

SICO70

SISTEMAS INTELIGENTES 2

Aula 02 - Perceptron Simples

Prof. Rafael G. **Mantovani**

Roteiro



- 1** Introdução
- 2** Perceptron
- 3** Teorema de Convergência
- 4** Algoritmo de Treinamento para Perceptron
- 5** Exemplo / Exercício
- 6** Referências

Roteiro

1 Introdução

2 Perceptron

3 Teorema de Convergência

4 Algoritmo de Treinamento para Perceptron

5 Exemplo / Exercício

6 Referências

Introdução

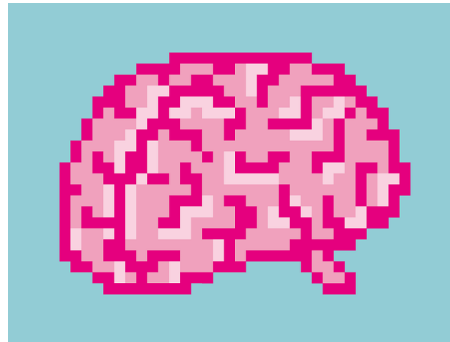


Introdução



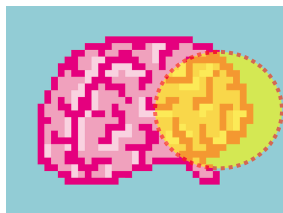
Relembrando nossa última aula ... :)

Introdução

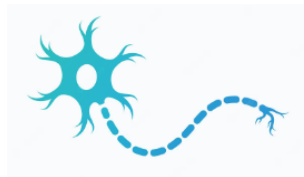


cérebro

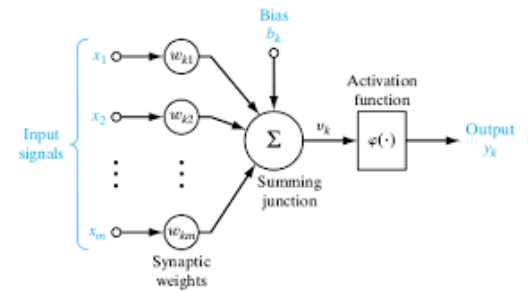
Introdução



Cérebro



**Neurônio
biológico**



**Neurônio
artificial**

Introdução



Introdução



Introdução

Perceptron

1958

Introdução

- primeira rede neural descrita algoritmicamente
- Frank Rosenblatt (psicólogo)
- modelo mais simples de rede neural que existe

Perceptron



Introdução

- primeira rede neural descrita algoritmicamente
- Frank Rosenblatt (psicólogo)
- modelo mais simples de rede neural que existe

Perceptron

1958

- Classifica padrões **linearmente separáveis**
- Possui um único neurônio com pesos sinápticos ajustáveis e **bias**

Introdução

- Rosenblatt definiu um algoritmo de treinamento:
 - onde ocorre o ajuste dos parâmetros livres da rede (pesos sinápticos - W)
 - provou que se os exemplos utilizados no treino forem linearmente separáveis, o algoritmo converge, posicionando um **hiperplano (reta)** entre as duas classes

Roteiro

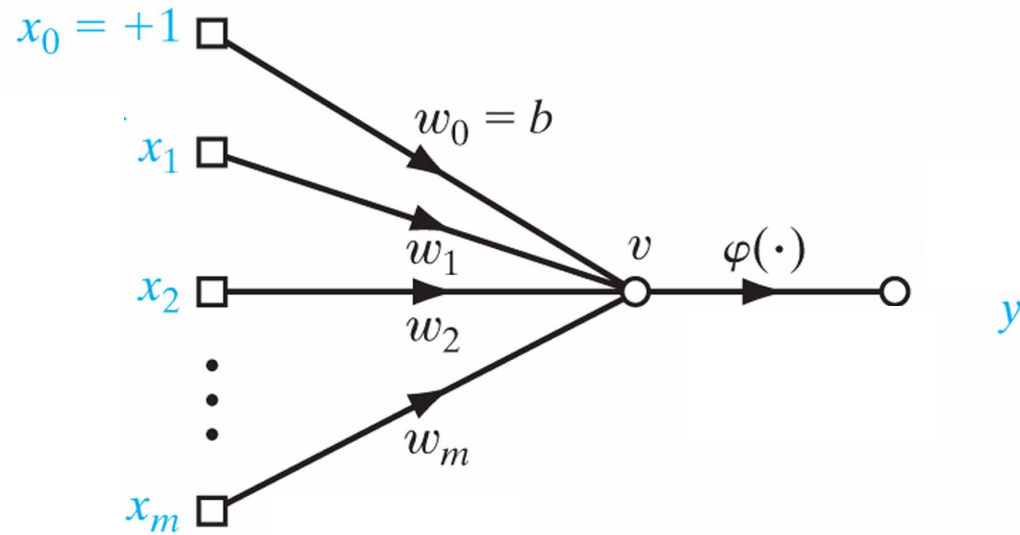
- 1 Introdução
- 2 Perceptron
- 3 Teorema de Convergência
- 4 Algoritmo de Treinamento para Perceptron
- 5 Exemplo / Exercício
- 6 Referências

Perceptron

- *Perceptron* → Neurônio de *McCulloch-Pitts*

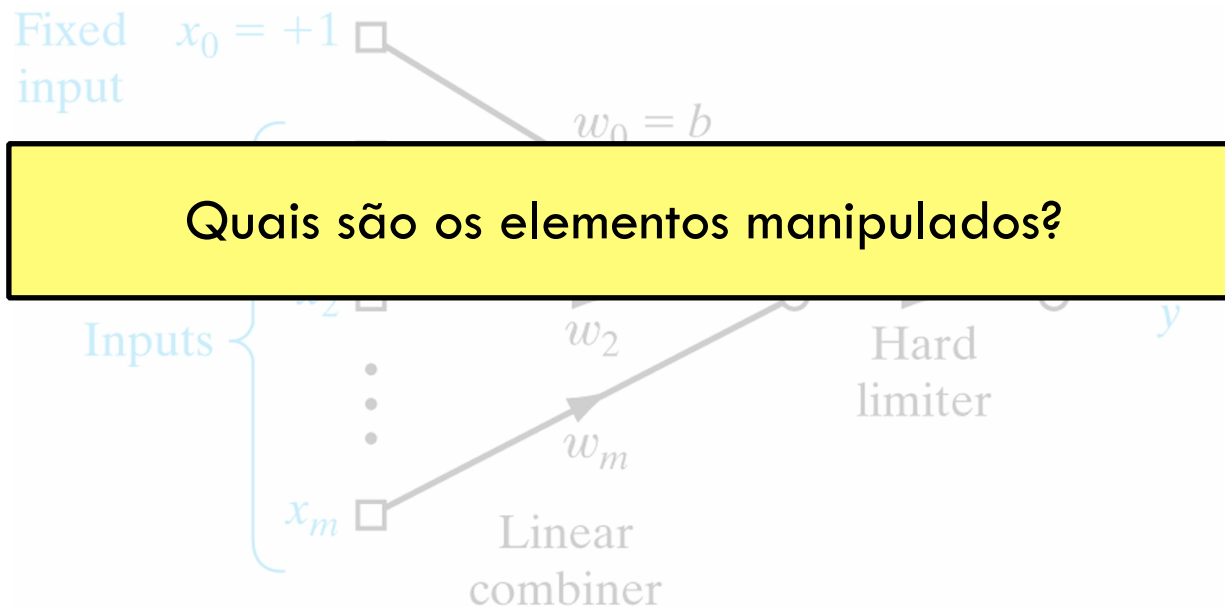
Perceptron

- *Perceptron* → Neurônio de McCulloch-Pitts



Perceptron

- *Perceptron* → Neurônio de *McCulloch-Pitts*



Perceptron

$$V_k = \sum_{j=0}^m w_{kj} * x_j$$

$$y_k = \phi(v_k)$$

Perceptron

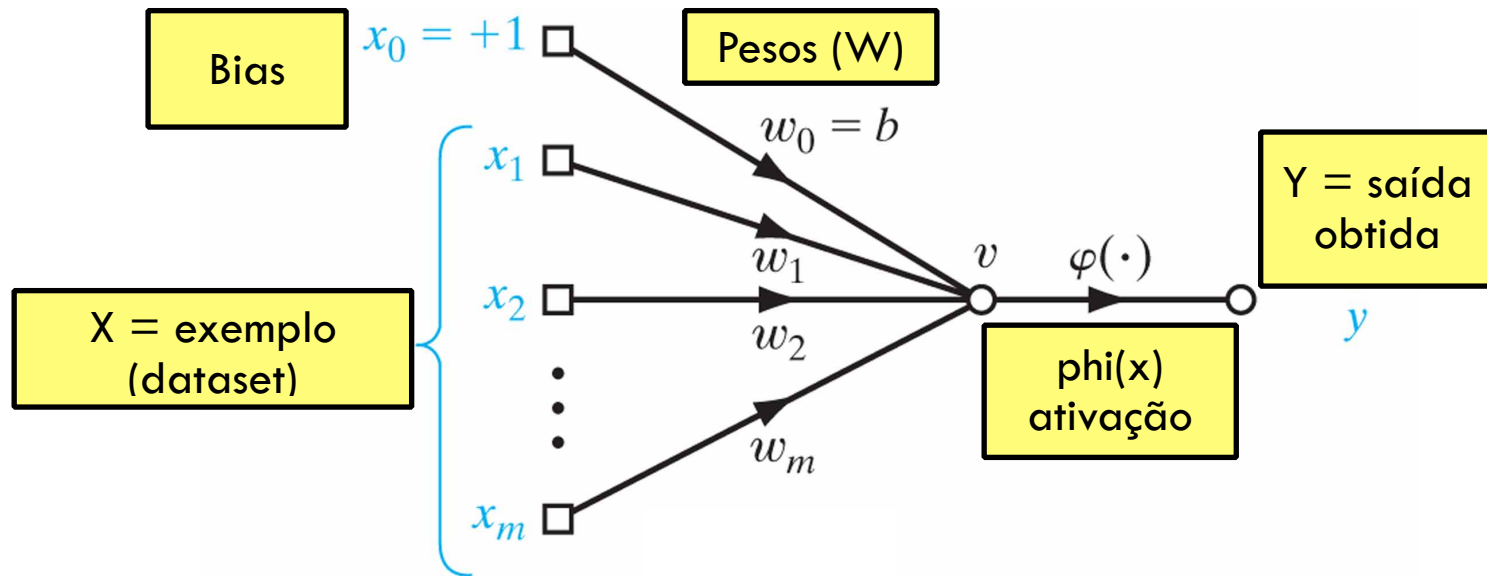
$$V_k = \sum_{j=0}^m w_{kj} * x_j$$

$$y_k = \phi(v_k)$$

- **X** são os sinais de entrada (exemplo do dataset)
- **W** são os pesos sinápticos do neurônio k
- v_k é a combinação linear de **W** e **X** (entradas)
- b_k é o bias
- $\phi(.)$ é a função de ativação
- y_k é a saída do neurônio

Perceptron

- *Perceptron* → Neurônio de McCulloch-Pitts

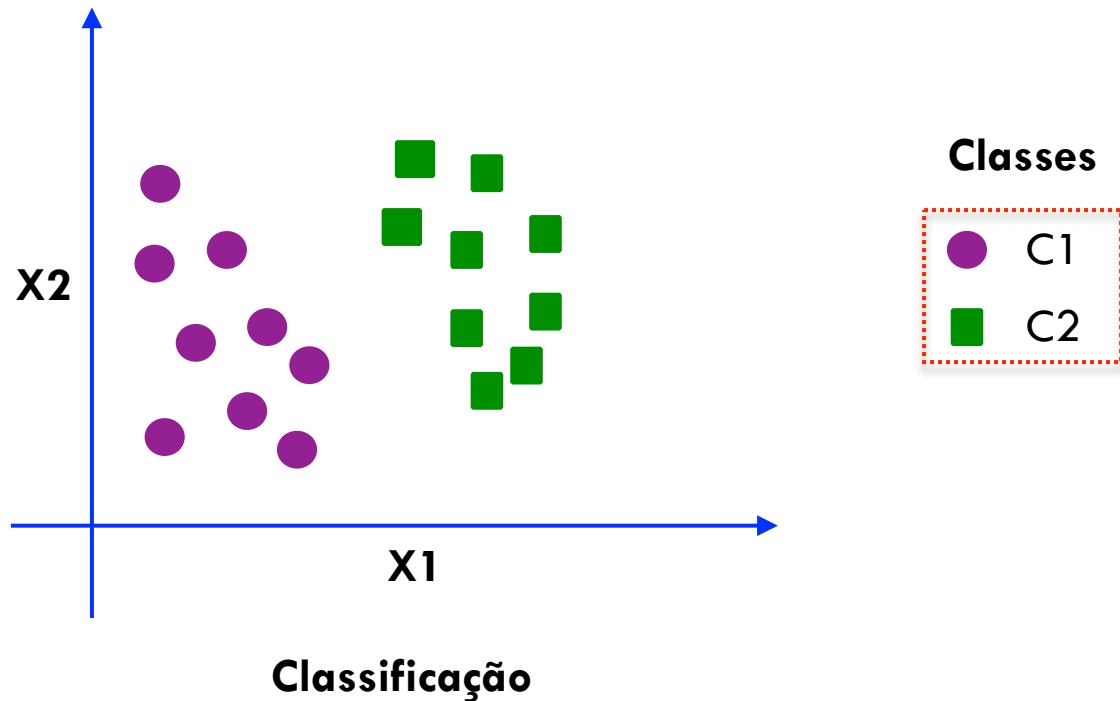


Perceptron

- **Objetivo:** classificar corretamente um conjunto de exemplos do dataset X em uma de duas classes, $C1$ ou $C2$

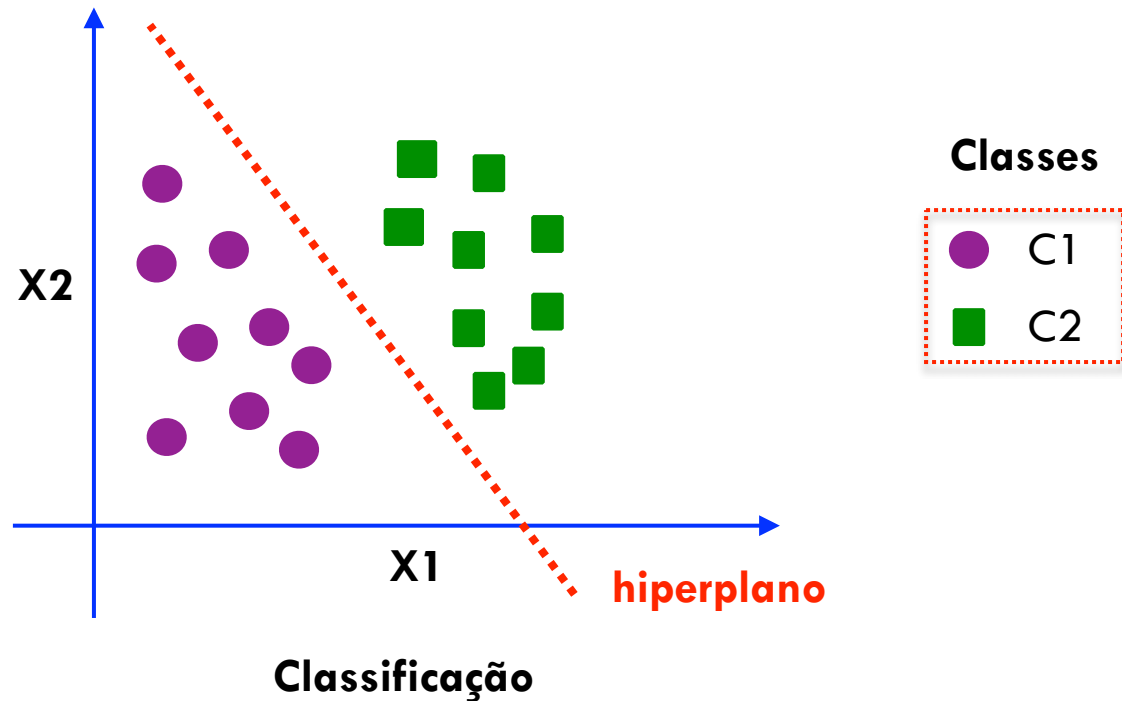
Perceptron

- **Objetivo:** classificar corretamente um conjunto de exemplos do dataset X em uma de duas classes, $C1$ ou $C2$



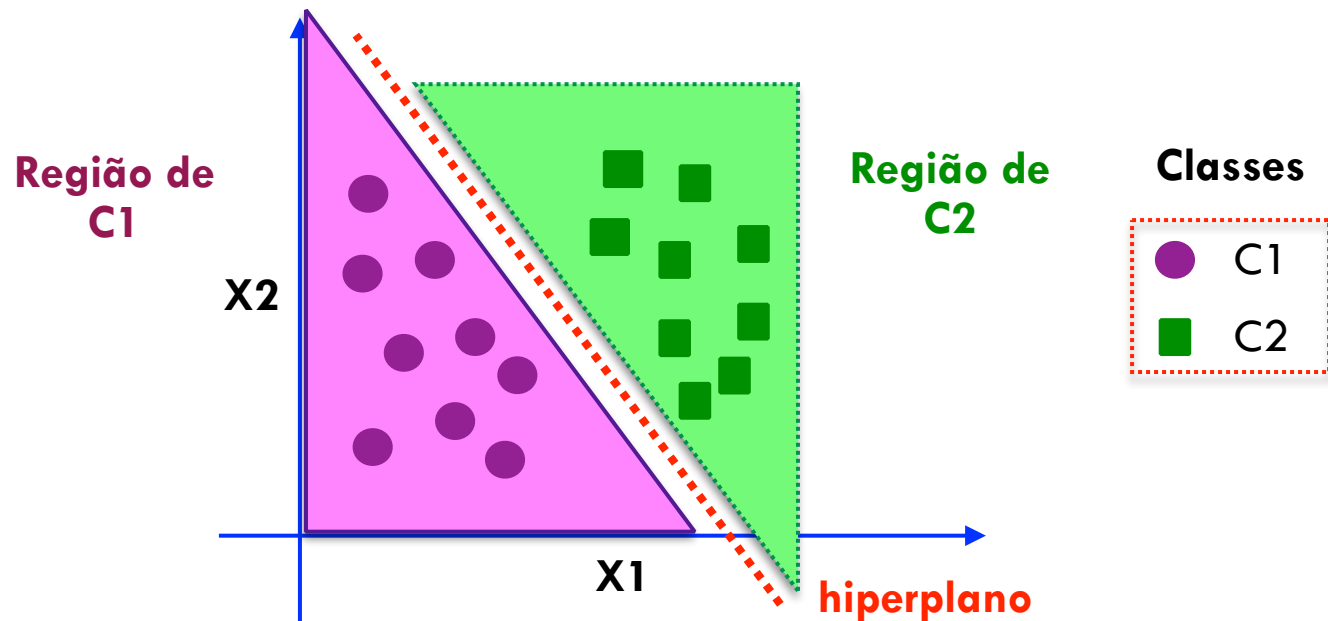
Perceptron

- **Objetivo:** classificar corretamente um conjunto de exemplos do dataset X em uma de duas classes, $C1$ ou $C2$



Perceptron

- **Objetivo:** classificar corretamente um conjunto de exemplos do dataset X em uma de duas classes, $C1$ ou $C2$



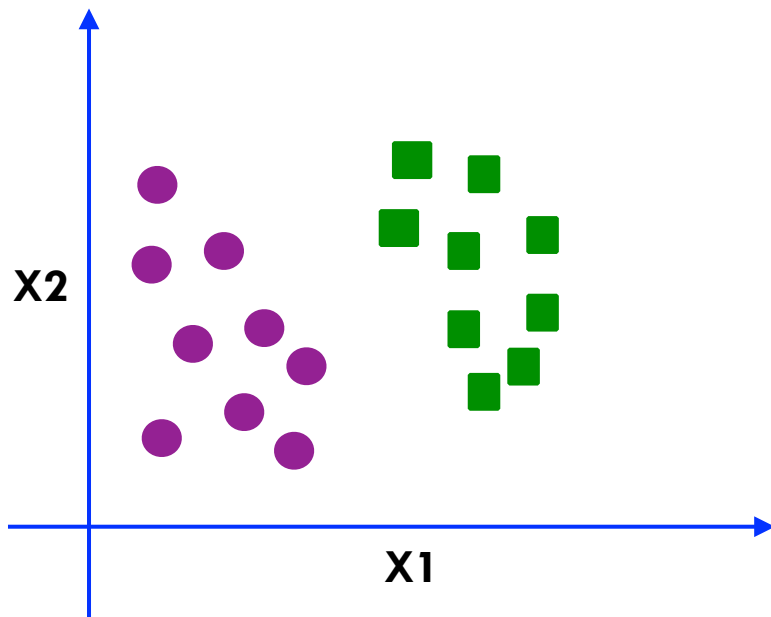
Perceptron gera um **hiperplano** (reta) separador

Perceptron

- **Aprendizado:** ajuste **iterativo** dos pesos sinápticos usando o algoritmo de convergência do *Perceptron*

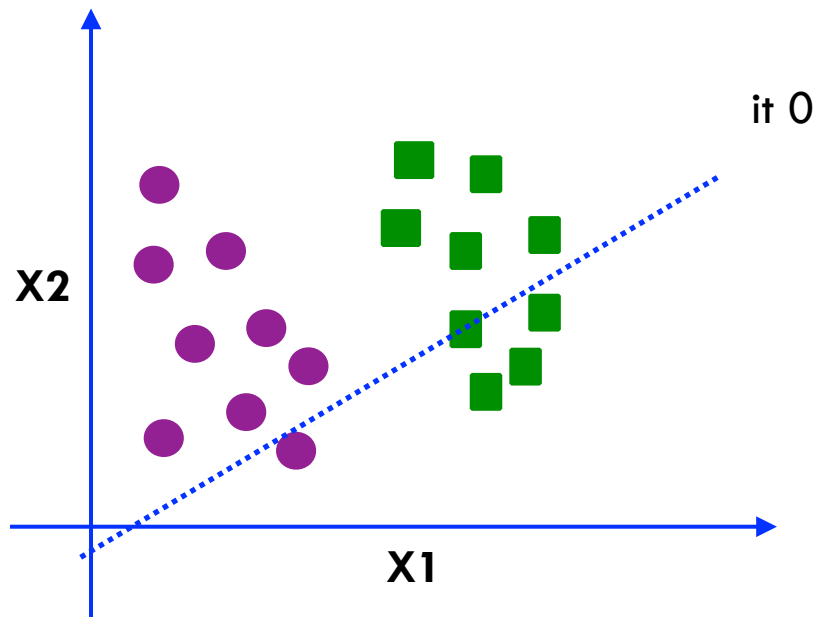
Perceptron

- **Aprendizado:** ajuste **iterativo** dos pesos sinápticos usando o algoritmo de convergência do perceptron



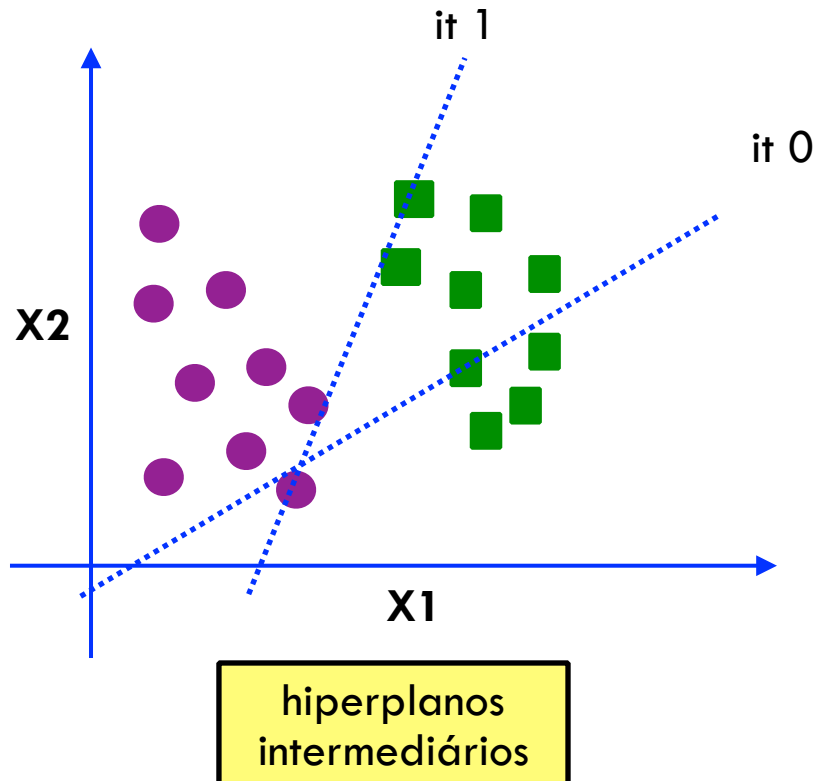
Perceptron

- **Aprendizado:** ajuste **iterativo** dos pesos sinápticos usando o algoritmo de convergência do perceptron



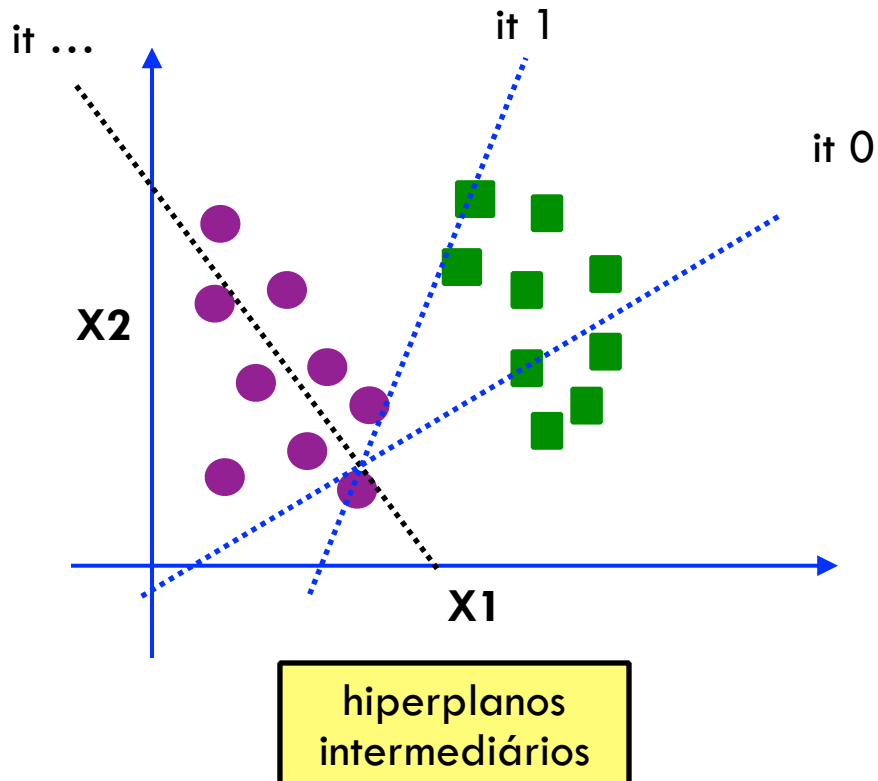
Perceptron

- **Aprendizado:** ajuste **iterativo** dos pesos sinápticos usando o algoritmo de convergência do perceptron



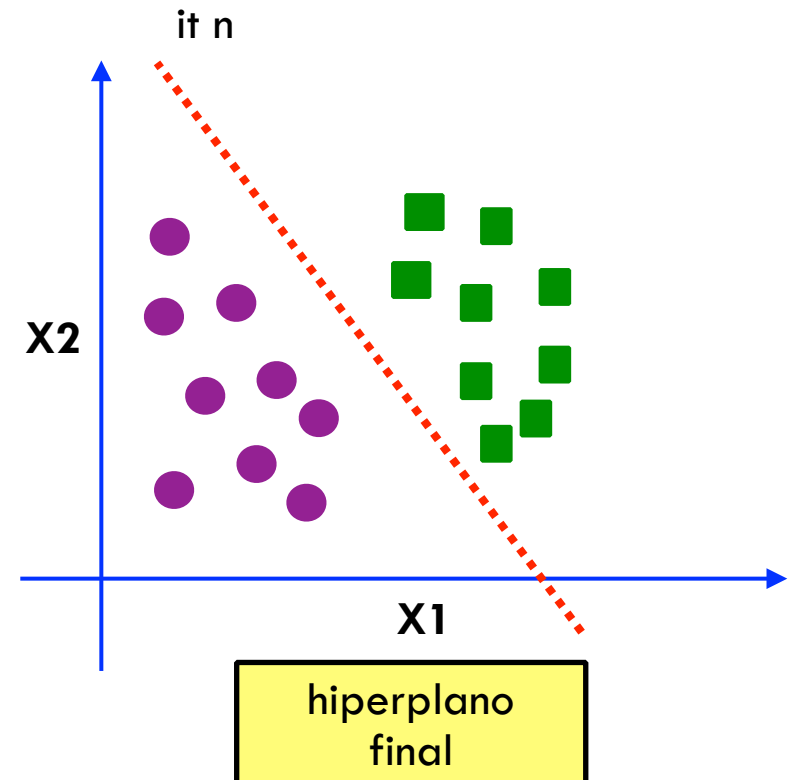
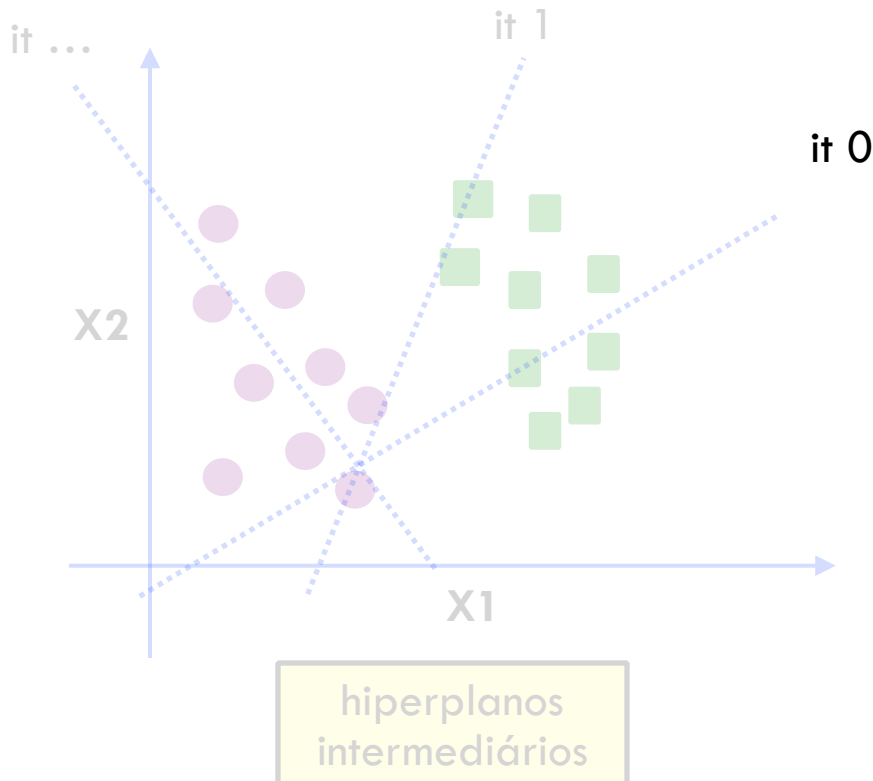
Perceptron

- **Aprendizado:** ajuste **iterativo** dos pesos sinápticos usando o algoritmo de convergência do perceptron



Perceptron

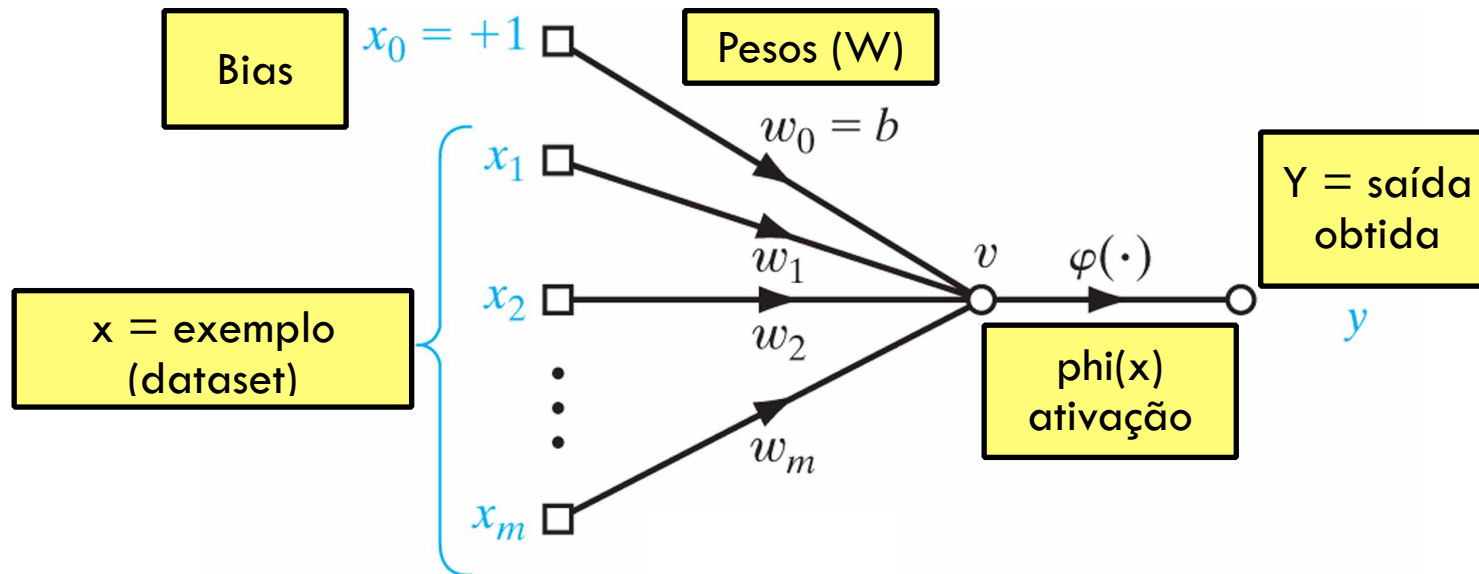
- **Aprendizado:** ajuste **iterativo** dos pesos sinápticos usando o algoritmo de convergência do perceptron



Roteiro

- 1 Introdução
- 2 Perceptron
- 3 Teorema de Convergência
- 4 Algoritmo de Treinamento para Perceptron
- 5 Exemplo / Exercício
- 6 Referências

Teorema de Convergência



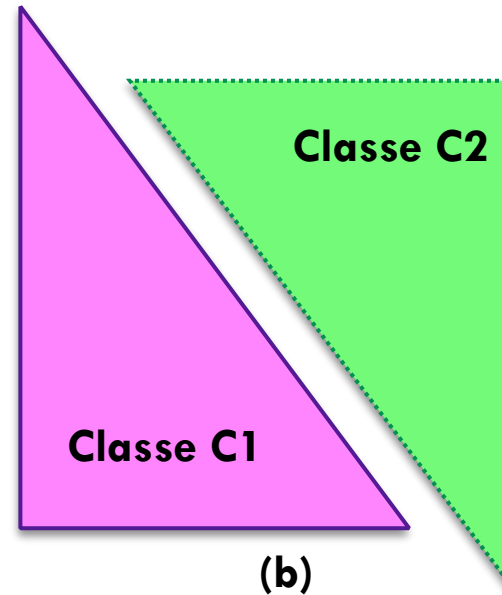
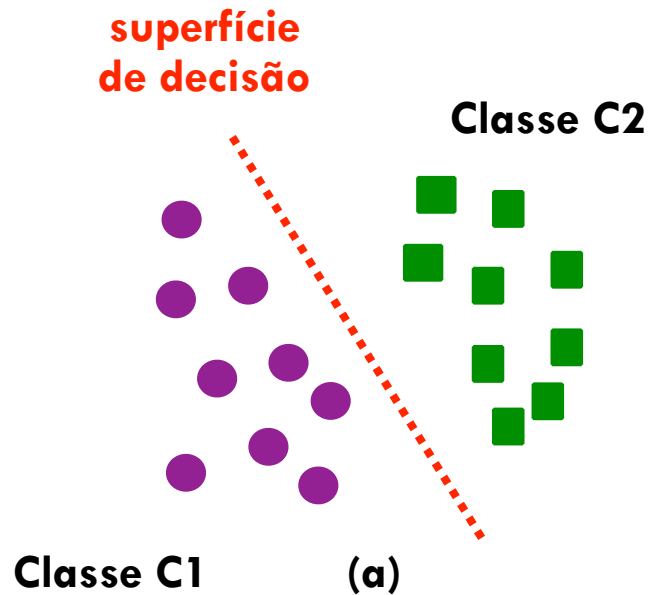
Teorema de Convergência

- **bias $b(n)$:** é um peso w_0 associado a uma entrada $+1$
- **vetor de entrada $\mathbf{X}(n)$:** $[+1, x_1(n), x_2(n), \dots, x_m(n)]^T$
- **vetor de pesos $\mathbf{W}(n)$:** $[b, w_1(n), w_2(n), \dots, w_m(n)]^T$

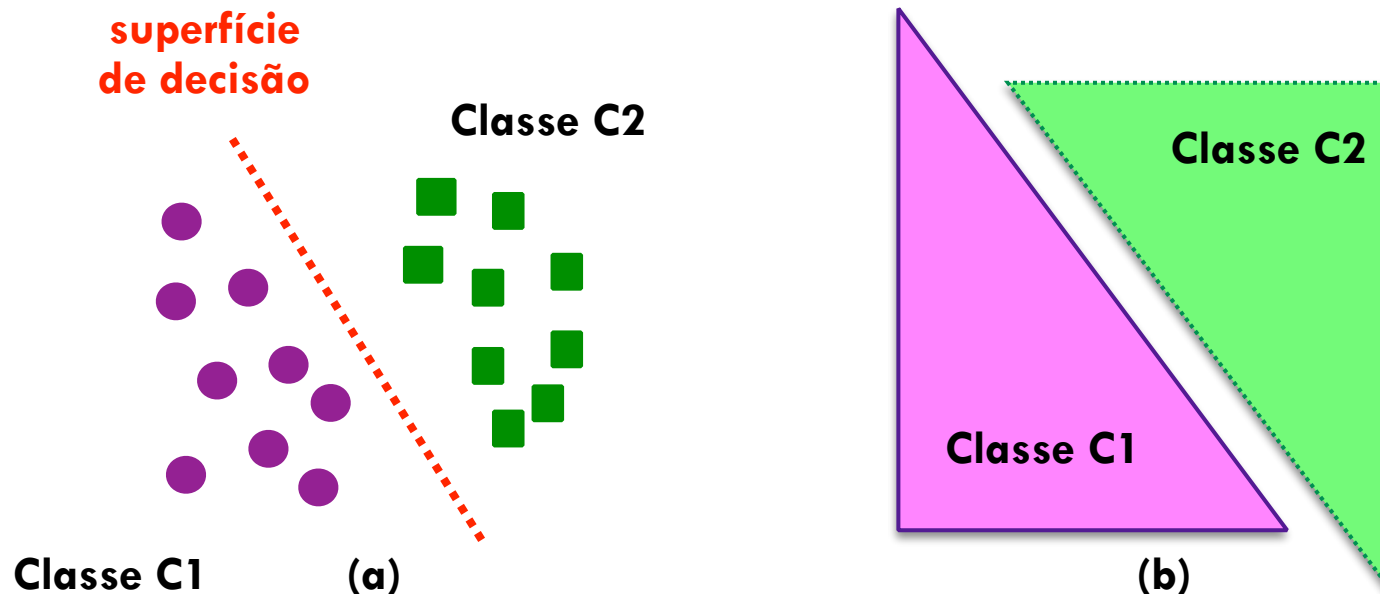
$$v(n) = \sum_{i=0}^m w_i(n) x_i(n) = \mathbf{w}^T(n) \mathbf{x}(n)$$

v = sinal do neurônio

Teorema de Convergência



Teorema de Convergência



$\mathbf{w}^T \mathbf{x} = 0$ define um hiperplano de separação
 $\mathbf{w}^T \mathbf{x} > 0$ para todo vetor \mathbf{x} pertencente à classe C1
 $\mathbf{w}^T \mathbf{x} \leq 0$ para todo vetor \mathbf{x} pertencente à classe C2

Teorema de Convergência

- Se o n -ésimo vetor $\mathbf{x}(n)$ é corretamente classificado pelo vetor $\mathbf{w}(n)$ na n -ésima iteração do algoritmo, nenhuma correção é feita no vetor de pesos:
 - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T \mathbf{x}(n) > 0$ e $\mathbf{x}(n) \ni$ a classe C1
 - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T \mathbf{x}(n) \leq 0$ e $\mathbf{x}(n) \ni$ a classe C2

Teorema de Convergência

- Se o n -ésimo vetor $\mathbf{x}(n)$ é corretamente classificado pelo vetor $\mathbf{w}(n)$ na n -ésima iteração do algoritmo, nenhuma correção é feita no vetor de pesos:
 - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T \mathbf{x}(n) > 0$ e $\mathbf{x}(n) \ni$ a classe C1
 - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T \mathbf{x}(n) \leq 0$ e $\mathbf{x}(n) \ni$ a classe C2
- Caso contrário, o vetor de pesos é atualizado:
 - $\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n) \mathbf{x}(n)$ se $\mathbf{w}^T \mathbf{x}(n) > 0$ e $\mathbf{x}(n) \ni$ classe C2
 - $\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n) \mathbf{x}(n)$ se $\mathbf{w}^T \mathbf{x}(n) \leq 0$ e $\mathbf{x}(n) \ni$ classe C1

Teorema de Convergência

- Se o n -ésimo vetor $\mathbf{x}(n)$ é corretamente classificado pelo vetor $\mathbf{w}(n)$ na n -ésima iteração do algoritmo, nenhuma correção é feita no vetor de pesos:
 - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T \mathbf{x}(n) > 0$ e $\mathbf{x}(n) \ni$ a classe C1
 - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T \mathbf{x}(n) \leq 0$ e $\mathbf{x}(n) \ni$ a classe C2
 - Caso contrário, o vetor de pesos é atualizado:
 - $\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n) \mathbf{x}(n)$ se $\mathbf{w}^T \mathbf{x}(n) > 0$ e $\mathbf{x}(n) \ni$ classe C2
 - $\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n) \mathbf{x}(n)$ se $\mathbf{w}^T \mathbf{x}(n) \leq 0$ e $\mathbf{x}(n) \ni$ classe C1
- η é a taxa de aprendizado que controla o ajuste dos pesos
 - hiper-parâmetro do algoritmo
 - parâmetro x hiper-parâmetro

Teorema de Convergência

- A saída do neurônio é computada usando a função sinal $\text{sgn}(\cdot)$:

**função
sinal**

$$\text{sgn}(v) = \begin{cases} +1 & \text{se } v > 0 \\ -1 & \text{se } v < 0 \end{cases}$$

- Expressamos a saída $y(n)$ de maneira compacta:

$$y(n) = \text{sgn}[w^T(n) x(n)]$$

Teorema de Convergência

- Regra de Atualização dos Pesos sinápticos:
 - Dada uma instância n ,

$$w(n+1) \leftarrow w(n) + \eta * (d(n) - y(n)) * x(n)$$

Teorema de Convergência

- Regra de Atualização dos Pesos sinápticos:
 - Dada uma instância n ,

$$w(n+1) \leftarrow w(n) + \eta * (d(n) - y(n)) * x(n)$$

$d(n) - y(n) = \text{sinal do erro (+ ou -)}$


"erro entre a saída real (d) e a saída obtida (y)"

Teorema de Convergência

$$E^2 = (d(n) - y(n))$$


Teorema de Convergência

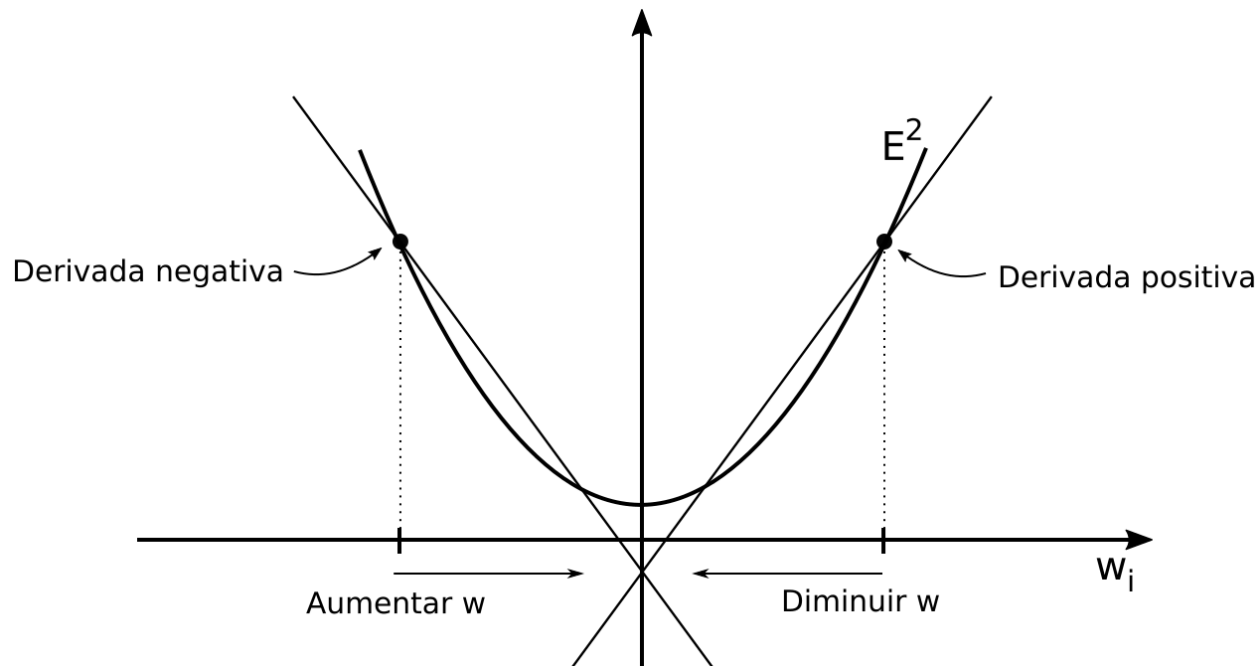
$$E^2 = (d(n) - y(n))^2$$

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$


Teorema de Convergência

$$E^2 = (d(n) - y(n))^2$$

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$




Teorema de Convergência

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$

erro quadrático

1

2

Teorema de Convergência

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$

erro quadrático

1 $w_i(n+1) = w_i(n) - \eta \frac{dE^2}{dw_i}$

2

Teorema de Convergência

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$

erro quadrático

1 $w_i(n+1) = w_i(n) - \eta \frac{dE^2}{dw_i}$



2 $\frac{dE^2}{dw_i} = \frac{d(d(n) - y(n))^2}{dw_i} = 2 * (d(n) - w^T(n) x(n)) * -x_i$

Teorema de Convergência

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$

erro quadrático

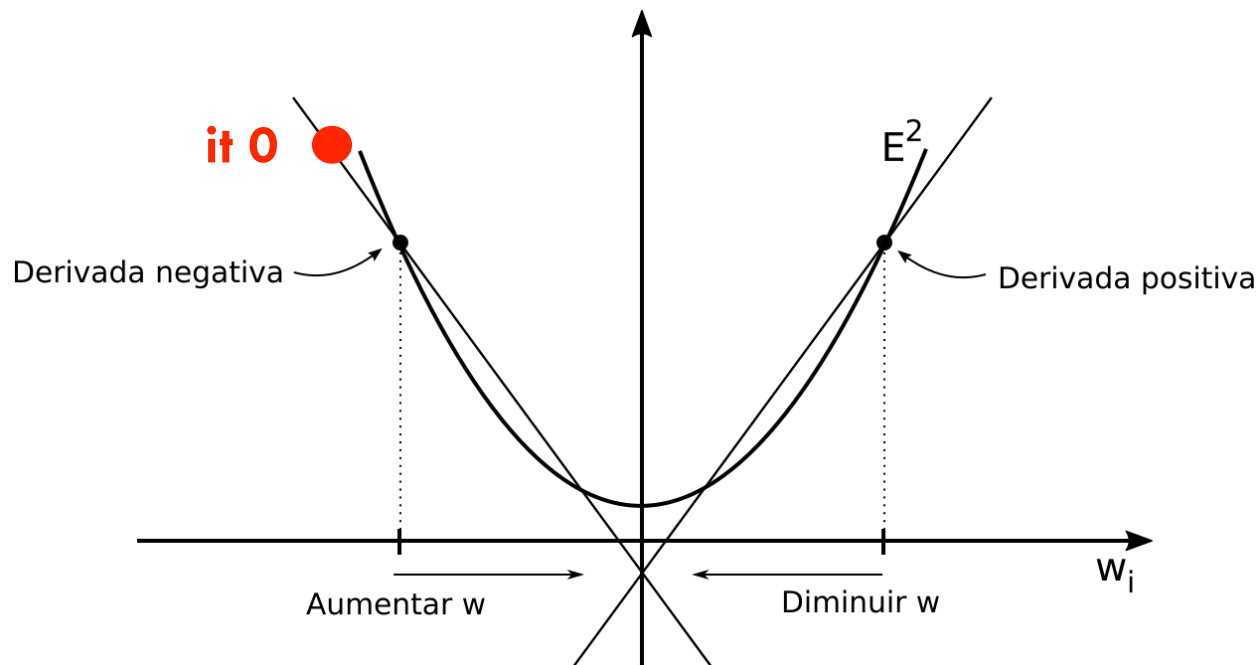
1 $w_i(n+1) = w_i(n) - \eta \frac{dE^2}{dw_i}$

2 $\frac{dE^2}{dw_i} = \frac{d(d(n) - y(n))^2}{dw_i} = 2 * (d(n) - w^T(n) x(n)) * -x_i$

Teorema de Convergência

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$

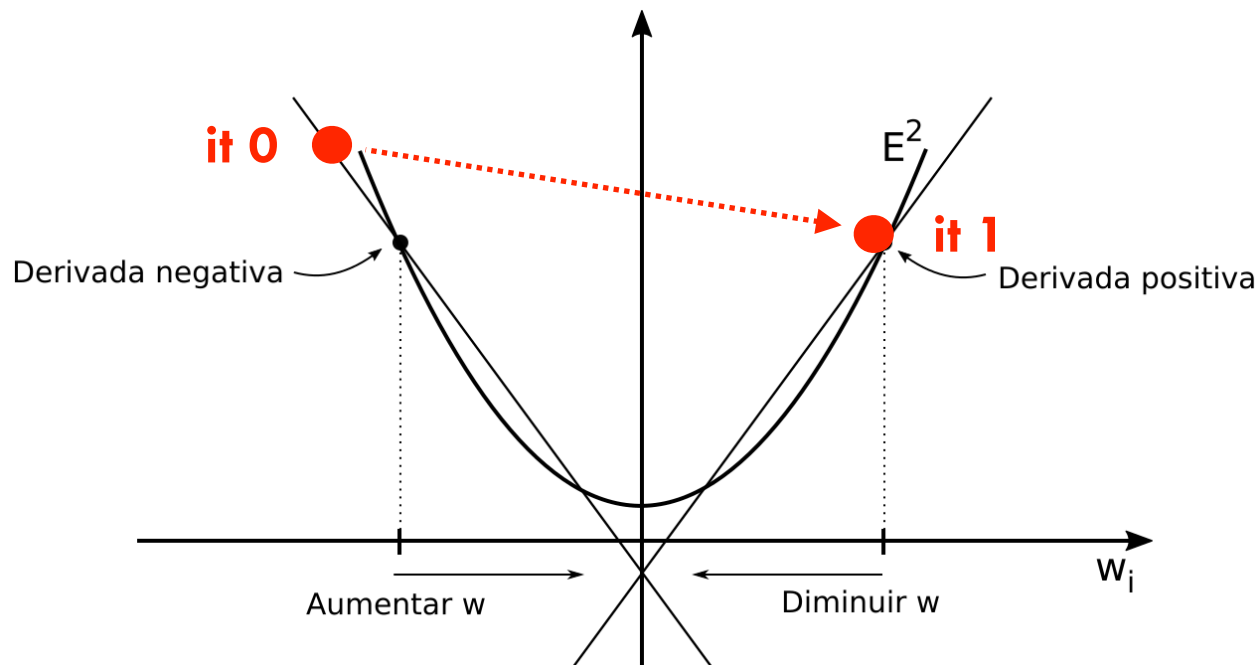
erro quadrático



Teorema de Convergência

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$

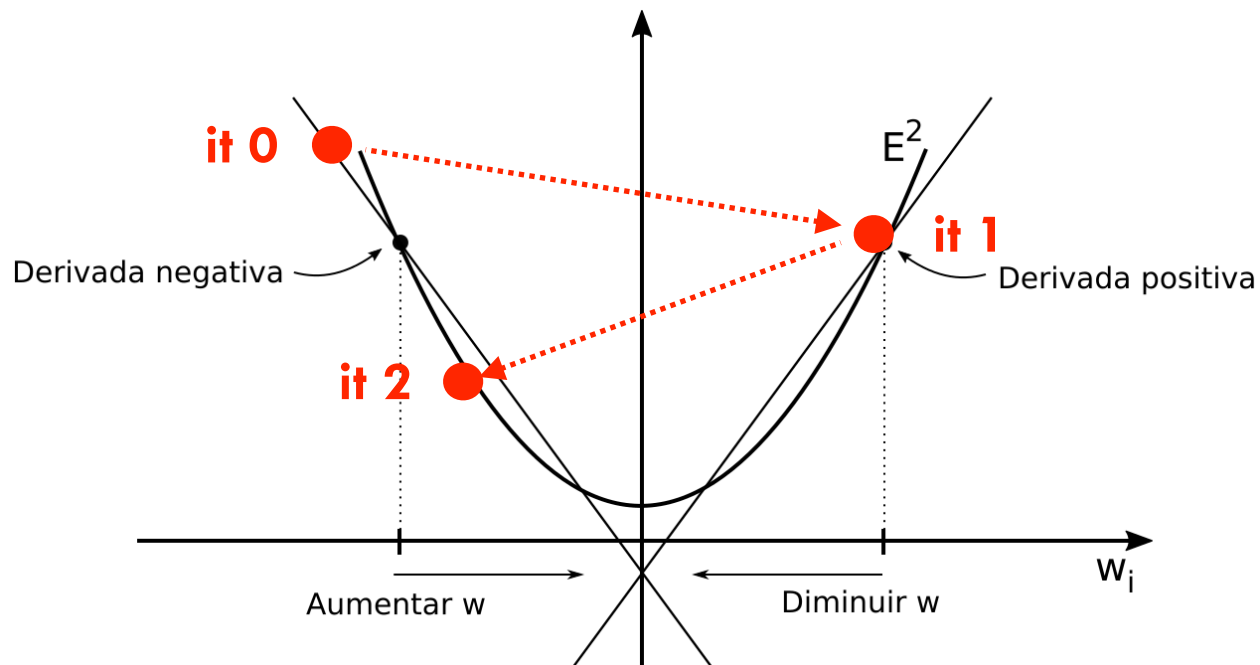
erro quadrático



Teorema de Convergência

$$E^2 = (d(n) - y(n)) = (d(n) - w^T(n) x(n))^2$$

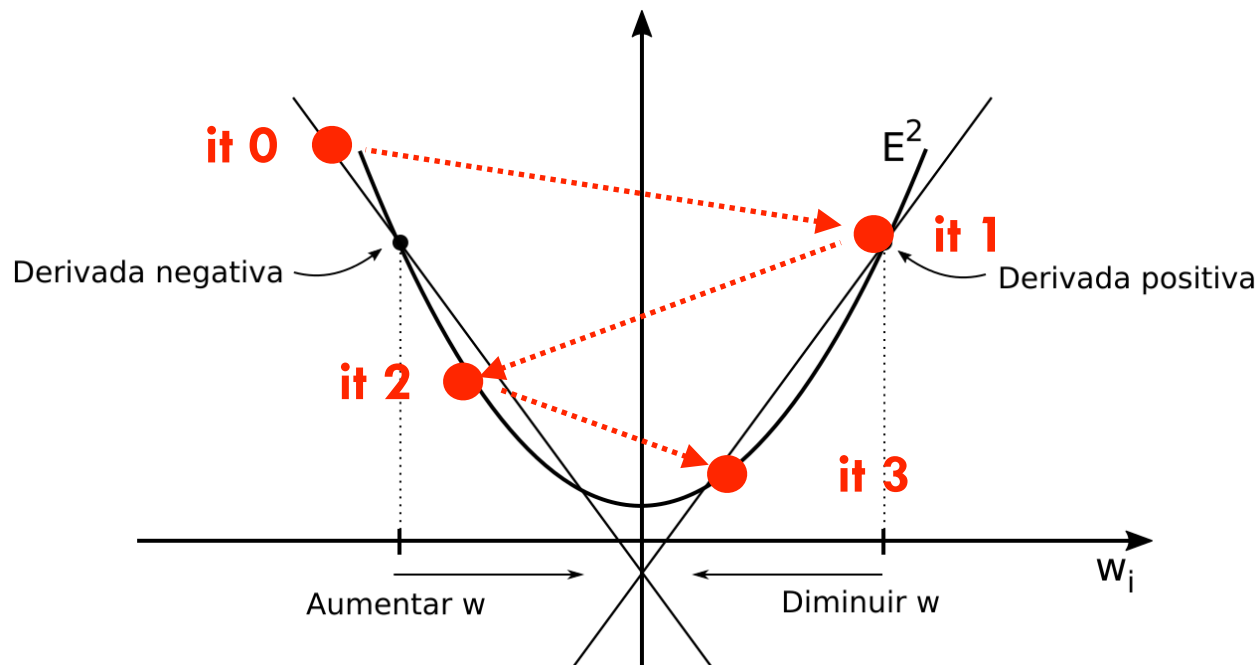
erro quadrático



Teorema de Convergência

$$E^2 = (d(n) - y(n)) = (d(n) - w^T(n) x(n))^2$$

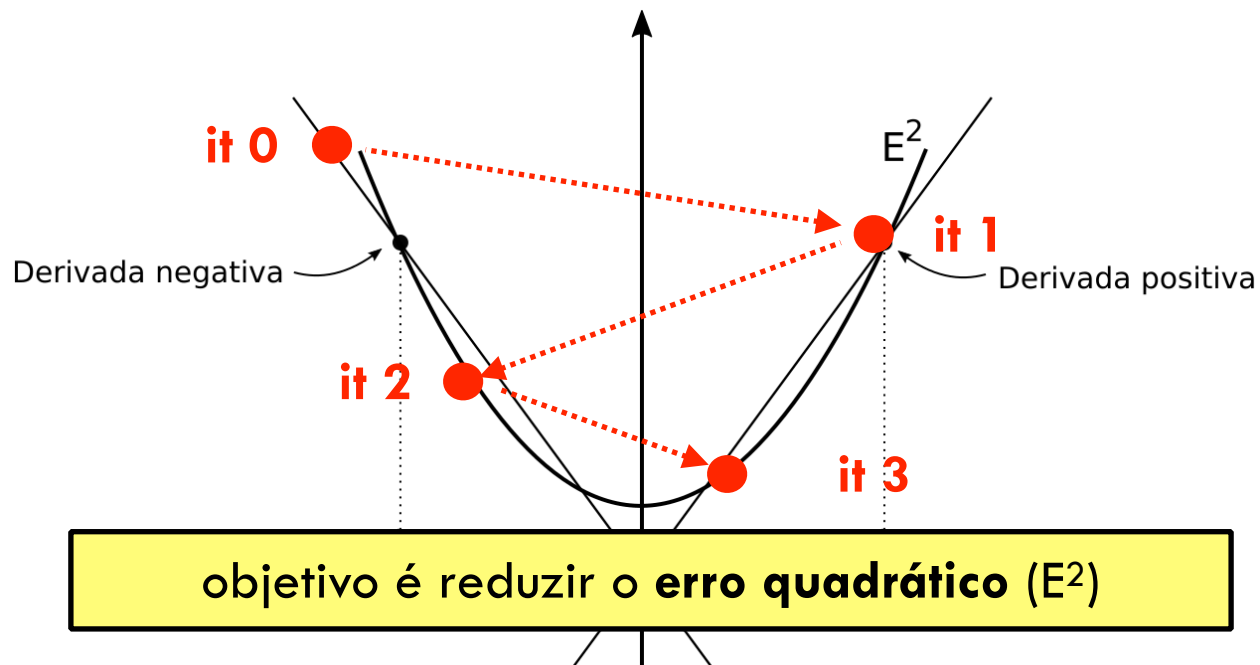
erro quadrático



Teorema de Convergência

$$E^2 = (d(n) - y(n))^2 = (d(n) - w^T(n) x(n))^2$$

erro quadrático



Roteiro

- 1 Introdução
- 2 Perceptron
- 3 Teorema de Convergência
- 4 Algoritmo de Treinamento para Perceptron
- 5 Exemplo / Exercício
- 6 Referências

Algoritmo de Treinamento

- Entradas e hiper-parâmetros:
 - $\mathbf{X}(n)$: vetor de entrada
 - $\mathbf{W}(n)$: vetor de pesos
 - \mathbf{b} : bias
 - $\mathbf{y}(n)$: saída obtida
 - $\mathbf{d}(n)$: saída desejada (real)
 - η : taxa de aprendizado
- Funcionamento:
 - reduzir o erro entre as saídas esperadas, e as saídas obtidas

Algoritmo de Treinamento

□ Entradas:

- conjunto de treinamento com exemplos rotulados $[X \mid D]$
 - X são os exemplos de treinamento
 - D são as saídas reais, esperadas
- taxa de aprendizagem (η)
- pesos sinápticos iniciais (\mathbf{W}) [opcional]
- número máximo de iterações para treinamento (**$n.iter$**)

□ Saídas:

- \mathbf{W} ajustados para todos os exemplos de treinamento
- Épocas: numero de épocas

Pseudocódigo: DFS

Perceptron (Inicial):

// *Inicialização*

Pseudocódigo: DFS

Perceptron (Inicial):

// *Inicialização*

1. Iniciar o vetor **W** com valores aleatórios pequenos. Sugestão: $[-1, 1]$ ou $[-0.5, 0.5]$
2. Iniciar o contador de número de épocas (épocas $\leftarrow 0$)
3. Iniciar variável de controle (erro \leftarrow **TRUE**)

Pseudocódigo: DFS

Perceptron (Inicial):

// *Inicialização*

1. Iniciar o vetor **W** com valores aleatórios pequenos. Sugestão: $[-1, 1]$ ou $[-0.5, 0.5]$
2. Iniciar o contador de número de épocas ($\text{épocas} \leftarrow 0$)
3. Iniciar variável de controle ($\text{erro} \leftarrow \text{TRUE}$)
4. **Repetir** enquanto ($\text{error} == \text{TRUE} \ \& \ \text{epoca} < \text{n.iter}$)
- 5.

Pseudocódigo: DFS

Perceptron (Inicial):

// Inicialização

1. Iniciar o vetor **W** com valores aleatórios pequenos. Sugestão: $[-1, 1]$ ou $[-0.5, 0.5]$
2. Iniciar o contador de número de épocas ($\text{épocas} \leftarrow 0$)
3. Iniciar variável de controle ($\text{erro} \leftarrow \text{TRUE}$)
4. **Repetir** enquanto ($\text{error} == \text{TRUE} \ \& \ \text{epoca} < \text{n.lter}$)
5. **Para** todas as amostras de treinamento em **X**, fazer:

Pseudocódigo: DFS

Perceptron (Inicial):

```
// Inicialização
1. Iniciar o vetor W com valores aleatórios pequenos. Sugestão: [-1, 1] ou [-0.5, 0.5]
2. Iniciar o contador de número de épocas (épocas  $\leftarrow$  0)
3. Iniciar variável de controle (erro  $\leftarrow$  TRUE)
4. Repetir enquanto (error == TRUE & epoca < n.iter)
5.   Para todas as amostras de treinamento em X, fazer:
6.      $V = W' * X$  //Calcular o sinal do neurônio (spike)
7.      $Y = \text{phi}(V)$  // Calcular o sinal de saída do neurônio (Y)
8.
9.
10.
```

Pseudocódigo: DFS

Perceptron (Inicial):

```
// Inicialização
1. Iniciar o vetor W com valores aleatórios pequenos. Sugestão: [-1, 1] ou [-0.5, 0.5]
2. Iniciar o contador de número de épocas (épocas  $\leftarrow$  0)
3. Iniciar variável de controle (erro  $\leftarrow$  TRUE)
4. Repetir enquanto (error == TRUE & epoca < n.iter)
5.   Para todas as amostras de treinamento em X, fazer:
6.      $V = W' * X$  //Calcular o sinal do neurônio (spike)
7.      $Y = \text{phi}(V)$  // Calcular o sinal de saída do neurônio (Y)
8.     Se Y (saída obtida)  $\neq$  Di (saída real): // erro na predição
9.        $W = W + \eta * (D_i - Y) * X$ 
10.      erro  $\leftarrow$  TRUE
```

Pseudocódigo: DFS

Perceptron (Inicial):

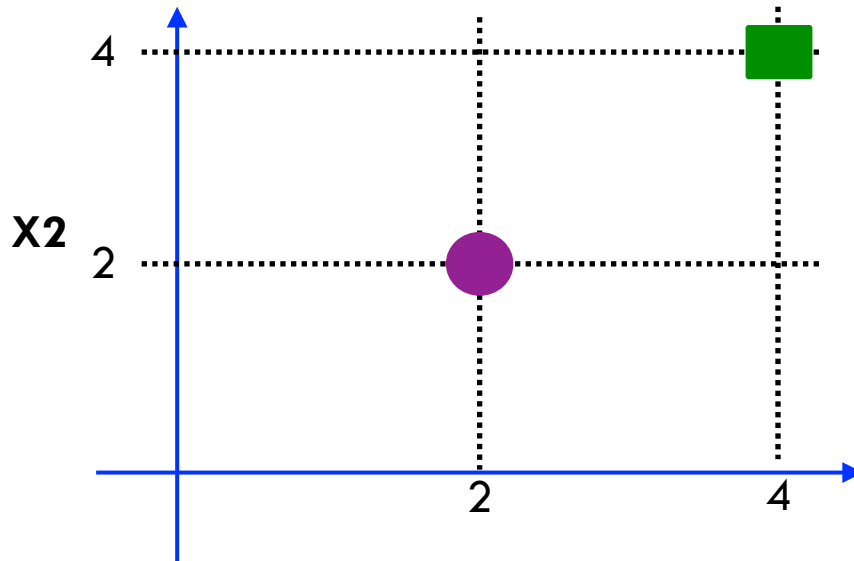
11. | **Fim se.**
12. | **Fim Para.**
13. | $\text{epocas} \leftarrow \text{epocas} + 1$ // *Incrementar o contador do numero de épocas*
14. | **Fim Repita.**
15. | **Fim Pseudocódigo.**

Roteiro

- 1 Introdução
- 2 Perceptron
- 3 Teorema de Convergência
- 4 Algoritmo de Treinamento para Perceptron
- 5 Exemplo / Exercício
- 6 Referências

Exemplo

- Treinar o perceptron para o problema abaixo:
 - $w_0 = -0.5441$, $w_1 = 0.5562$, $w_2 = 0.4074$
 - $\text{bias} = -1$
 - $\eta = 0.1$



Exemplo	X1	X2	Classe
E1	2	2	1
E2	4	4	0

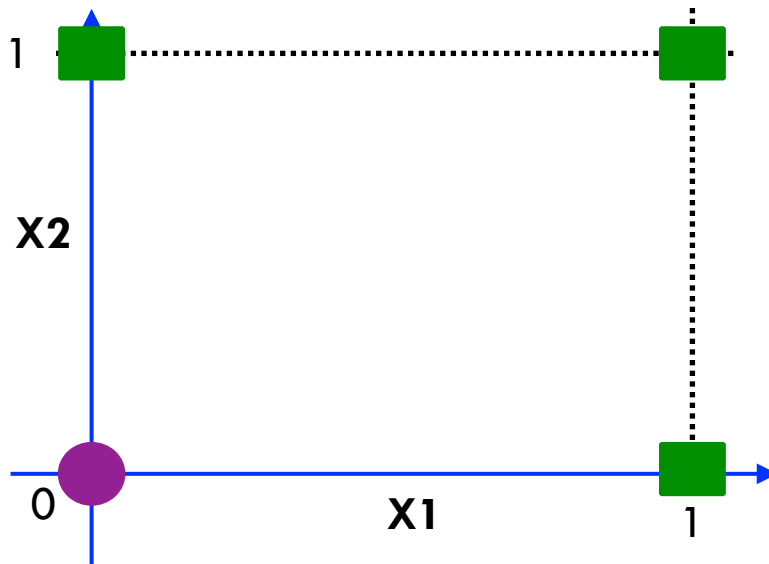


Exercício

- Treinar o perceptron para reconhecer o problema lógico OR.

Dados:

- $w_0 = w_1 = w_2 = 0.5$
- $\text{bias} = +1$
- $\eta = 0.1$



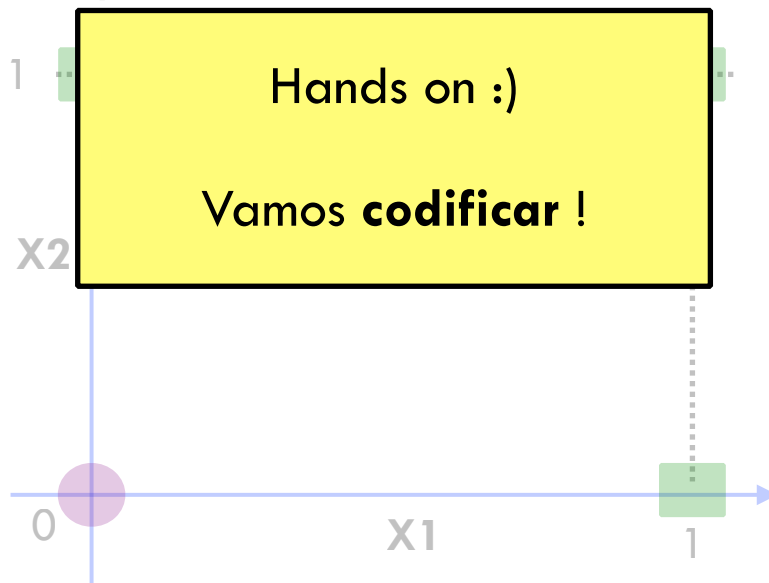
x_1	x_2	D	
0	0	0	●
0	1	1	■
1	0	1	■
1	1	1	■

Exercício

- Treinar o perceptron para reconhecer o problema lógico OR.

Dados:

- $w_0 = w_1 = w_2 = 0.5$
- $\text{bias} = +1$
- $\eta = 0.1$



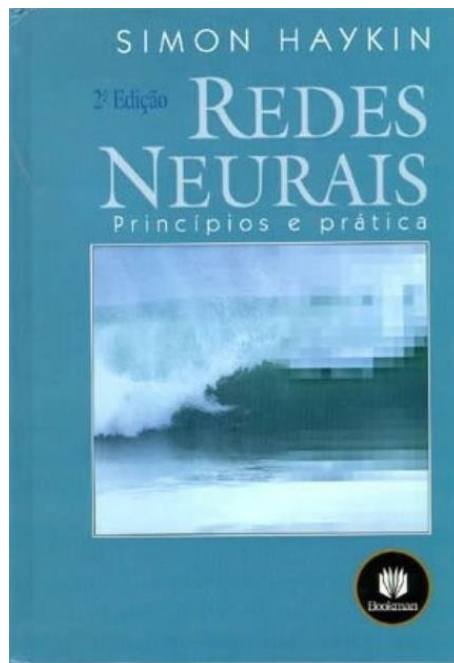
Síntese/Revisão

- Perceptron
 - um neurônio de McCulloch Pitts
 - bias
 - função de ativação degrau
- Teorema de Convergência
- Algoritmo de Aprendizado do Perceptron
- Hands-on

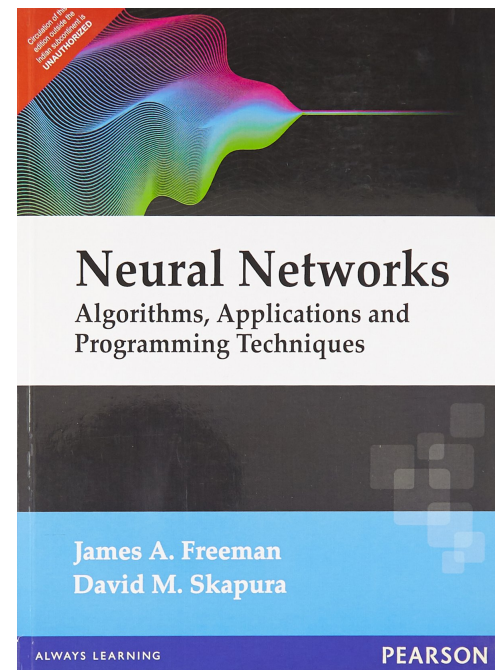
Roteiro

- 1 Introdução
- 2 Perceptron
- 3 Teorema de Convergência
- 4 Algoritmo de Treinamento para Perceptron
- 5 Exemplo / Exercício
- 6 Referências

Literatura Sugerida



(Haykin, 1999)

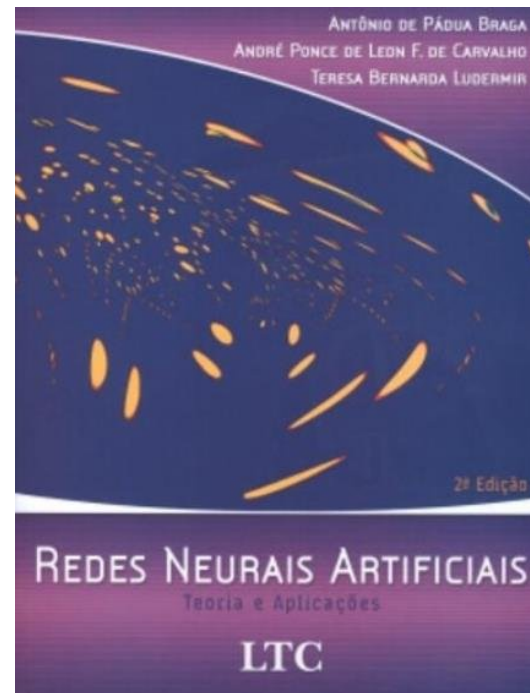


(Freeman & Skapura, 1991)

Literatura Sugerida



[Faceli et al, 2011]



[Braga et al, 2007]



Perguntas?

Prof. Rafael G. Mantovani

rgmantovani@gmail.com

Hiperplano obtido

- **Calcular o hiperplano** (após treinamento), problema 2D
 - Equação da reta: $y = mx + b$
 - m = inclinação da reta (*slope*)
 - b = interseção no eixo y (*y-intercept*)
 - Sendo W o vetor dos pesos, com w_0 sendo o peso do bias:
 - $\text{slope } (m) = - (w_0/w_2) / (w_0/w_1)$
 - $\text{y-intercept}(b) = -w_0/w_2$