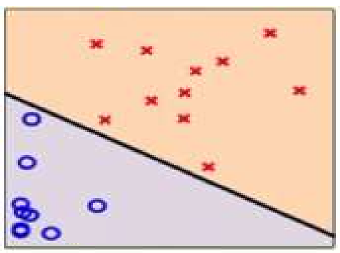
Classification

A_α: Binary Classification

Motivation

Data

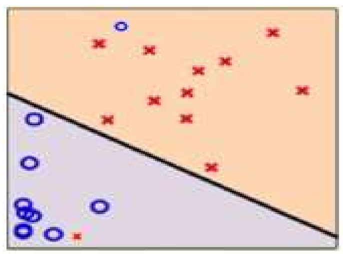
Linearly separable



Without noise

Perceptron

PLA



With noise

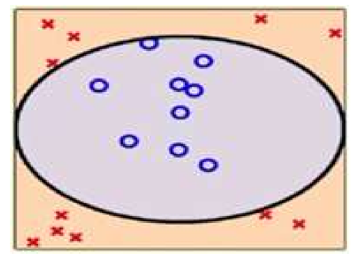
Adaline

Delta rule

Perceptron

Pocket

Nonlinearly separable



Nonlinear Transformation

Learning Model A_α : Perceptron Learning Algorithm

Let the Training Data $\boldsymbol{S} = \{(\boldsymbol{x_1}, \boldsymbol{y_1}), \dots, (\boldsymbol{x_m}, \boldsymbol{y_m})\}$, $\boldsymbol{x_i} \in \mathbb{R}^d$, d is the dimension of the input space

Purpose:

Find a classifier $\boldsymbol{h_S}(\boldsymbol{x_i}) = \boldsymbol{y_i} \in \{-\mathbf{1}, +\mathbf{1}\}$ ***such that is the sign of hyperplan*** $h_{w,b}(\boldsymbol{x}) = \boldsymbol{w^T x} + \boldsymbol{b}$

- $\boldsymbol{w} \in \mathbb{R}^d, \boldsymbol{b} \in \mathbb{R}$
- $w^T x + b = w^T x$ ***such that:*** $x = (\mathbf{1}, \boldsymbol{x}) \in \mathbb{R}^{d+1}, \boldsymbol{w} = (\boldsymbol{b}, \boldsymbol{w}) \in \mathbb{R}^{d+1}, w_0 = b \Longrightarrow h_w(x) = w^T x$

The perceptron hypothesis is: $H = \left\{h_{w,b}, \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\right\} = \{h_w, w \in \boldsymbol{w} \in \mathbb{R}^{d+1}\}$

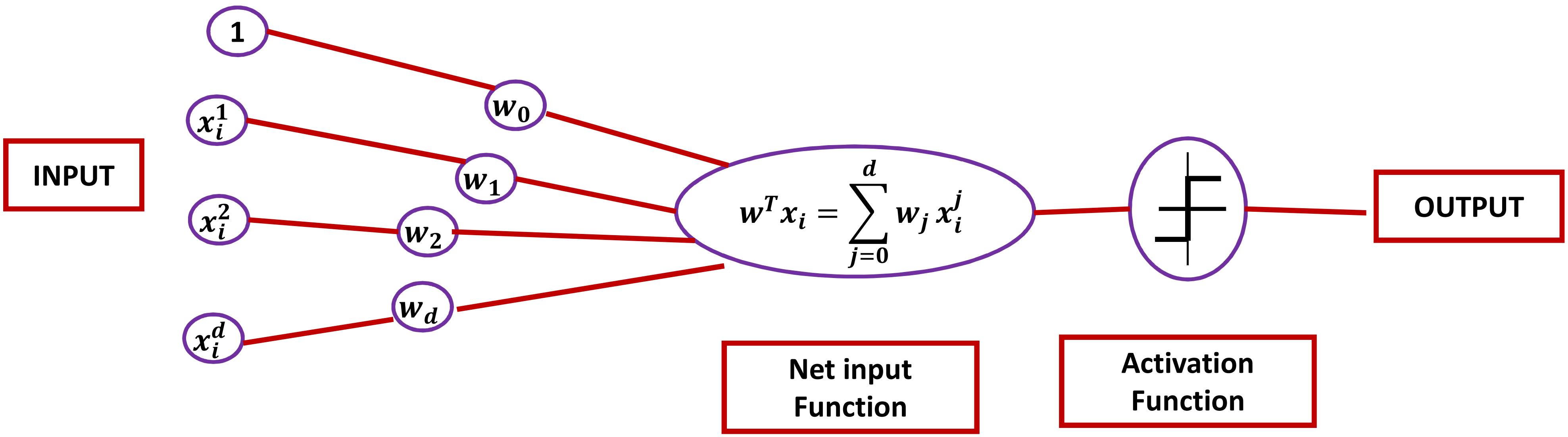
$$\boldsymbol{h_S}(\boldsymbol{x_i}) = \begin{cases} +1 & si \quad w^T x > 0 \\ -1 & si \quad w^T x < 0 \end{cases} \text{où } w \in R^{d+1}, \boldsymbol{x_i} = (1, x_i^1, \dots, x_i^d) \in \{1\} \times \mathbb{R}^d$$

$$\boldsymbol{h_S}(\boldsymbol{x_i}) = sign\big(h_w(\boldsymbol{x_i})\big) = sign(w^T \boldsymbol{x_i}) = \begin{cases} +1 & si \quad w^T \boldsymbol{x_i} > 0 \\ -1 & si \quad w^T \boldsymbol{x_i} < 0 \end{cases} \text{où } w \in \mathbb{R}^{d+1}$$

$$H = \{\boldsymbol{h_S}: S \to \{-\mathbf{1}, +\mathbf{1}\} | \boldsymbol{h_S}(\boldsymbol{x}) = sign(w^T \boldsymbol{x}), w \in \mathbb{R}^{d+1}: x \in S\} \Longrightarrow |H| = \infty$$

Learning Model A_α : Diagram of the Perceptron Learning Algorithm

- $x_i = (1, x_i^1, \dots, x_i^d) \in S \Longrightarrow w^T\ x_i = \sum_{j=0}^{d} w_j\, x_i^j \Longrightarrow x \in S\,, h_S(x) = sign(w^T x) \Longrightarrow output\ \ y \in \{-1,1\}$



Learning Model A_α : Perceptron Learning Algorithm

Best classifier: $y_i \in \{-1, +1\}$

Loss Function:

$$L_S(\boldsymbol{h_S}) = \frac{|\boldsymbol{x_i} \in S : \boldsymbol{h_S}(\boldsymbol{x_i}) \neq \boldsymbol{y_i}|}{|S|}$$

- $0 \leq L_S(\boldsymbol{h_S}) \leq 1$
- $\boldsymbol{h_S} = sign(h_w), \boldsymbol{w} \in \mathbb{R}^{d+1}$
- $L_S(\boldsymbol{h_S}) = L_S(w^T x) = L_S(w)$

Purpose:

$$\min_{w \in \mathbb{R}^{d+1}} L_S(w) \Longrightarrow w^* = \underset{w \in \mathbb{R}^{d+1}}{\operatorname{argmin}} L_S(w) \Longrightarrow L_S(w^*) = 0$$

- $L_S(w) = \frac{1}{n} \sum_{i=1}^{n} 1_{[w^T x_i \neq y_i]}$
- $1_{[w^T x_i \neq y_i]}(x_i) = \begin{cases} 1 \; si \;\; w^T x_i \neq y_i \\ 0 \;\; si \;\; w^T x_i = y_i \end{cases}$
- if $L_S(w) \neq 0 \; then \; \exists \boldsymbol{x_i} \in S \; such \; that \; w^T x_i \neq y_i \Leftrightarrow signe(w^T x_i . y_i) < 0$
- $\Longrightarrow w \leftarrow w + y_i x_i$

Learning Model A_α : Perceptron Learning Algorithm

We have two sets N and P:

$$\begin{cases} if\ x \in \boldsymbol{P} & \rightarrow \quad y = +1 \\ if\ x \in \boldsymbol{N} & \rightarrow \quad y = -1 \end{cases}$$

Objective:

We look for w capable of absolutely separating the two sets N and P:

$$P = \text{open positive half space}$$
$$\text{N} = \text{open negative half space}$$

To simplify the visualization of the algorithm, we are going to take $d = 2$.

So:

$$x = (x_1, x_2) \text{ et } \mathrm{w} = (w_1, w_2)$$

$$h(x) = \langle w, x \rangle = w_1 x_1 + w_2 x_2$$

Learning Model A_α : Perceptron Learning Algorithm

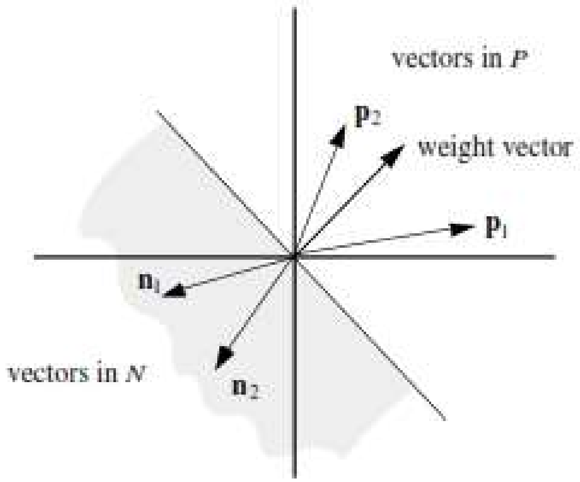
Notice that:

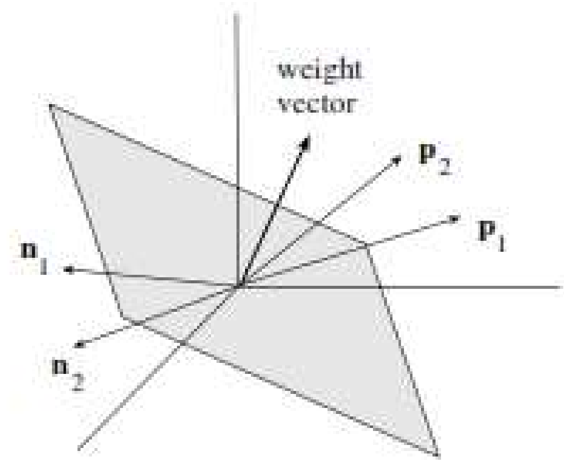
$$w_1 x_1 + w_2 x_2 = 0$$

Is the equation of a plane. And the vector normal to this plane is the weight vector $\mathrm{w} = (w_1, w_2)$.

We can visualize the linear representation in two different spaces:

- Input space: $x = (x_1, x_2)$ etr $\mathrm{w} = (w_1, w_2)$
- Extended input space: $x = (1, x_1, x_2)$ et $\mathrm{w} = (w_0, w_1, w_2)$





Learning Model A_α : Perceptron Learning Algorithm

Perceptron learning algorithm for Linearly separable

Input: $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ and w^0.
Output: w^*, t and $L_S(w^*)$
Start: $w \leftarrow w^0$ and t $\leftarrow 0$

Compute: $L_S(w) = \frac{1}{n}\sum_{i=1}^{n} 1_{[w^T x_i \neq y_i]}$

While $(L_S(w)! = 0)$:
  for $i = 1, \dots, n$:
    if $signe(w^T x_i).\, y_i < 0$
      $w \leftarrow w + y_i x_i$
      t$\leftarrow t + 1$
    endif
  endfor
  compute $L_S(w)$
endWhile

Return $w^* \leftarrow$ w, $L_S(w^*)$ and t.

end

Learning Model A_α : Perceptron Learning Algorithm

Objective: Reformulation

We should have that:

$$\begin{cases} \forall x \in P \ , & \langle w, x \rangle \geq 0 \\ \forall x \in \mathrm{N} \ , & \langle w, x \rangle < 0 \end{cases}$$

We know that:

$$\langle w, x \rangle = \|w\|\|x\| \cos(w, x) = \|w\|\|x\| \cos(\alpha) \Longrightarrow \cos(\alpha) = \frac{\langle w, x \rangle}{\|w\|.\|x\|} \Longrightarrow \alpha = \arccos\left(\frac{\langle w, x \rangle}{\|w\|.\|x\|}\right)$$

- $if\ \langle w, x \rangle < 0 \ \Longrightarrow \cos(\alpha) < 0 \Longrightarrow \alpha \in]\frac{\pi}{2} + 2k\pi, \frac{3\pi}{2} + 2k\pi[, \ (k \in \mathbb{Z})$
- $if\ \langle w, x \rangle \geq 0 \ \Longrightarrow \cos(\alpha) \geq 0 \Longrightarrow \alpha \in [\frac{-\pi}{2} + 2k\pi, \frac{\pi}{2} + 2k\pi], \ (k \in \mathbb{Z})$

We are going to deal with angles within the range $[0, \pi]$.

- $if\ \langle w, x \rangle < 0 \ \Longrightarrow \alpha > \frac{\pi}{2}$
- $if\ \langle w, x \rangle \geq 0 \ \Longrightarrow \alpha \leq \frac{\pi}{2}$

Learning Model A_α : Perceptron Learning Algorithm

Notice that:

$$if\ \langle w,x\rangle < 0 \ \Longrightarrow \cos(\alpha) < 0 \Longrightarrow \alpha \in]\frac{\pi}{2} + 2k\pi, \frac{3\pi}{2} + 2k\pi[$$

$$if\ \langle w,x\rangle \geq 0 \ \Longrightarrow \cos(\alpha) \geq 0 \Longrightarrow \alpha \in [\frac{-\pi}{2} + 2k\pi, \frac{\pi}{2} + 2k\pi]$$

$$(k \in \mathbb{Z})$$

We are going to deal with angles within the range $[0, \pi]$.

$$\begin{cases} if\ \langle w,x\rangle < 0 \Longrightarrow \alpha > \dfrac{\pi}{2} \\ if\ \langle w,x\rangle \geq 0 \Longrightarrow \alpha \leq \dfrac{\pi}{2} \end{cases}$$

Learning Model A_α : Perceptron Learning Algorithm:

$\boldsymbol{signe(w^T x_i). y_i}$ **and** $\boldsymbol{w \leftarrow w + y_i x_i}$

- If $x \in P(y = 1) and \langle w, x\rangle < 0 \Rightarrow$ we should rotate w near to x so that $\alpha \leq 90°$, this is can be done by adding x to w:

$$w_{new} \leftarrow w + x$$

Here:

$$\alpha_{new} < \alpha$$

- If $x \in N(y = -1) and \langle w, x\rangle \geq 0 \Rightarrow$ we should rotate w away from x so that $\alpha > 90°$, this is can be done by substructing x from w:

$$w_{new} \leftarrow w - x$$

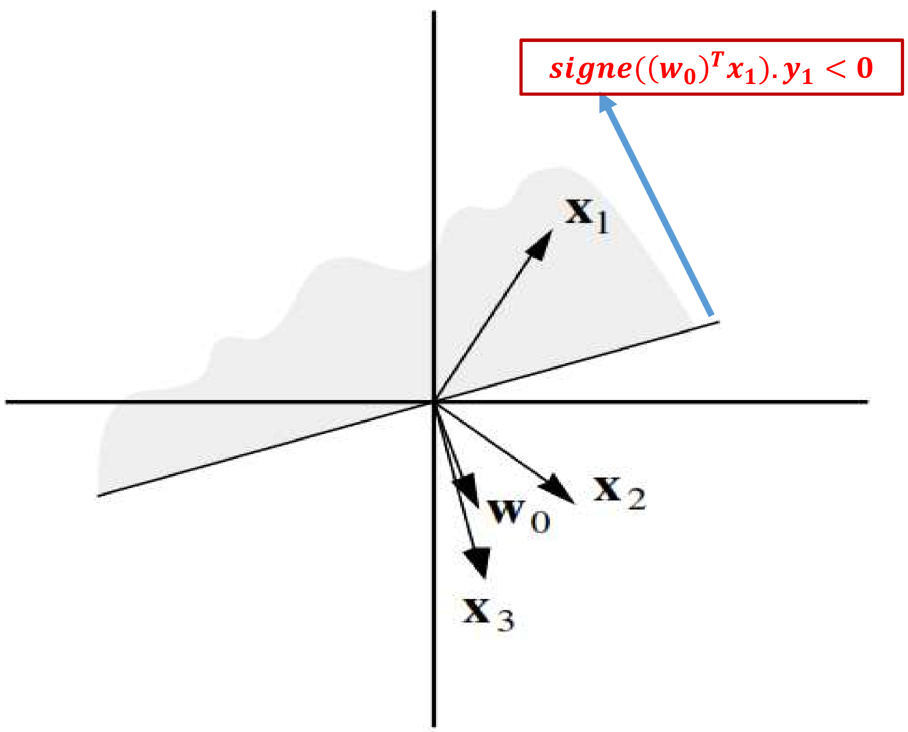
Here:

$$\alpha_{new} > \alpha$$

Learning Model A_α : Perceptron Learning Algorithm

Geometric Visualization

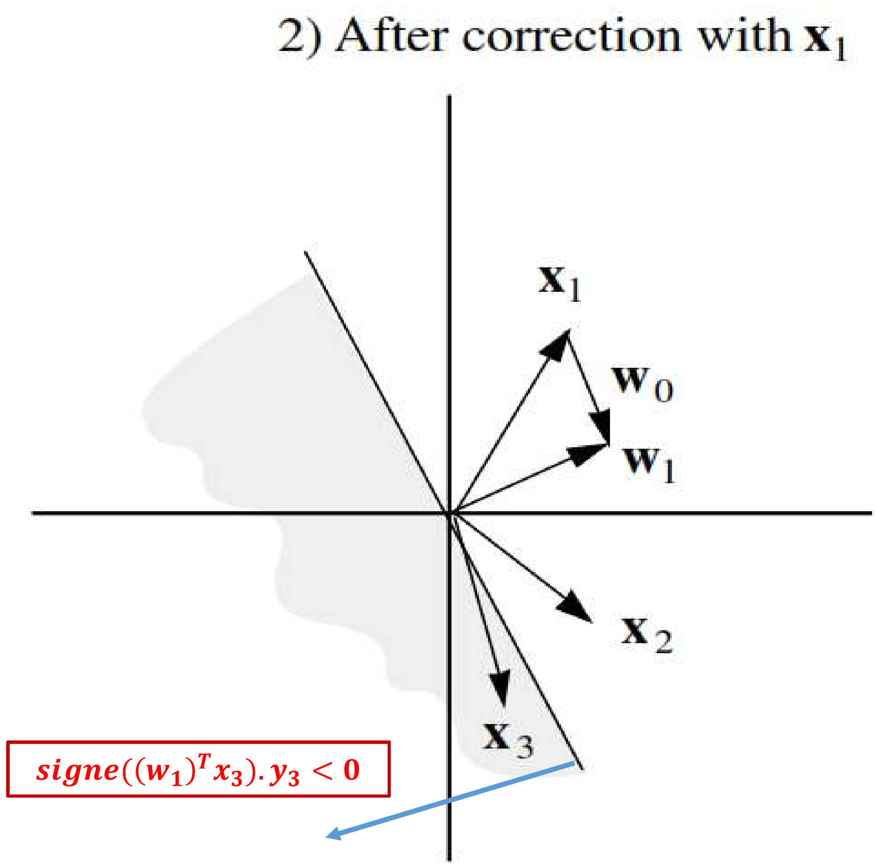
1) Initial configuration



- Data=$\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$
- $y_i = 1, i = 1, 2, 3$
- $L_S(w) = \frac{1}{3}\sum_{i=1}^{3} 1_{[w^T x_i \neq y_i]} = \frac{1}{3} \neq 0$
- $signe((w_0)^T x_1). y_1 < 0$
- $\langle w_0, x_1 \rangle < 0 \Longrightarrow \alpha > \frac{\pi}{2}$
- $y_1 = 1$
- $w_1 \leftarrow w_0 + y_1 x_1$

Learning Model A_α : Perceptron Learning Algorithm

Geometric Visualization:

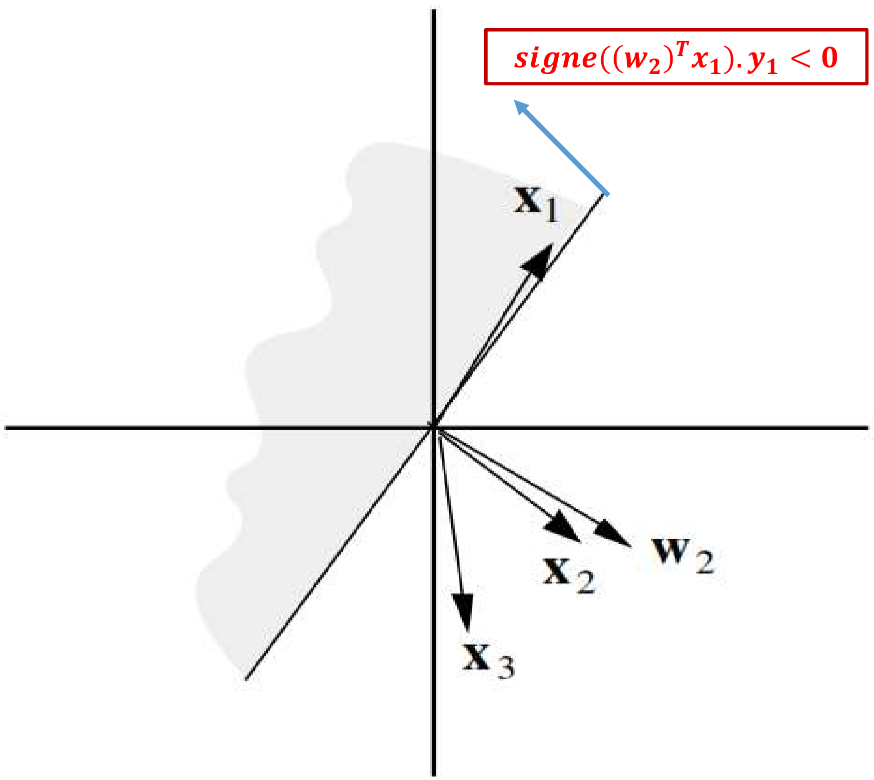


- $L_S(w) = \frac{1}{3}\sum_{i=1}^{3} 1_{[w^T x_i \neq y_i]} = \frac{1}{3} \neq 0$
- $\boldsymbol{signe((w_1)^T x_3).y_3 < 0}$
- $\boldsymbol{\langle w_1, x_3 \rangle < 0 \Rightarrow \alpha > \frac{\pi}{2}}$
- $\boldsymbol{y_3 = 1}$
- $\boldsymbol{w_2 \leftarrow w_1 + y_3 x_3}$

Learning Model A_α : Perceptron Learning Algorithm

Geometric Visualization:

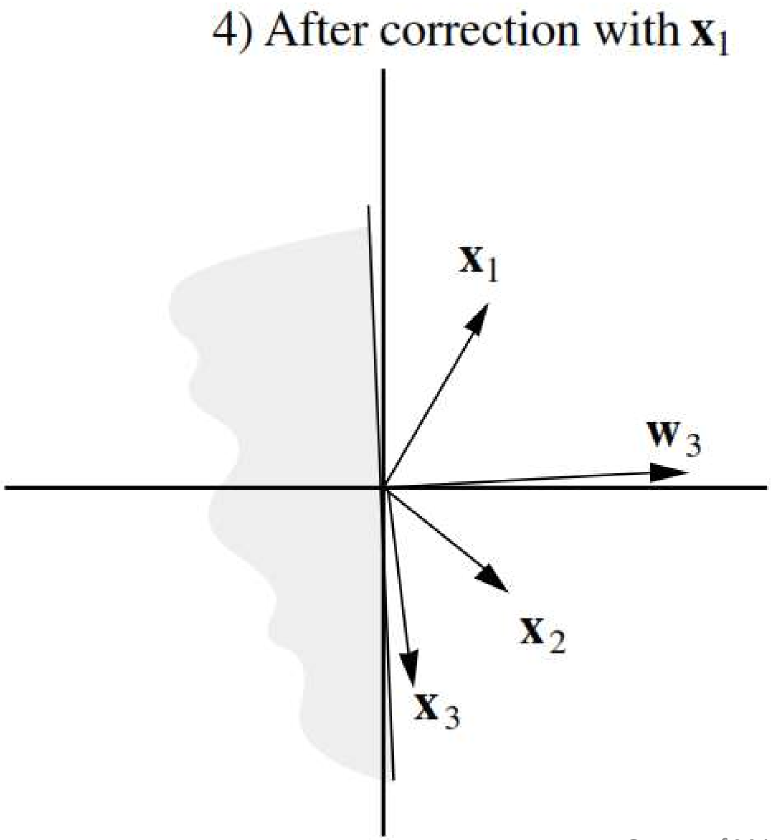
3) After correction with $\mathbf{x}_3$



- $L_S(w) = \frac{1}{3}\sum_{i=1}^{3} 1_{[w^T x_i \neq y_i]} = \frac{1}{3} \neq 0$
- $\boldsymbol{signe((w_2)^T x_1).y_1 < 0}$
- $\boldsymbol{\langle w_2, x_1 \rangle < 0 \Rightarrow \alpha > \frac{\pi}{2}}$
- $\boldsymbol{y_1 = 1}$
- $\boldsymbol{w_3 \leftarrow w_2 + y_1 x_1}$

Learning Model A_α : Perceptron Learning Algorithm

Geometric Visualization:



- $L_S(w) = \frac{1}{3}\sum_{i=1}^{3} 1_{[w^T x_i \neq y_i]} = 0$
- ***Stop***

Learning Model A_α : Perceptron Learning Algorithm

Pocket learning algorithm for Linearly separable with noise $A_{\alpha=(w_0,T_{max})}$

Input: $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ and w_0.

Output: w^*, t and $L_S(w^*)$

Start: $w(0) \leftarrow w_0$

Initialize the weight vector of pocket by the weight vector of PLA.

$$w_s \leftarrow w_0$$

for t = 1 , . . . , T_{max} :

Execute PLA for one weight update to obtain w(t):

PLA: for $i = 1, \dots, n$:

if $signe(w^T x_i).y_i < 0$

$w_s \leftarrow w_s + y_i x_i$

t$\leftarrow t + 1$

End for $w(t)$

Evaluate $L_S(w(t)) = \frac{1}{n}\sum_{i=1}^{n} 1_{[w(t)^T x_i \neq y_i]}$

if $\mathrm{L_S}\big(w(t)\big) < \mathrm{L_S}(w_s)$: $w_s \leftarrow w(t)$

endif

Return $\mathrm{w}^* \leftarrow w_s$, t and $L_S(w^*)$

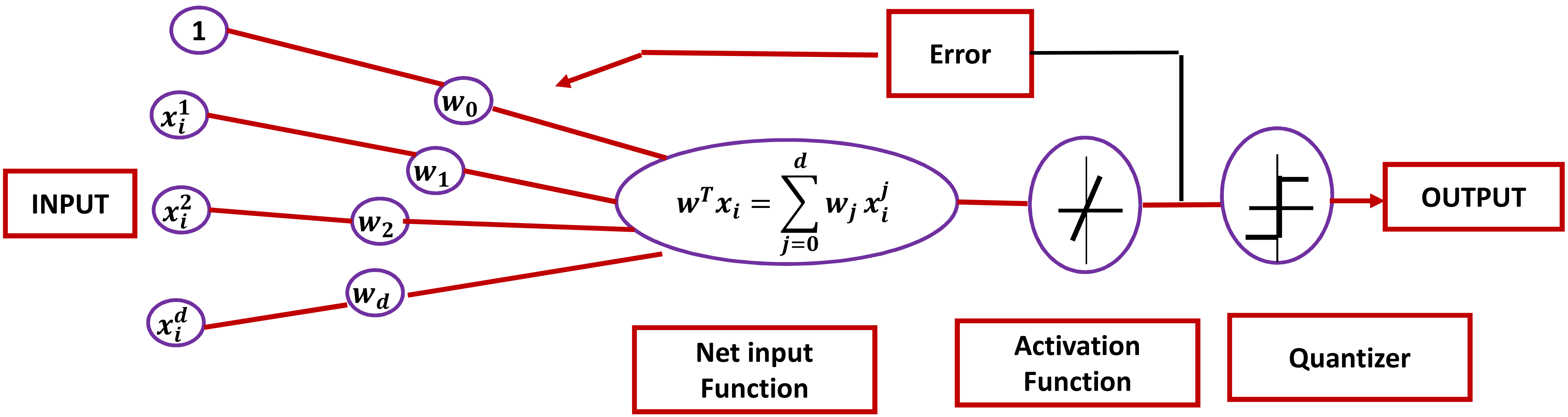
endfor

end

Learning Model A_α : Diagram of the Perceptron Learning Algorithm

Adaline Learning Algorithm for Linearly separable with noise

- $x_i = \left(1, x_i^1, \dots, x_i^d\right) \in S \Longrightarrow w^T\ x_i = \sum_{j=0}^{d} w_j\, x_i^j \ \Longrightarrow x \in S\,, h_S(x) = \ sign(w^T x) \Longrightarrow output\ \ y \in \{-1{,}1\}$



Learning Model A_α : Perceptron Learning Algorithm

Adaline Learning Algorithm for Linearly separable with noise

Adaline is an improvement of perceptron model developped in 1960 by Widrow and Hoff.

Adaline owns two hypotheses.

- During the training:

$$\boldsymbol{h_1}(\boldsymbol{x}) = \hat{\boldsymbol{y}} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{d}} \boldsymbol{w_i x_i} = \boldsymbol{w^T x}$$

- After the training:

$$\boldsymbol{h_2}(\boldsymbol{x}) = \boldsymbol{sign}(\boldsymbol{w^T x}) = \begin{cases} +1 \text{ si } \boldsymbol{w^T x} \geq \boldsymbol{0} \\ -1 \text{ si } \boldsymbol{w^T x} < \boldsymbol{0} \end{cases}$$

Empirical error:

$$\boldsymbol{L_S}(\boldsymbol{w}) = \frac{\boldsymbol{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{w^T x_i})^2 \text{ **MSE**}$$

Learning Model A_α : Perceptron Learning Algorithm

Delta rule learning algorithm: $L_S(w(t)) = \frac{1}{n}\sum_{i=1}^{n} 1_{[w(t)^T x_i \neq y_i]}$, $A_{\alpha=(w_0, T_{max})}$

Input: $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and w_0.
Output: w^*, t and $L_S(w^*)$
Start: $w \leftarrow w_0$
Compute: $L_S(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$
for t = 1 , . . . , T_{max}:
    for $i = 1, \ldots, n$:
      if $(e_i = \mathrm{y_i} - w^T x_i)\ ! = 0$
          $w \leftarrow w + 2.\, e_i.\, x_i$
      endif
    endfor
Endfor
Return $\mathrm{w}^* \leftarrow w$, t and $L_S(w^*)$
end

Learning Model A_{α} : Perceptron Learning Algorithm

Delta rule learning algorithm 2 $A_{\alpha=(w_0, T_{max}, \delta)}$

Input: $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and w_0, δ
Output: w^*, t and $L_S(w^*)$
Start: $w \leftarrow w_0$
Compute: $L_S(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$

While $\nabla_w L_S(w) > \delta$ do
  for $i = 1, \ldots, n$:
    if $(e_i = y_i - w^T x_i) != 0$
      $w \leftarrow w + 2. e_i . x_i$
    endif
  endfor

Return $w^* \leftarrow w$, t and $L_S(w^*)$
end

Learning Model A_α : Perceptron Learning Algorithm

Delta rule learning algorithm 3 $A_{\alpha=(w_0, T_{max}, \delta)}$

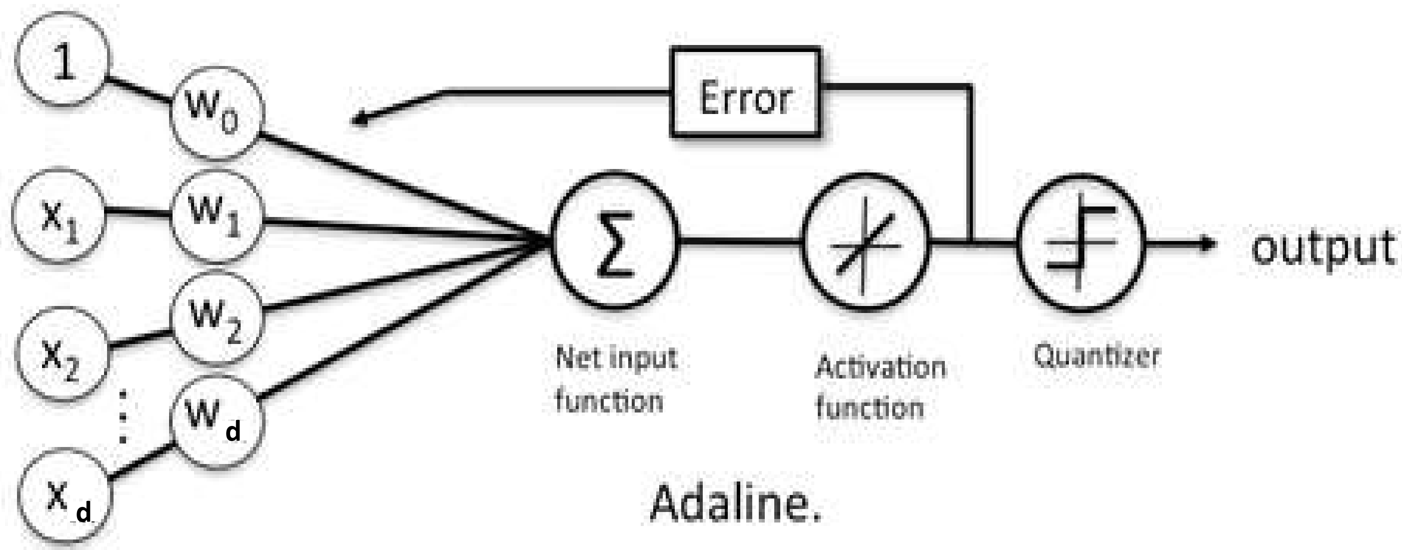
Input: $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ and w_0, δ
Output: w^*, t and $L_S(w^*)$
Start: $w \leftarrow w_0$
Compute: $L_S(w(t)) = \frac{1}{n}\sum_{i=1}^{n} 1_{[w(t)^T x_i \neq y_i]}$

While $\nabla_w L_S(w) > \delta$ do
    for $i = 1, \dots, n$:
      if $(e_i = \mathrm{y_i} - w^T x_i)\ != 0$
          $w \leftarrow w + subgradient$
      endif
    endfor

Return $\mathrm{w}^* \leftarrow w$, t and $L_S(w^*)$
end

Learning Algorithm: Adaline

- $\boldsymbol{h_2(x) = sign\left(w^T x\right) = sign(h_1(x) = \hat{y})}$
- $L_S(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$ et $e_i(w) = y_i - w^T x_i$
- If $e_i(w) = y_i - w^T x_i \begin{cases} = 0 \ \ classified \\ \neq 0 \ no \ classified \end{cases}$
- $\nabla_w L_S(w) = -\frac{1}{n}\sum_{i=1}^{n} 2x_i e_i(w)$
- $\nabla_w L_S(w) = 0 \iff \forall i, e_i(w) = 0$


1
w_0
Error
x_1
w_1
x_2
w_2
w_d
x_d
output
Net input
function
Activation
function
Quantizer

Adaline.