hyperplane classifier of a support vector machine 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 $L(\underline{w}, w_0, \underline{\lambda})$
  - 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 into the game in pairs, in the form of inner products

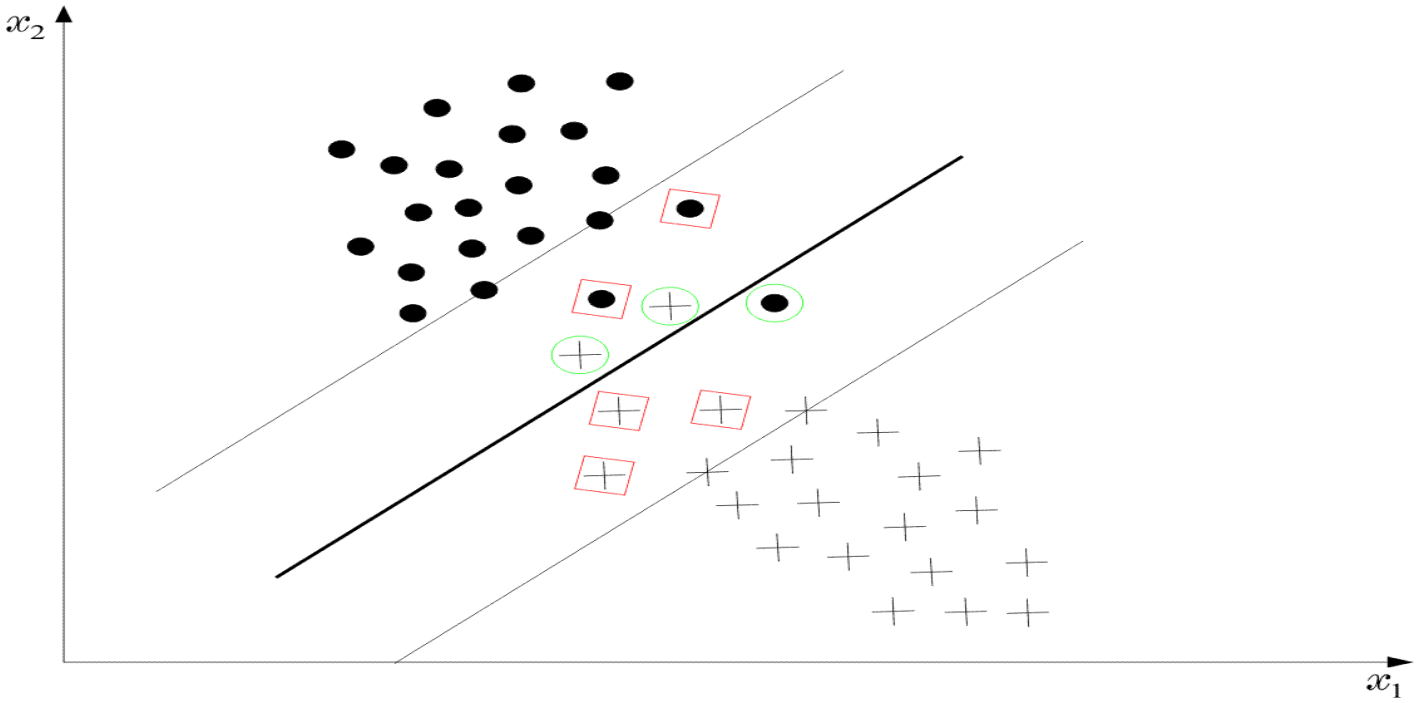
❖ Non-Separable classes

In this case, there is no hyperplane such that

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

➢ 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) $\rightarrow$ $0 < \xi_i \leq 1$

(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 $\xi_i > 0$, 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(.)* 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

➢ KKT conditions

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

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

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

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

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

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

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

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

The associated Wolfe dual representation now becomes

$$\text{maximize} \quad \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}, \boldsymbol{\xi}, \boldsymbol{\mu})$$

$$\text{subject to} \quad \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i$$

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

$$C - \mu_i - \lambda_i = 0, \quad i = 1, 2, \ldots, N$$

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

- The associated dual problem

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

subject to

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

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

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

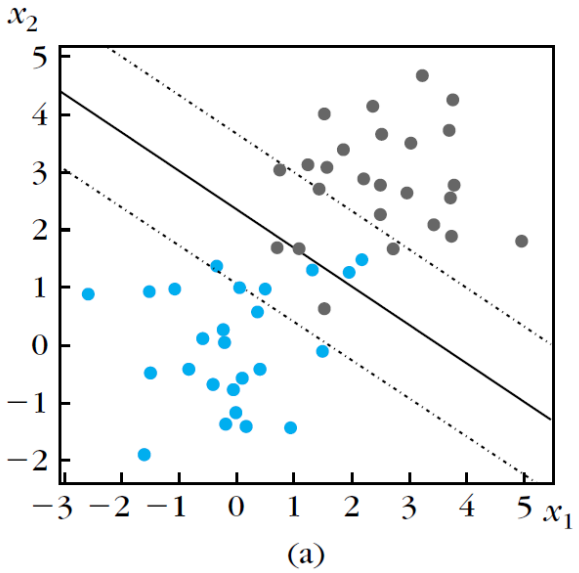
- Training the SVM

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

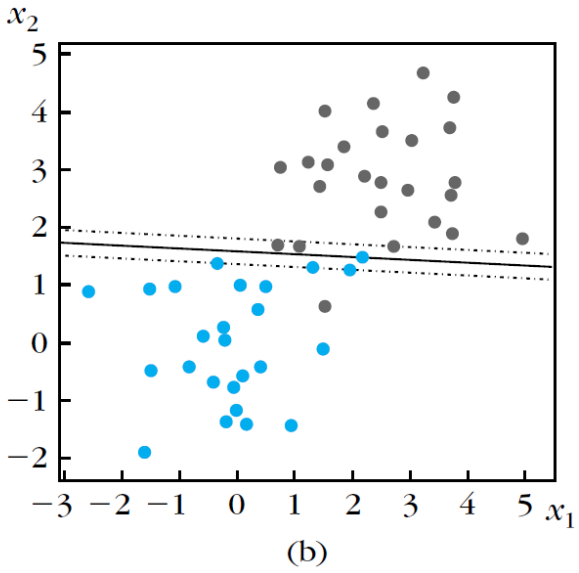
  - Start with an arbitrary data subset (working set) that can fit in the memory. Perform optimization, via a general purpose optimizer.
  - Resulting support vectors remain in the working set, while others are replaced by new ones (outside the set) that violate severely the KKT conditions.
  - Repeat 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.

❖ 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.

C = 0.2



(a)



(b)

C =1000

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

- 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,

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

- 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 $\rightarrow$one-against-one approach, error correcting coding approach and extending the two class SVM mathematical formulation to the M-class problem