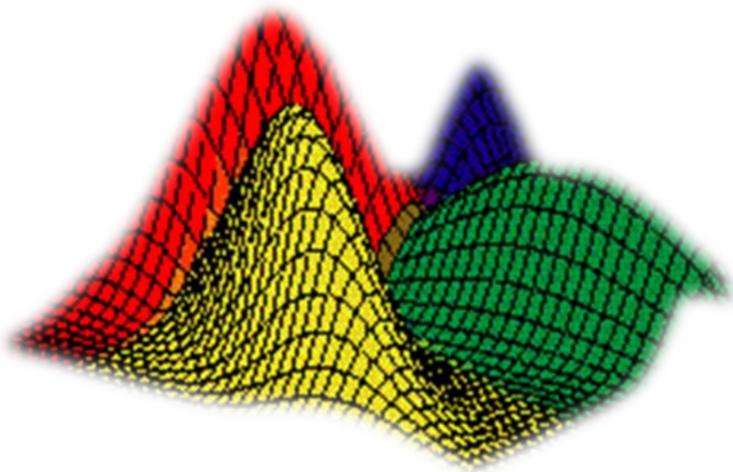
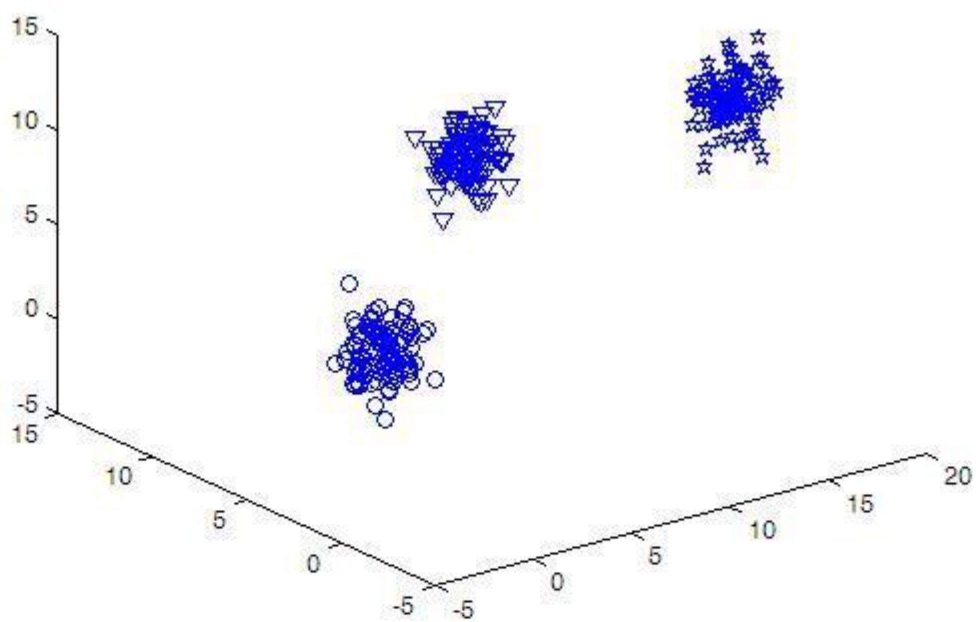


CSE 473
Pattern Recognition

Sample Data for Sessional on Bayesian Classification (Week2)



3 3 300

Sample Data for Bayesian Classification

Feature1	Feature2	Feature3	Class
11.0306	9.0152	8.0199	1
11.4008	8.7768	6.7652	1
11.2489	9.5744	8.0812	1
9.3157	7.4360	5.6128	1
15.7777	1.5879	11.4440	2
15.8685	2.7902	11.2532	2
14.9448	0.7798	12.7481	2
15.9801	1.0142	14.2029	2
2.3979	5.6525	2.7566	3
2.5103	6.3484	1.4272	3
1.3739	3.2679	1.2037	3
2.7527	4.6571	3.1138	3
-0.0195	4.5524	0.0118	3

Assumption on Data Distribution

- Assume multivariate density
  - Multivariate normal density in *d* dimensions is:

$$P(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

where:

$x = (x_1, x_2, \ldots, x_d)^t$ (*t* stands for the transpose vector form)

$\mu = (\mu_1, \mu_2, \ldots, \mu_d)^t$ mean vector

$\Sigma = d*d$ covariance matrix

$|\Sigma|$ and Σ^{-1} are determinant and inverse respectively

Algorithmic Steps

- No. of features and classes will be variable

- Use library functions to find $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $|\boldsymbol{\Sigma}|$ and $\boldsymbol{\Sigma}^{-1}$ from training data for each class

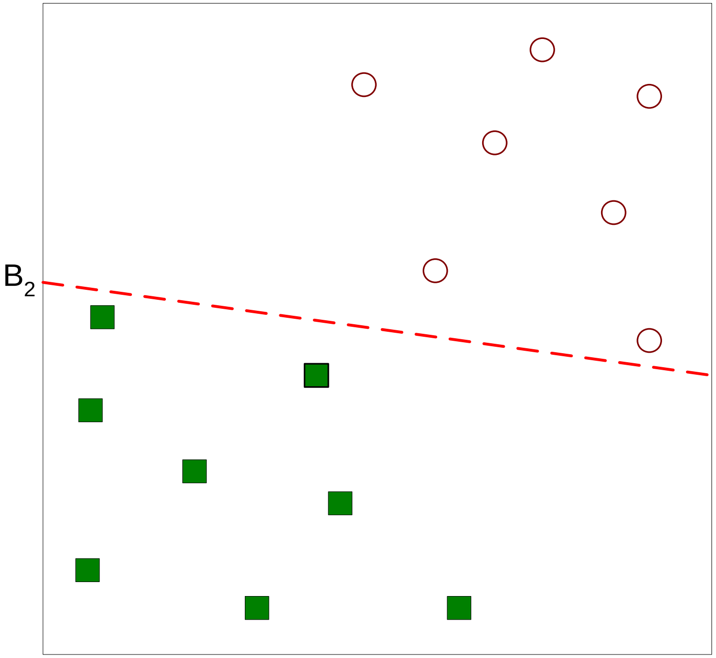
- For a test sample **x**, find $p(c_i|\mathbf{x})$ or $p(c_i)\times p(\mathrm{x}|c_i)$ for each class c_i where

$$P(\mathbf{x} \mid c_i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

- Use different data files during evaluation

Linear Classifier: Introduction

- Classifies linearly separable patterns
- Assume proper forms for the discriminant functions
- may not be optimal
- very simple to use



Linear discriminant functions and decisions surfaces

- Definition

Let a pattern vector $\mathbf{x} = \{x_1, x_2, x_3, \ldots,\}$

a weight vector $\mathbf{w} = \{w_1, w_2, w_3, \ldots,\}$

A discriminant function :

$$g(\mathbf{x})=x_1w_1+x_2w_2+x_3w_3+\ldots$$

OR

$$g(\mathbf{x}) = w^t x + w_0$$

where w is the weight vector and w_0 the bias

- 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 $\underline{w}$.

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

– 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

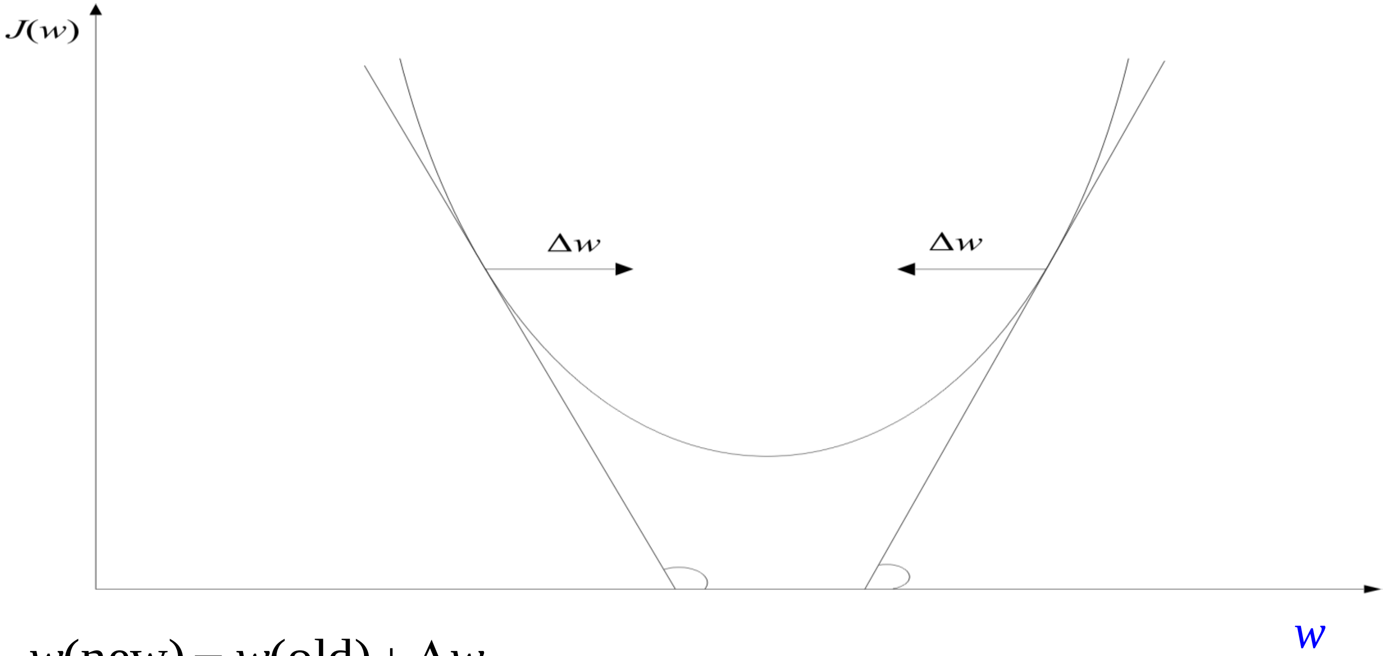
$$J(\underline{w}) = 0$$

otherwise

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

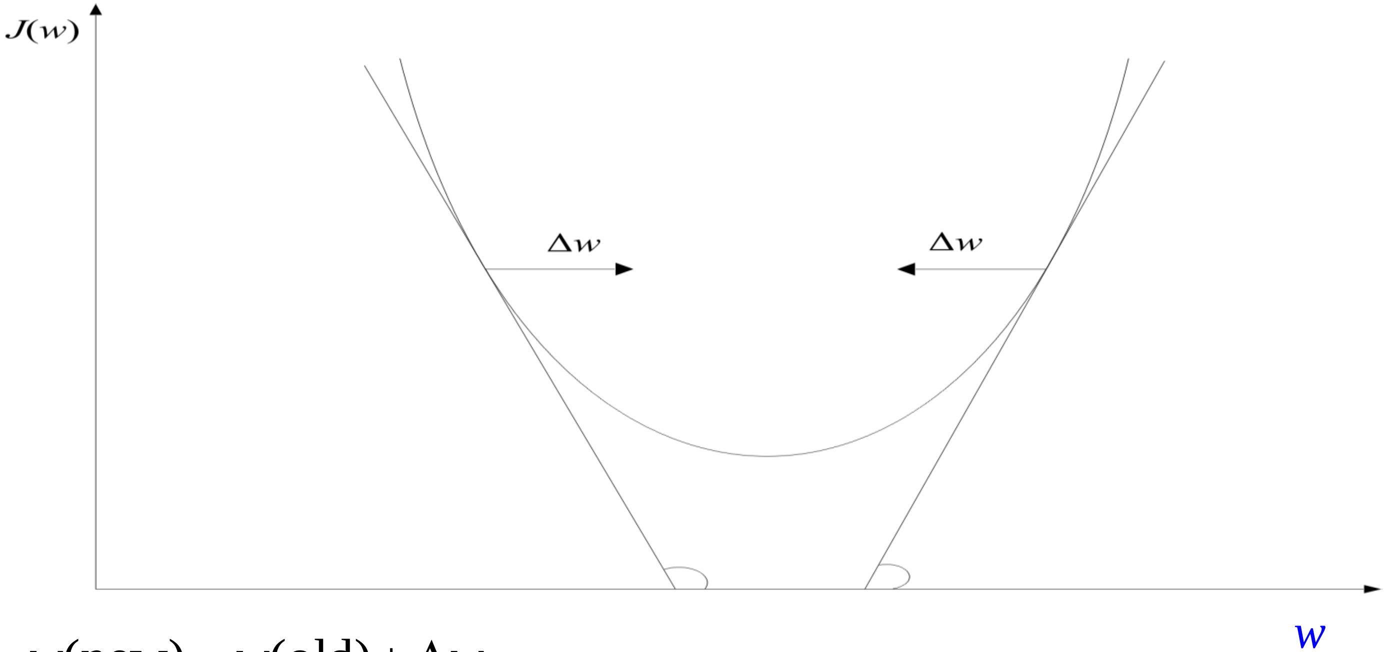
- $J(\underline{w})$ is piecewise linear (WHY?)

– The Algorithm
  - The philosophy of the gradient descent is adopted.



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

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

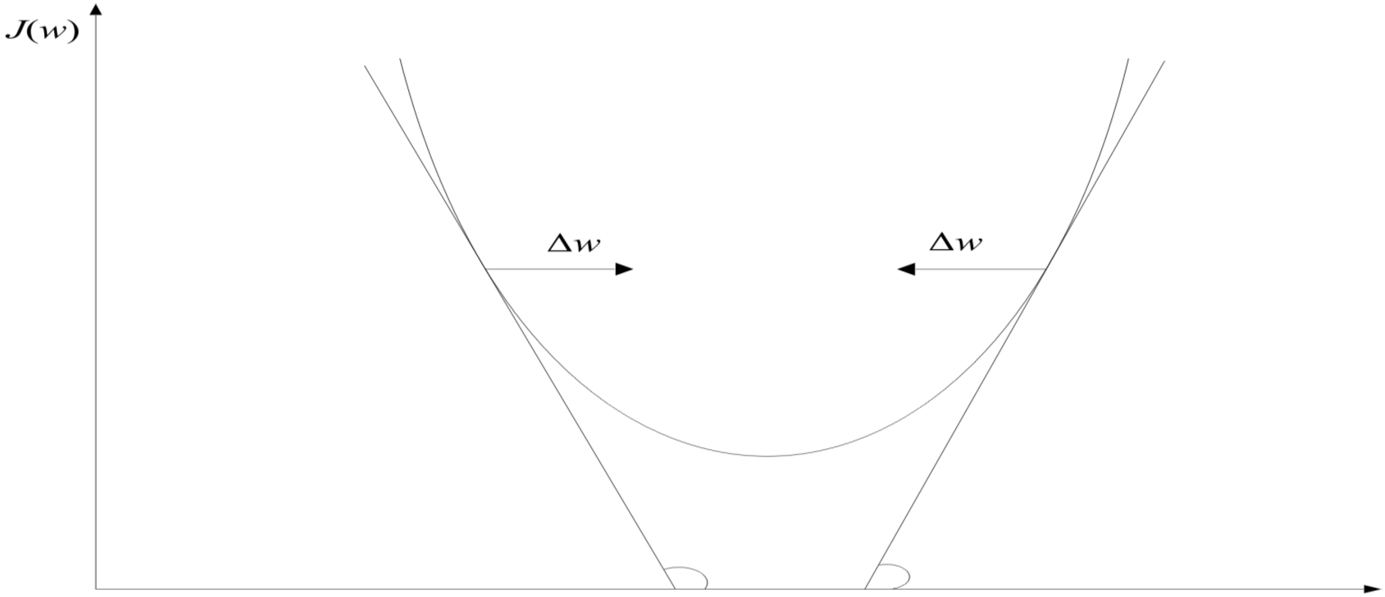


$$\underline{w}(\text{new}) = \underline{w}(\text{old}) + \Delta \underline{w}$$

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

- Wherever valid

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



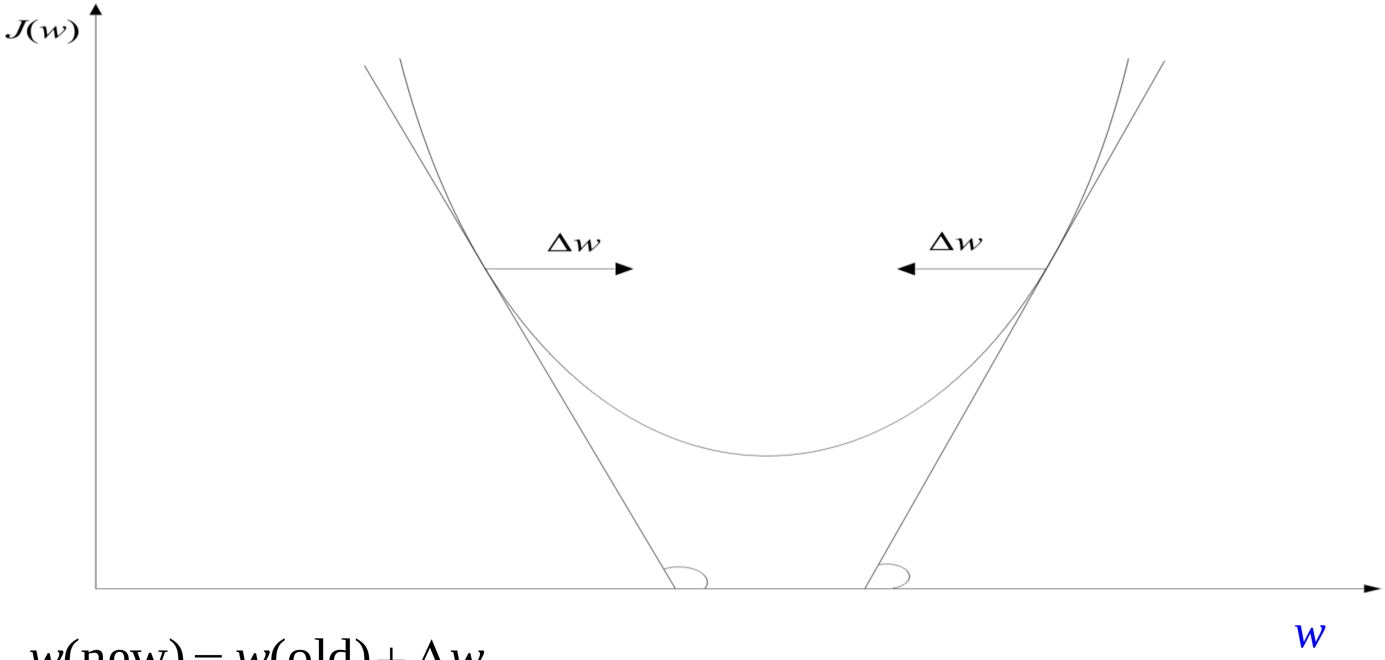
$$\underline{w}(\text{new}) = \underline{w}(\text{old}) + \Delta \underline{w}$$

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

- Wherever valid

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

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



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

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

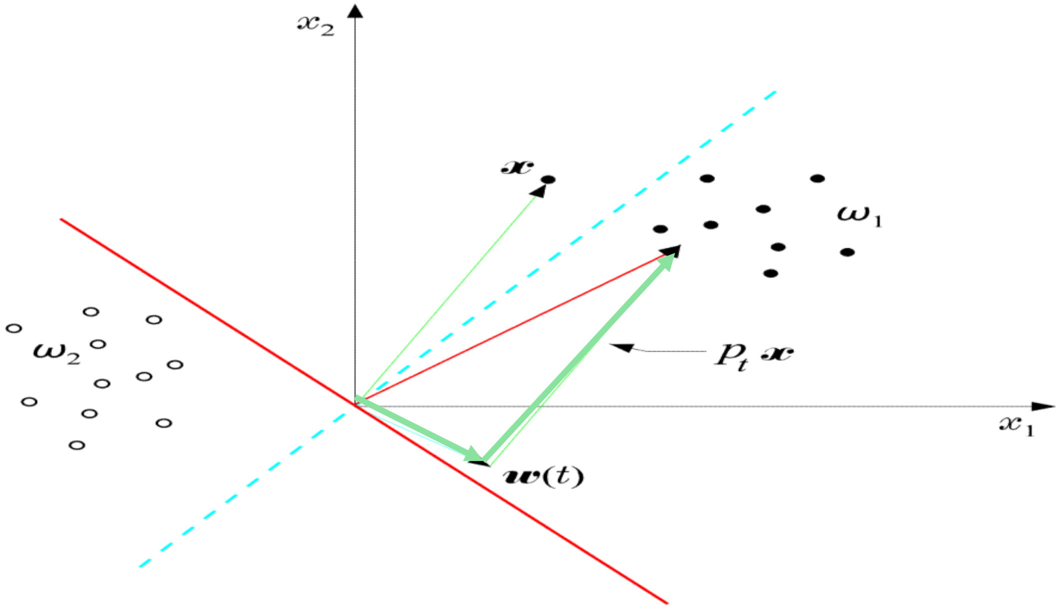
- Wherever valid

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

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

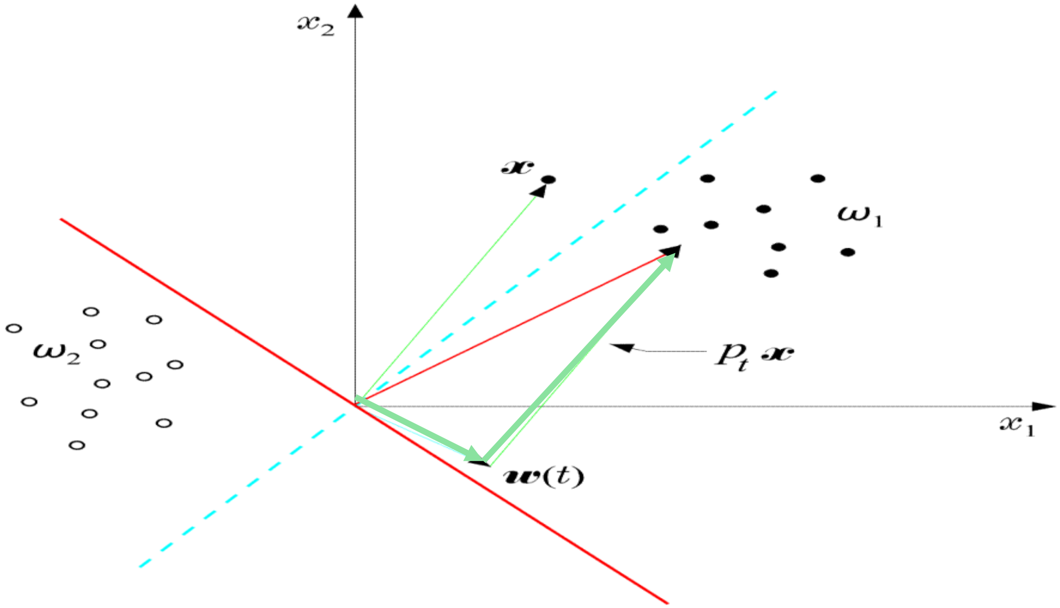
This is the celebrated Perceptron Algorithm

– An example:



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

– An example:



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

– The perceptron algorithm **converges** in a **finite** number of iteration steps to a solution **if patterns are linearly separable**

– Example: At some stage t the perceptron algorithm results in

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

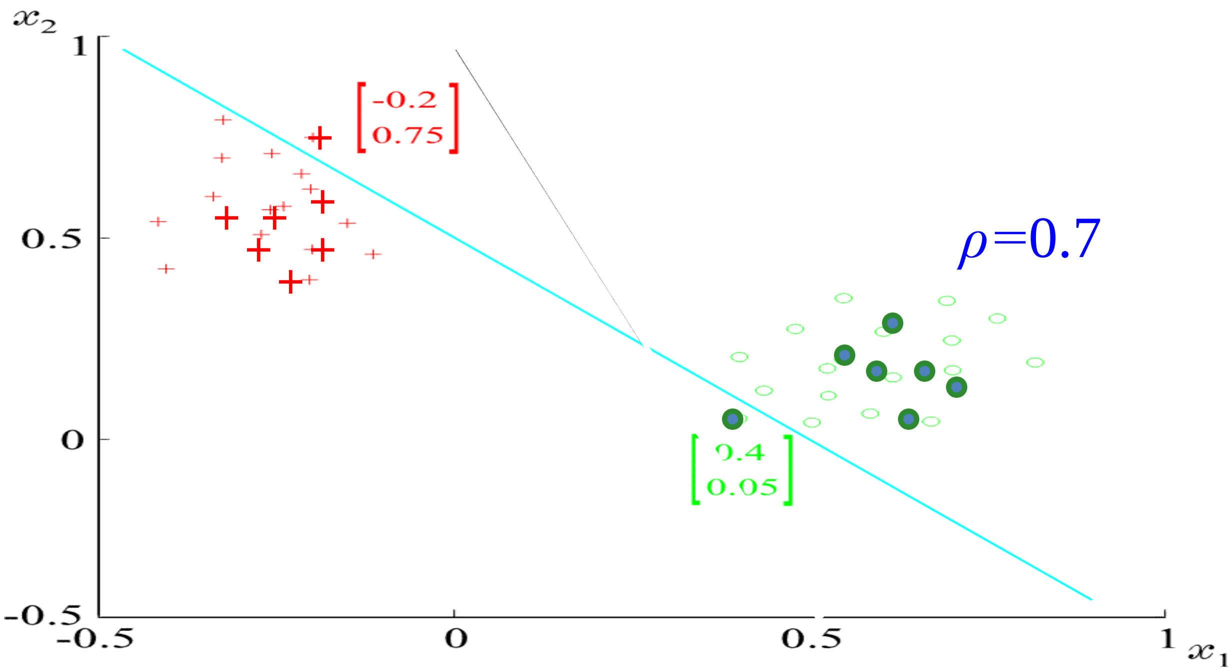
$$x_1 + x_2 - 0.5 = 0$$

– 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

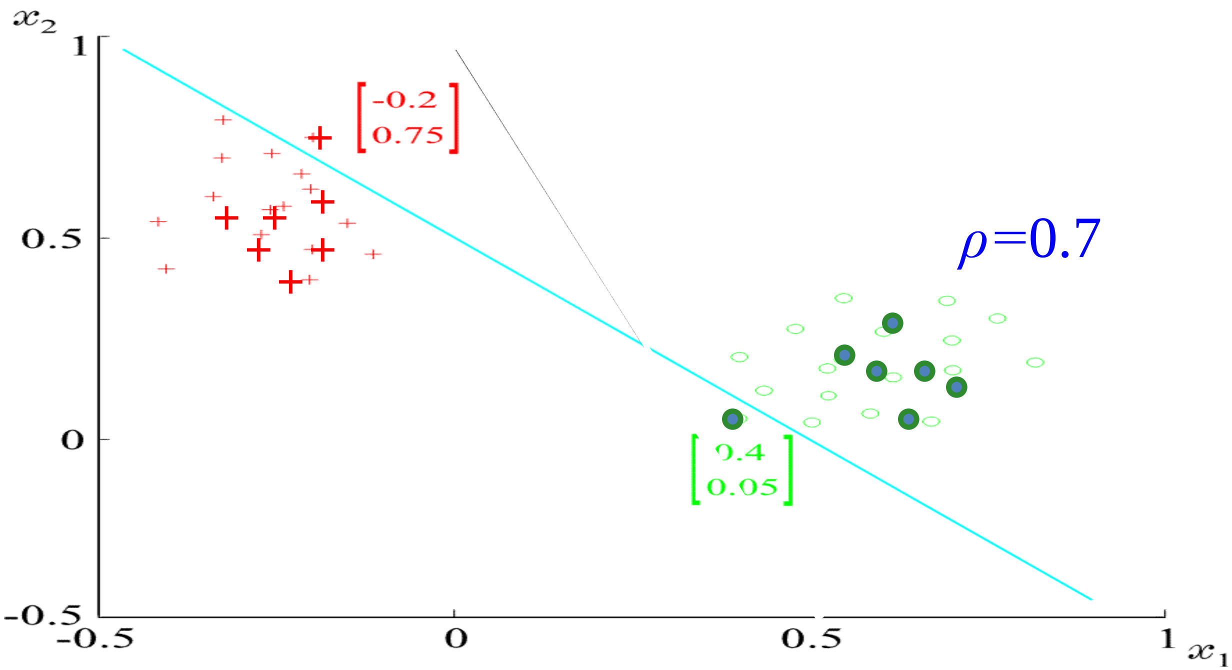


– Example: At some stage t the perceptron algorithm results in

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

$$x_1 + x_2 - 0.5 = 0$$

The corresponding hyperplane is



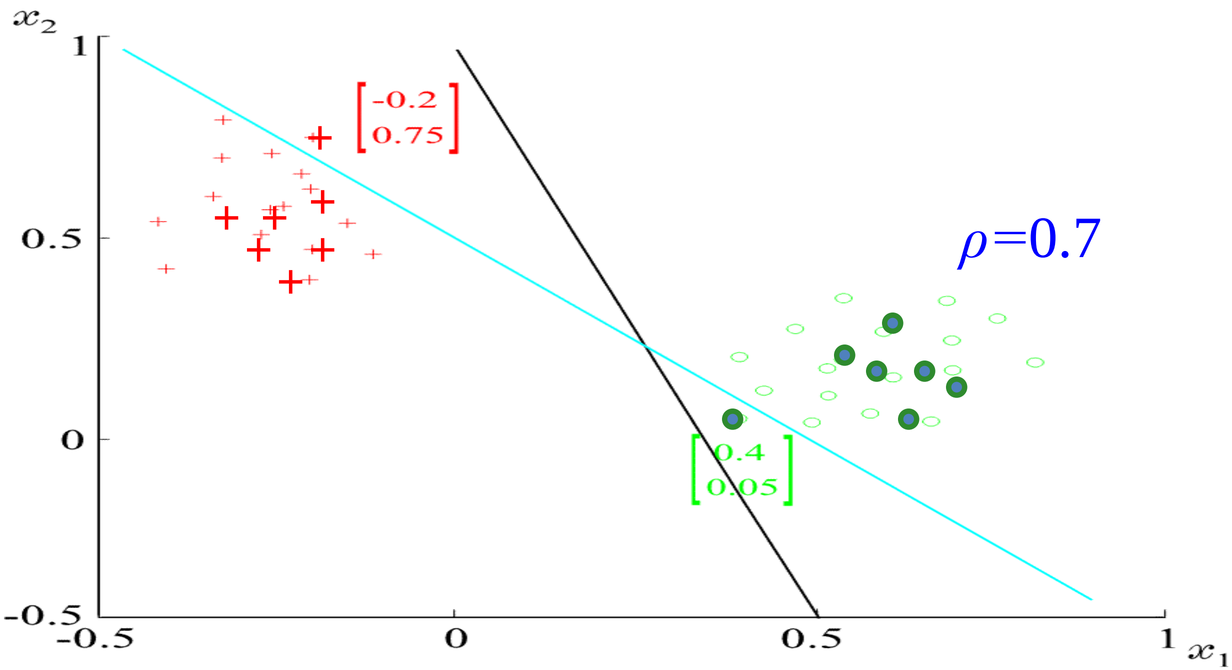
$$\underline{w}(t+1) = \begin{bmatrix}1\\1\\-0.5\end{bmatrix} - 0.7(-1)\begin{bmatrix}0.4\\0.05\\1\end{bmatrix} - 0.7(+1)\begin{bmatrix}-0.2\\0.75\\1\end{bmatrix} = \begin{bmatrix}1.42\\0.51\\-0.5\end{bmatrix}$$

– 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



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

Convergence Proof of Perceptron Algorithm

Let , $\underline{w}^*$ be the optimal weight vector

$\underline{w}(t)$ be the weight vector at the tth iteration

$\underline{w}(t+1)$ be the weight vector at the $(t+1)$th iteration

Convergence Proof of Perceptron Algorithm

Let , $\underline{w}^*$ be the optimal weight vector

$\underline{w}(t)$ be the weight vector at the tth iteration

$\underline{w}(t+1)$ be the weight vector at the $(t+1)$th iteration

We will prove that $\|\underline{w}(t+1) - \underline{w}^*\| < \|\underline{w}(t) - \underline{w}^*\|$

Convergence Proof of Perceptron Algorithm

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

Let , α be a positive real number

Then, $$w(t+1) - \alpha w^* = w(t) - \alpha w^* - \rho_t \sum_{x \in Y} \delta_x x$$

Convergence Proof of Perceptron Algorithm

$$\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^* = \boldsymbol{w}(t) - \alpha \boldsymbol{w}^* - \rho_t \sum_{x \in Y} \delta_x \boldsymbol{x}$$

Squaring both sides,

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 = \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \|\sum_{x \in Y} \delta_x \boldsymbol{x}\|^2 - 2\rho_t \sum_{x \in Y} \delta_x (\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*)^T \boldsymbol{x}$$

Convergence Proof of Perceptron Algorithm

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 = \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \|\sum_{x \in Y} \delta_x \boldsymbol{x}\|^2 - 2\rho_t \sum_{x \in Y} \delta_x (\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*)^T \boldsymbol{x}$$

However, $-\sum_{\boldsymbol{x} \in Y} \delta_x \boldsymbol{w}^T(t)\, \boldsymbol{x} < 0$

Convergence Proof of Perceptron Algorithm

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 = \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \|\sum_{x \in Y} \delta_x \boldsymbol{x}\|^2 - 2\rho_t \sum_{x \in Y} \delta_x (\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*)^T \boldsymbol{x}$$

However, $-\sum_{\boldsymbol{x} \in Y} \delta_x \boldsymbol{w}^T(t)\, \boldsymbol{x} < 0$

Hence,

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 \leq \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \|\sum_{x \in Y} \delta_x \boldsymbol{x}\|^2 + 2\rho_t \alpha \sum_{x \in Y} \delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$$

Convergence Proof of Perceptron Algorithm

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 \leq \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \|\sum_{x \in Y} \delta_x \boldsymbol{x}\|^2 + 2\rho_t \alpha \sum_{x \in Y} \delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$$

Now, define

$$\beta^2 = \max_{\widetilde{Y} \subseteq \omega_1 \cup \omega_2} \|\sum_{x \in \widetilde{Y}} \delta_x \boldsymbol{x}\|^2$$ and $$\gamma = \max_{\widetilde{Y} \subseteq \omega_1 \cup \omega_2} \sum_{x \in \widetilde{Y}} \delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$$

Convergence Proof of Perceptron Algorithm

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 \le \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \|\sum_{x \in Y} \delta_x \boldsymbol{x}\|^2 + 2\rho_t \alpha \sum_{x \in Y} \delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$$

Now, define

$$\beta^2 = \max_{\widetilde{Y} \subseteq \omega_1 \cup \omega_2} \|\sum_{x \in \widetilde{Y}} \delta_x \boldsymbol{x}\|^2 \quad \text{and} \quad \gamma = \max_{\widetilde{Y} \subseteq \omega_1 \cup \omega_2} \sum_{x \in \widetilde{Y}} \delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$$

Here, $\delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$ is always negative, so is γ

Convergence Proof of Perceptron Algorithm

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 \leq \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \|\sum_{x \in Y} \delta_x \boldsymbol{x}\|^2 + 2\rho_t \alpha \sum_{x \in Y} \delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$$

Now, define

$$\beta^2 = \max_{\tilde{Y} \subseteq \omega_1 \cup \omega_2} \|\sum_{x \in \tilde{Y}} \delta_x \boldsymbol{x}\|^2 \quad \text{and} \quad \gamma = \max_{\tilde{Y} \subseteq \omega_1 \cup \omega_2} \sum_{x \in \tilde{Y}} \delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$$

Here, $\delta_x \boldsymbol{w}^{*T} \boldsymbol{x}$ is always negative, so is γ

We can write,

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 \leq \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \beta^2 - 2\rho_t \alpha |\gamma|$$

Convergence Proof of Perceptron Algorithm

$$\|w(t+1) - \alpha w^*\|^2 \leq \|w(t) - \alpha w^*\|^2 + \rho_t^2 \beta^2 - 2\rho_t \alpha |\gamma|$$

If we choose, $\alpha = \dfrac{\beta^2}{2|\gamma|}$

We can write,

$$\|w(t+1) - \alpha w^*\|^2 \leq \|w(t) - \alpha w^*\|^2 + \rho_t^2 \beta^2 - \rho_t \beta^2$$

Convergence Proof of Perceptron Algorithm

$$\|\boldsymbol{w}(t+1) - \alpha \boldsymbol{w}^*\|^2 \leq \|\boldsymbol{w}(t) - \alpha \boldsymbol{w}^*\|^2 + \rho_t^2 \beta^2 - \quad \rho_t \, \beta^2$$

Here, $\rho_t^2 \beta^2 - \rho_t \beta^2 < 0$

How?

Convergence Proof of Perceptron Algorithm

$$\|w(t+1) - \alpha w^*\|^2 \leq \|w(t) - \alpha w^*\|^2 + \rho_t^2 \beta^2 - \rho_t \beta^2$$

Applying the above equation successively for steps *t*, *t*-1, . . ., 0, we get

$$\|w(t+1) - \alpha w^*\|^2 \leq \|w(0) - \alpha w^*\|^2 + \beta^2 \left(\sum_{k=0}^{t} \rho_k^2 - \sum_{k=0}^{t} \rho_k \right)$$

Convergence Proof of Perceptron Algorithm

$$\|w(t+1) - \alpha w^*\|^2 \leq \|w(t) - \alpha w^*\|^2 + \rho_t^2 \beta^2 - \quad \rho_t \, \beta^2$$

Applying the above equation successively for steps *t*, *t*-1, . . ., 0, we get

$$\|w(t+1) - \alpha w^*\|^2 \leq \|w(0) - \alpha w^*\|^2 + \beta^2 \left(\sum_{k=0}^{t} \rho_k^2 - \sum_{k=0}^{t} \rho_k \right)$$

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

Convergence Proof of Perceptron Algorithm

$$\|w(t+1) - \alpha w^*\|^2 \le \|w(0) - \alpha w^*\|^2 + \beta^2 \left(\sum_{k=0}^{t} \rho_k^2 - \sum_{k=0}^{t} \rho_k \right)$$

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

This means,
After some constant tine t_0 the R. H. S. will be non-positive

But, the L. H. S. cannot be negative

Therefore, $0 \le \|w(t_0 + 1) - \alpha w^*\| \le 0$

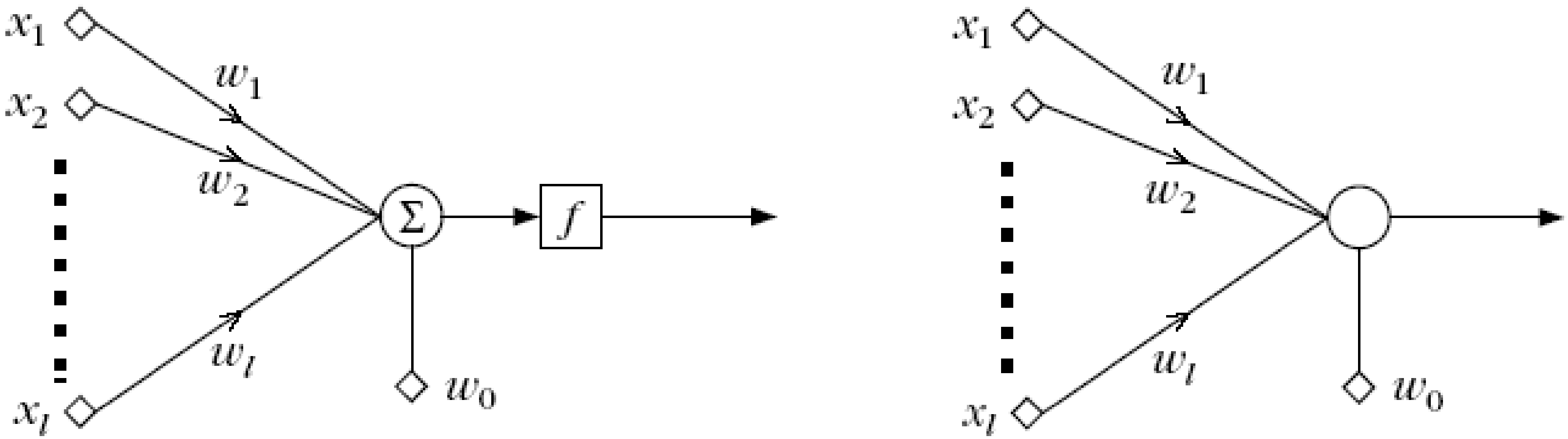
Convergence Proof of Perceptron Algorithm

$$0 \le \|w(t_0 + 1) - \alpha w^*\| \le 0$$

is equivalent to

$$w(t_0 + 1) = \alpha w^*$$

The Perceptron



$w_i' s$ synapses or synaptic weights

w_0 threshold

- This structure is called perceptron or neuron
- a learning machine that learns from the training vectors

Variants of Perceptron Algorithm (1)

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

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

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

Variants of Perceptron Algorithm (1)

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

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

update

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

No Update

Variants of Perceptron Algorithm (1)

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

update

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

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

No Update

– It is a $\boxed{\text{reward and punishment}}$ type of algorithm

Variants of Perceptron Algorithm (2)

- initialize weight vector $\mathbf{w}(0)$
- define pocket $\mathbf{w}_s$.and history h_s
- generate next $\mathbf{w}(t+1)$. If it is better than $\mathbf{w}(t)$, store $\mathbf{w}(t+1)$ in $\mathbf{w}_s$ and change the h_s

Variants of Perceptron Algorithm (2)

- initialize weight vector $\mathbf{w}(0)$
- define pocket $\mathbf{w}_s$.and history h_s
- generate next $\mathbf{w}(t+1)$. If it is better than $\mathbf{w}(t)$, store $\mathbf{w}(t+1)$ in $\mathbf{w}_s$ and change the h_s

  – It is $\boxed{\text{pocket}}$ algorithm

Generalization of Perceptron Algorithm for M- Class case

Generalization of Perceptron Algorithm for *M*- Class case

- Let M classes $\omega_{1,}\ \omega_{2,}\ \omega_{3,}\ldots,\ \omega_{M,}$
- Let M linear discriminant functions, $\underline{w}_i$

Generalization of Perceptron Algorithm for M- Class case

- Let M classes $\omega_{1,}$ $\omega_{2,}$ $\omega_{3,}$ . . ., $\omega_{M,}$
- Let M linear discriminant functions, $\underline{w}_i$
- The object $\underline{x}$ is classified to ω_i, if

$$\boldsymbol{w}_i^T \boldsymbol{x} > \boldsymbol{w}_j^T \boldsymbol{x}, \quad \forall j \neq i$$

Generalization of Perceptron Algorithm for M- Class case

- The object $\underline{x}$ is classified to ω_i, if

$$\boldsymbol{w}_i^T \boldsymbol{x} > \boldsymbol{w}_j^T \boldsymbol{x}, \quad \forall j \neq i$$

$$[w_i^T - w_j^T].x > 0$$

Generalization of Perceptron Algorithm for M- Class case

- The object $\underline{x}$ is classified to ω_i, if

$$\boldsymbol{w}_i^T \boldsymbol{x} > \boldsymbol{w}_j^T \boldsymbol{x}, \quad \forall j \neq i$$

$$[w_i^T - w_j^T].x > 0$$

can be written as

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, -w_j^T \cdots, 0^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, x^T \cdots, 0^T]^T > 0$$

Generalization of Perceptron Algorithm for M- Class case

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, -w_j^T \cdots, 0^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, x^T \cdots, 0^T]^T > 0$$

Generalization of Perceptron Algorithm for M- Class case

$$[0^T,\cdots,0^T,w_i^T,\cdots,0^T,-w_j^T\cdots,0^T].$$

$$[0^T,\cdots,0^T,x^T,\cdots,0^T,x^T\cdots,0^T]^T>0$$

$$[0^T,\cdots,0^T,w_i^T,\cdots,0^T,w_j^T\cdots,0^T].$$

$$[0^T,\cdots,0^T,x^T,\cdots,0^T,-x^T\cdots,0^T]^T>0$$

Generalization of Perceptron Algorithm for M- Class case

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, \boxed{-w_j^T} \cdots, 0^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, \boxed{x^T} \cdots, 0^T]^T > 0$$

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, \boxed{w_j^T} \cdots, 0^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, \boxed{-x^T} \cdots, 0^T]^T > 0$$

Generalization of Perceptron Algorithm for M- Class case

$$[w_i^T - w_j^T].x > 0$$

$$[0^T, \cdots, 0^T, w_i^T, \cdots, 0^T, w_j^T \cdots, 0^T].$$
$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T > 0$$

$$[w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T].$$
$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T > 0$$

Generalization of Perceptron Algorithm for M- Class case

$$[w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T].$$

$$[0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T > 0$$

Let, $w = [w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T]^T$

and $x_{i,j} = [0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T$

Then, the condition is $w^T x_{i,j} > 0$

Generalization of Perceptron Algorithm for M- Class case

- For each training vector of class ω_i, construct

$$x_{i,j} = [\underbrace{0^T, \cdots, 0^T, \underset{i\text{th location}}{x^T}, \cdots, 0^T, \underset{j\text{th location}}{-x^T} \cdots, 0^T}_{(l+1)M \text{ dimension}}]^T$$

- Concatenate the weight vectors:

$$w = [w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T]^T$$

Generalization of Perceptron Algorithm for M- Class case

$$x_{i,j} = [0^T, \cdots, 0^T, x^T, \cdots, 0^T, -x^T \cdots, 0^T]^T$$

$$w = [w_1^T, w_2^T \cdots, w_i^T, \cdots, w_j^T \cdots, w_M^T]^T$$

- Use a single Perceptron to solve
- Parameters:
  - $(l+1)M$ feature dimension
  - All $N(M\text{-}1)$ training vectors to be on positive side
- This reorganization is known as Kesler's construction