MÉTODOS E MODELOS AVANÇADOS EM CIÊNCIA DE DADOS

Aula 02A - Perceptron

Prof. Rafael G. **Mantovani**

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Universidade Tecnológica Federal do Paraná (UTFPR)
Especialização em Ciência de Dados

Roteiro

1. Introdução
2. Perceptron
3. Teorema de Convergência
4. Algoritmo de Treinamento Perceptron
5. Exemplo / Exercício
6. Síntese / Próximas Aulas
7. Referências

Roteiro

1. Introdução
2. Perceptron
3. Teorema de Convergência
4. Algoritmo de Treinamento Perceptron
5. Exemplo / Exercício
6. Síntese / Próximas Aulas
7. Referências

Introdução

Relembrando nossa última aula ... :)

Introdução

- Paradigma Conexionista
- Redes Neurais Artificiais
- Inspiração Biológica (estrutura do cérebro)
- Neurônio artificial
- Funções de Ativação
- Topologias
- Algoritmos de Aprendizado

Introdução

- Perceptron (Rosenblatt, 1958):
    - primeira rede neural descrita algoritmicamente
    - Frank Rosenblatt (psicólogo)
    - modelo mais simples de rede neural que existe

Introdução

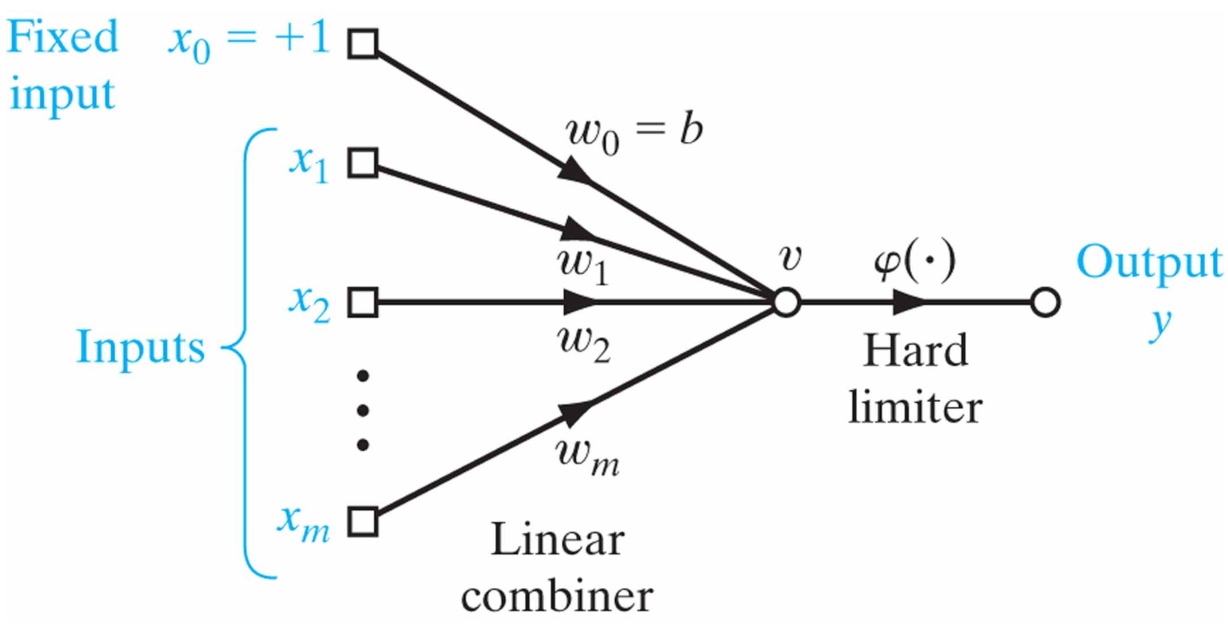
- Classifica padrões **linearmente separáveis**
- Possui um único neurônio com pesos sinápticos ajustáveis e **bias**
- Rosenblatt definiu um algoritmo de treinamento:
  - onde ocorre o ajuste dos parâmetros livres da rede (pesos sinápticos - W)
  - provou que se os exemplos utilizados no treino forem 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 Perceptron
5. Exemplo / Exercício
6. Síntese / Próximas Aulas
7. Referências

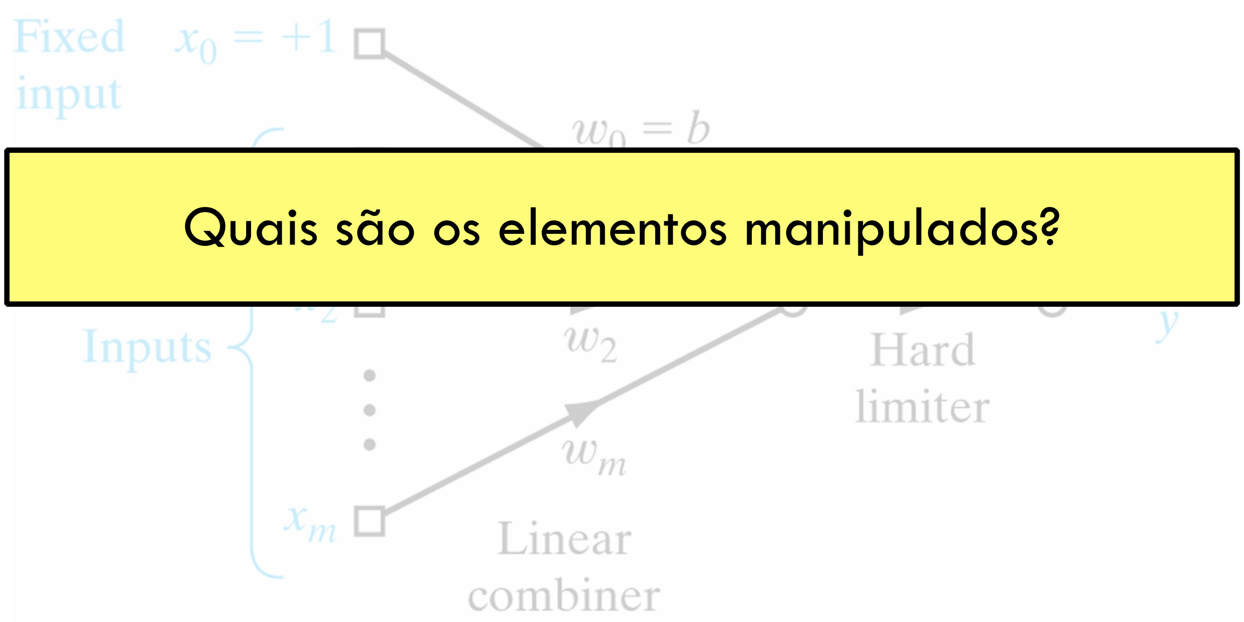
Perceptron

- Perceptron → Neurônio de McCulloch-Pitts



Perceptron

- Perceptron → Neurônio de McCulloch-Pitts



Quais são os elementos manipulados?

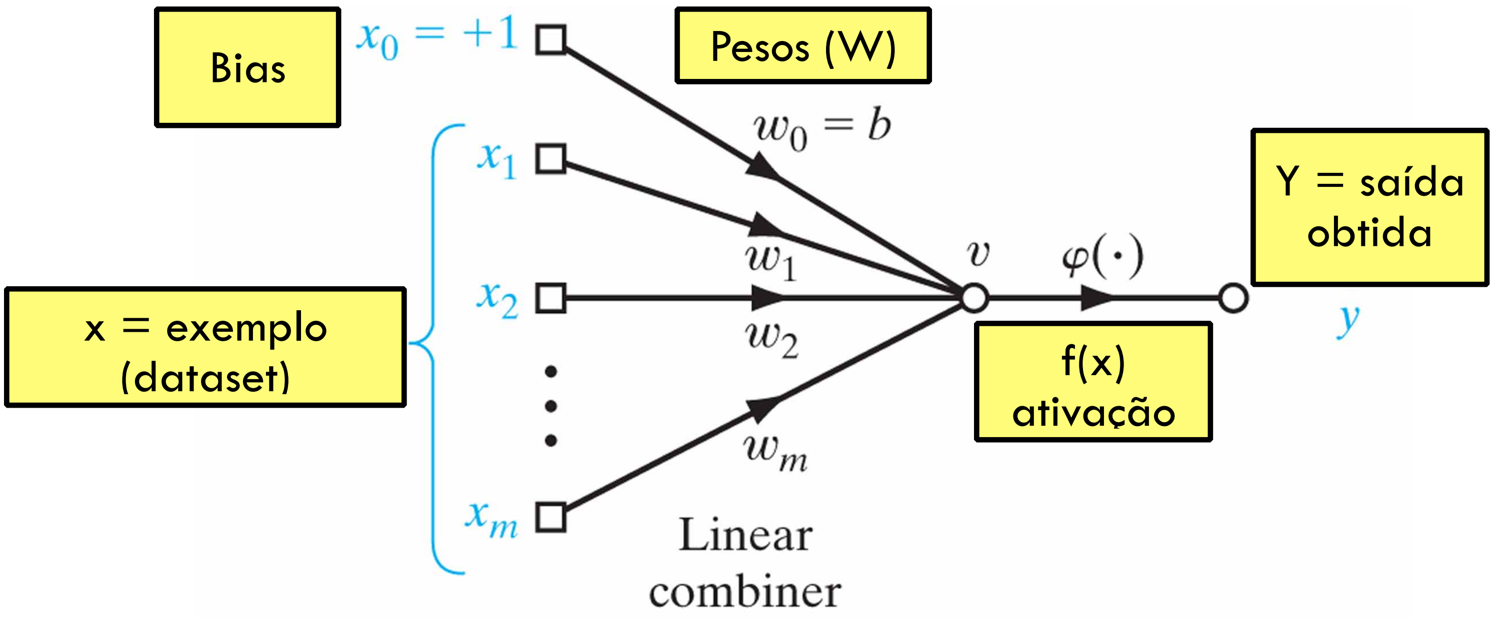
Perceptron

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \qquad \text{e} \qquad y_k = \varphi(v_k)$$

- **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

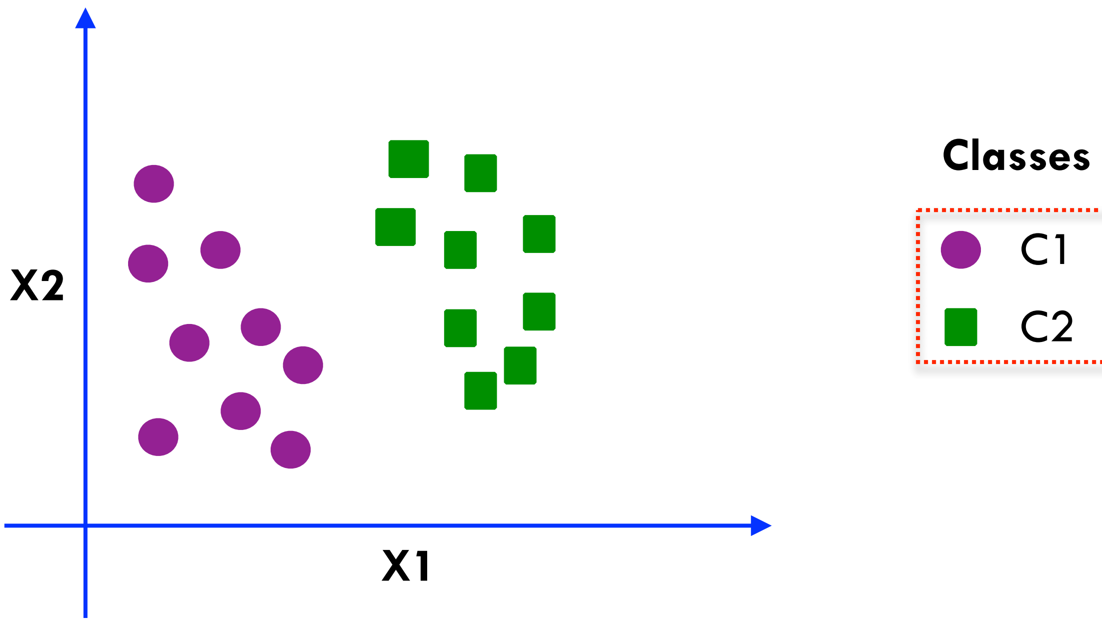
Perceptron

- Perceptron → Neurônio de McCulloch-Pitts



Perceptron

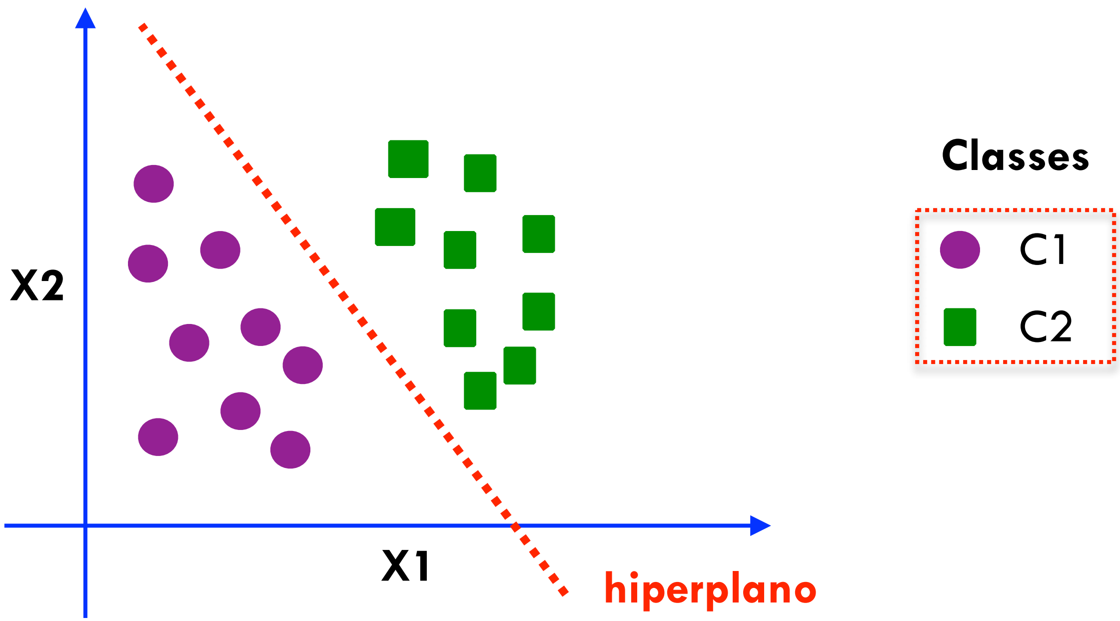
- **Objetivo:** classificar corretamente um conjunto de exemplos do dataset X em uma de duas classes, C1 ou C2



Classificação

Perceptron

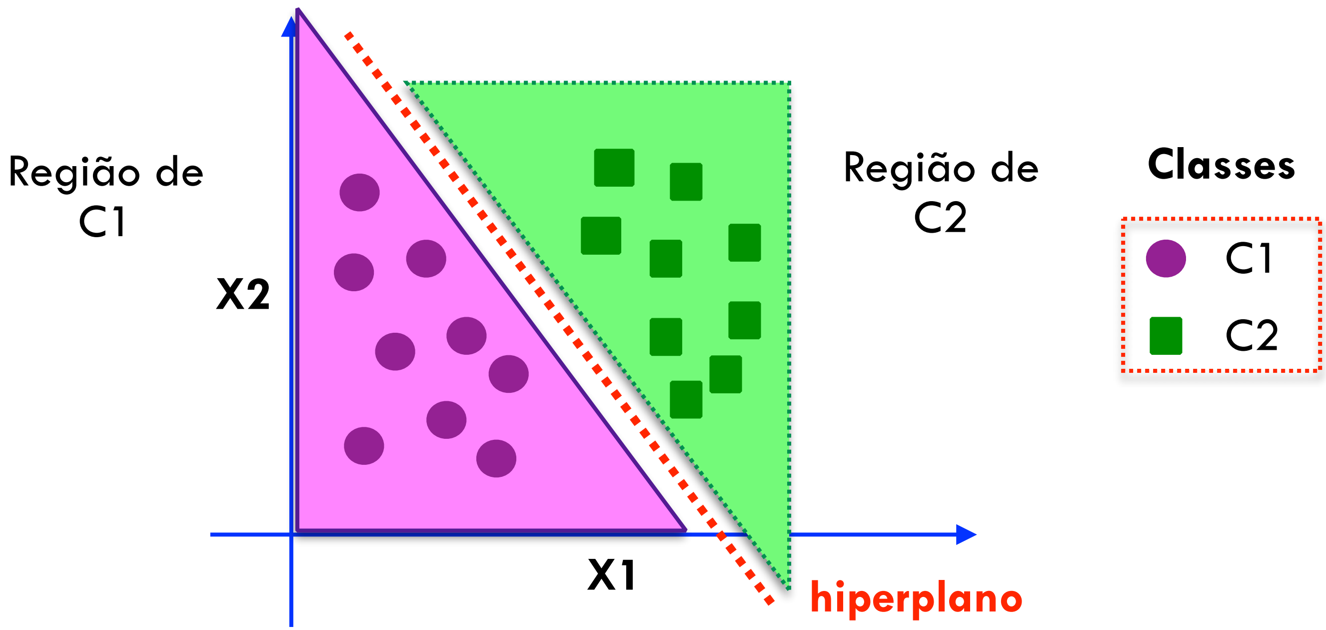
- **Objetivo:** classificar corretamente um conjunto de exemplos do dataset X em uma de duas classes, C1 ou C2



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

Perceptron

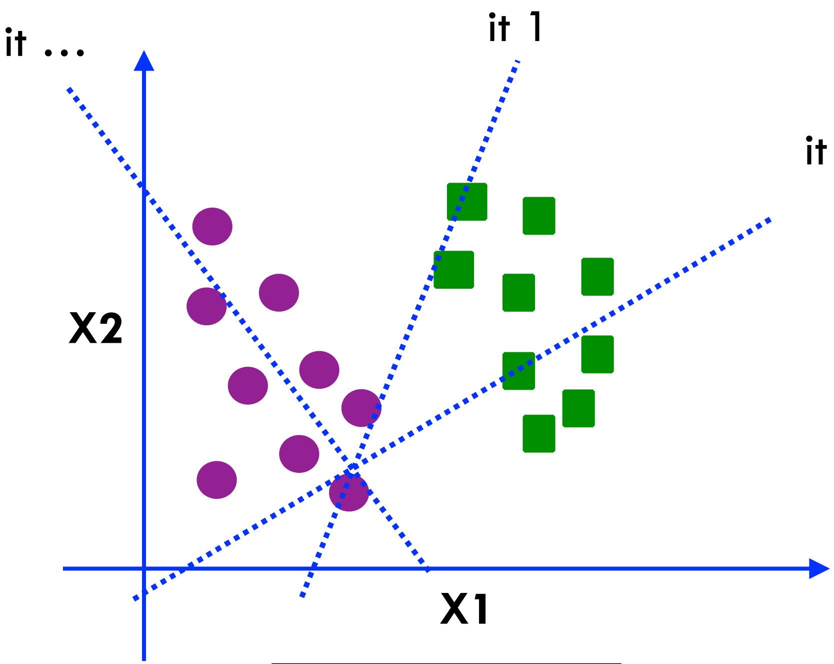
- **Objetivo:** classificar corretamente um conjunto de exemplos do dataset X em uma de duas classes, C1 ou C2



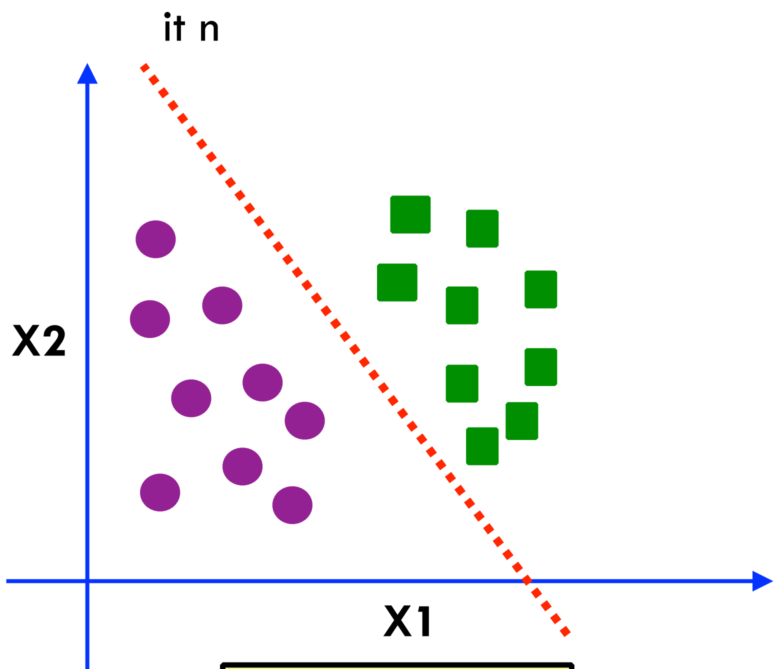
Perceptron gera um **hiperplano** (reta) separador

Perceptron

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



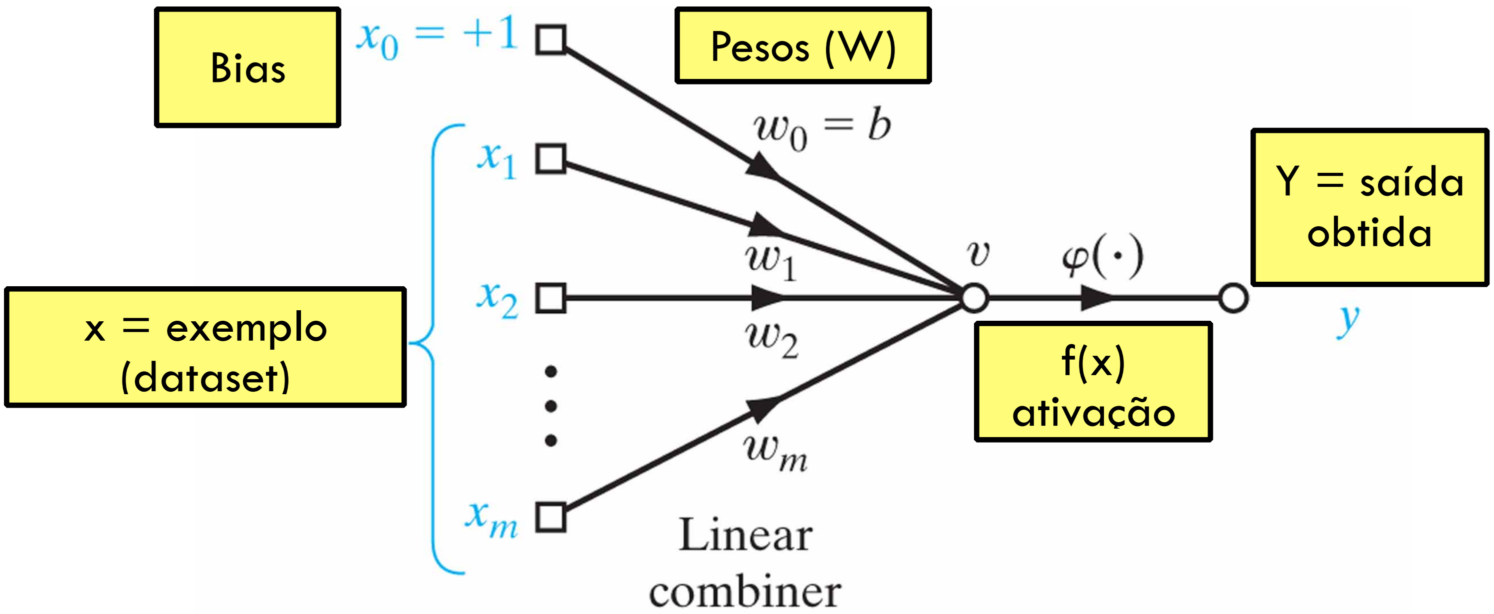
hiperplanos intermediários



hiperplano final

Roteiro

Teorema de Convergência



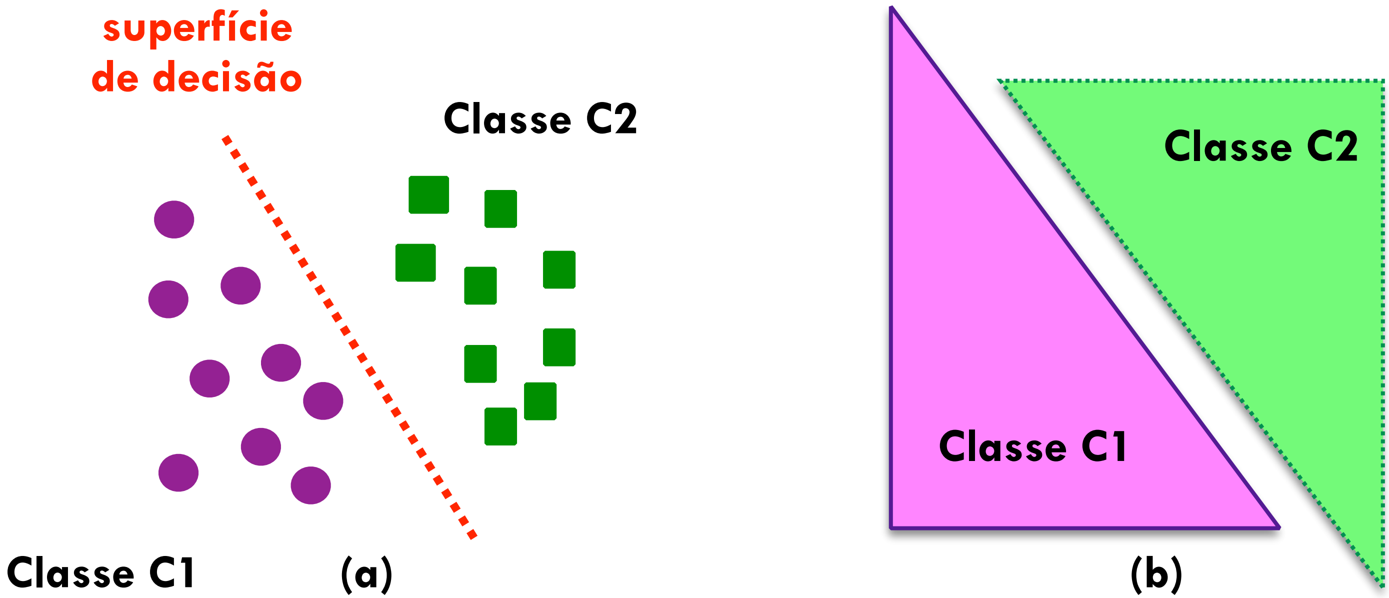
Teorema de Convergência

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

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

$\boxed{v = \text{sinal do neurônio}}$

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

- 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 sgn(.):

função sinal

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

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

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

Teorema de Convergência

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

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

Teorema de Convergência

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

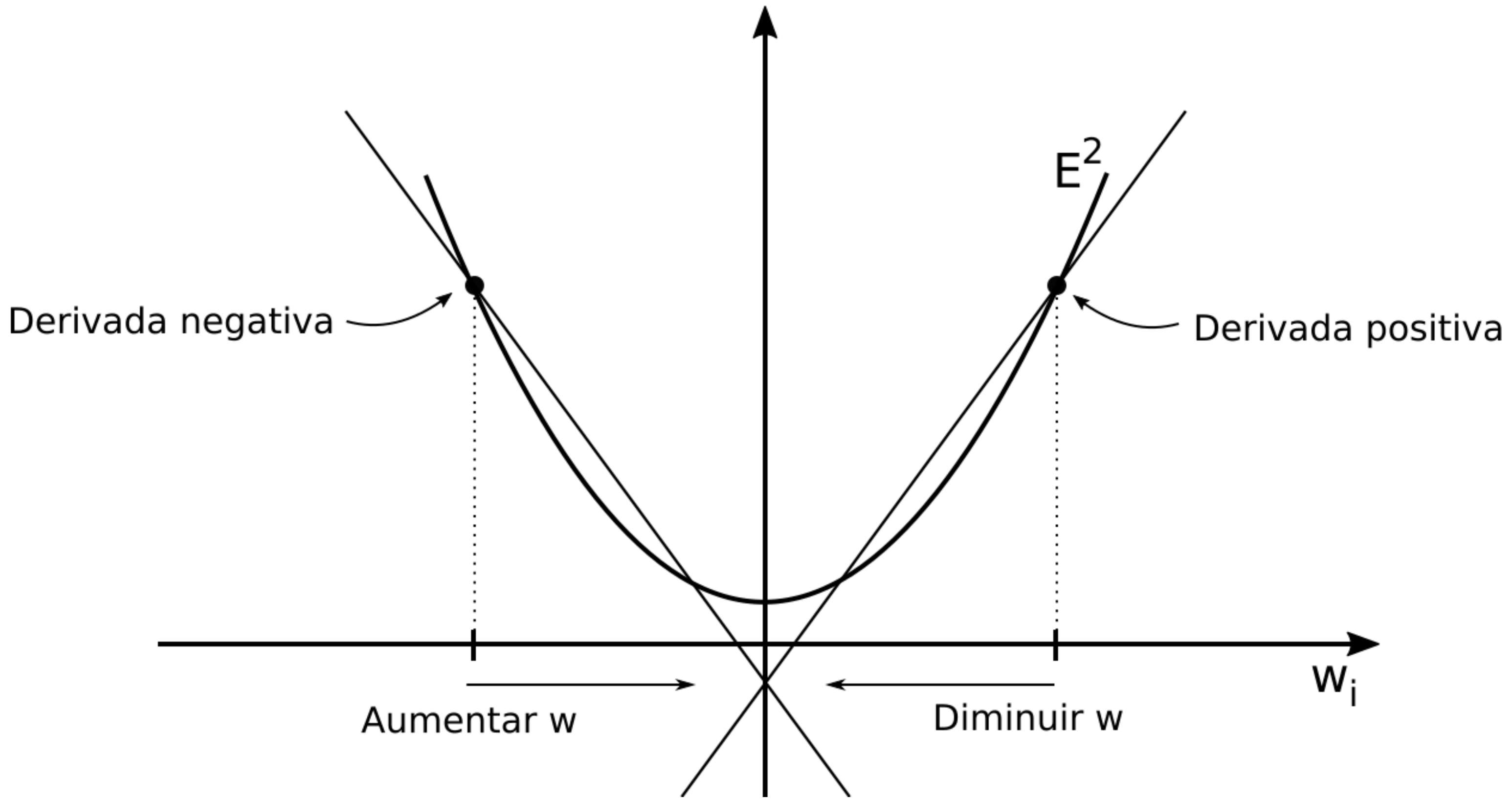
$$w(n+1) \leftarrow w(n) + \boldsymbol{\eta} * (d(n) - y(n)) * x(n)$$

d(n) - y(n) = sinal do erro (+ ou -)

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

Teorema de Convergência

$$E^2 = \left(d(n) - y(n)\right)^2 = \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^2$$



Teorema de Convergência

$$E^2 = \left(d(n) - y(n)\right)^2 = \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^2$$

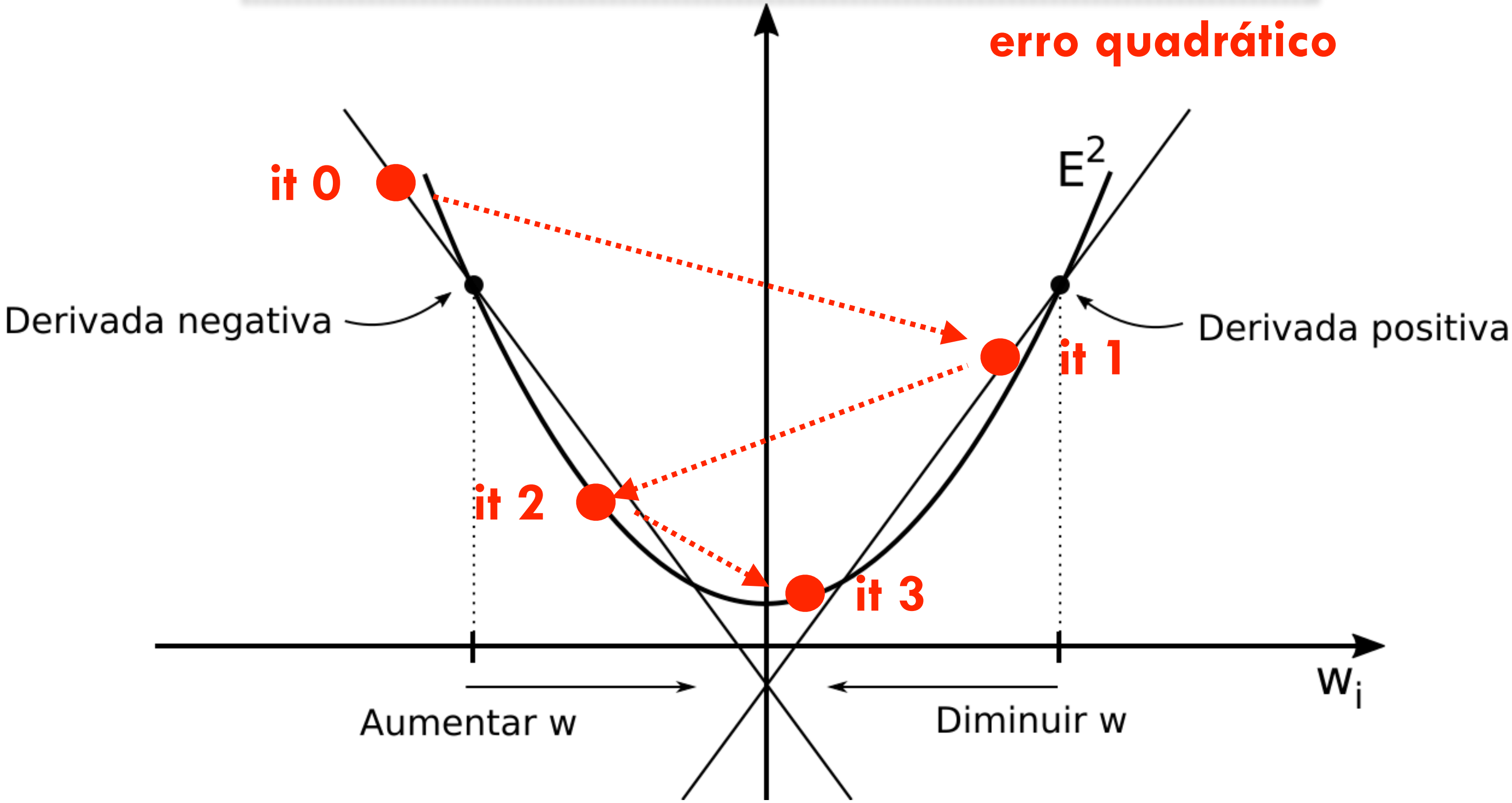
erro quadrático

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

2. $$\frac{dE^2}{dw_i} = \frac{d\left(d(n) - y(n)\right)^2}{dw_i(n)} = 2 \times \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right) \times -x_i$$

Teorema de Convergência

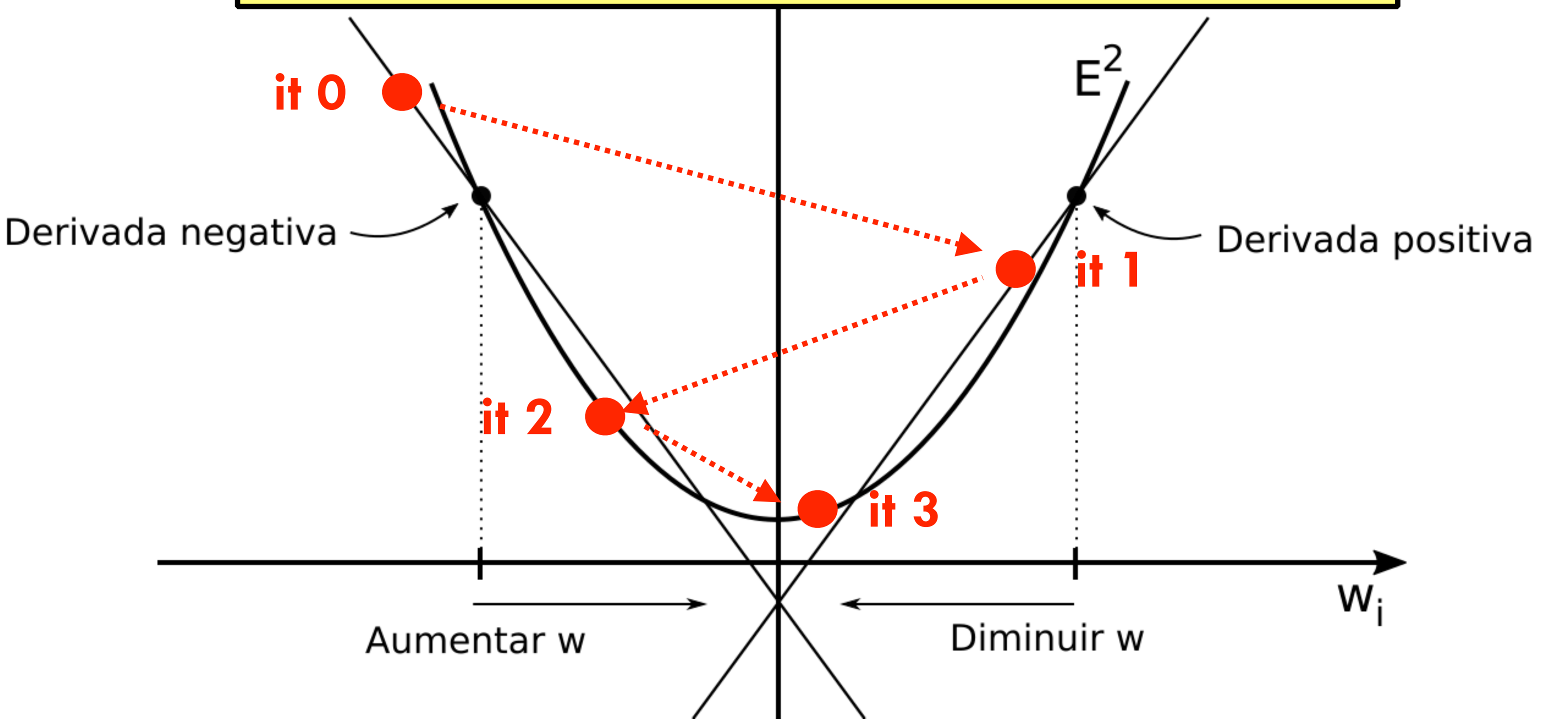
$$E^2 = \left(d(n) - y(n)\right)^2 = \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^2$$



Teorema de Convergência

$$E^2 = \left(d(n) - y(n)\right)^2 = \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^2$$

objetivo é reduzir o **erro quadrático** (E^2)



Roteiro

1. Introdução
2. Perceptron
3. Teorema de Convergência
4. **Algoritmo de Treinamento Perceptron**
5. Exemplo / Exercício
6. Síntese / Próximas Aulas
7. Referências

Algoritmo de Treinamento

- Entradas e hiper-parâmetros:
  - **X(n)**: vetor de entrada
  - **W(n)**: vetor de pesos
  - **b** : bias
  - **y(n)**: saída obtida
  - **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 | D]
    - X são os exemplos de treinamento
    - D são as saídas reais, esperadas
  - taxa de aprendizagem (**η**)
  - pesos sinápticos iniciais (**W**) [opcional]
  - número máximo de iterações para treinamento (**n.Iter**)

- **Saídas:**
  - W ajustados para todos os exemplos de treinamento
  - Epocas: numero de épocas

Algoritmo de Treinamento

Inicio algoritmo

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 ← 0)

3. Iniciar variável de controle (error ← **TRUE**)

4. Repetir enquanto (error == **TRUE** & epoca < n.Iter)

4.1 error ← FALSE

4.2 Para todas as amostras de treinamento em X, fazer:

4.2.1 Calcular o sinal do neurônio (spike)

$V = W' * X$

4.2.2 Calcular o sinal de saída do neurônio (Y)

$Y = phi(V)$

Algoritmo de Treinamento

4.2.3 Se houve erro na predição
Se Y (saida obtida) != Di (saida real):
$W = W + \boldsymbol{\eta} * (Di - Y) * X$
error ← **TRUE**

4.3 Incrementar o contador do numero de épocas
epocas ← epocas + 1
fim repetição
Fim do algoritmo

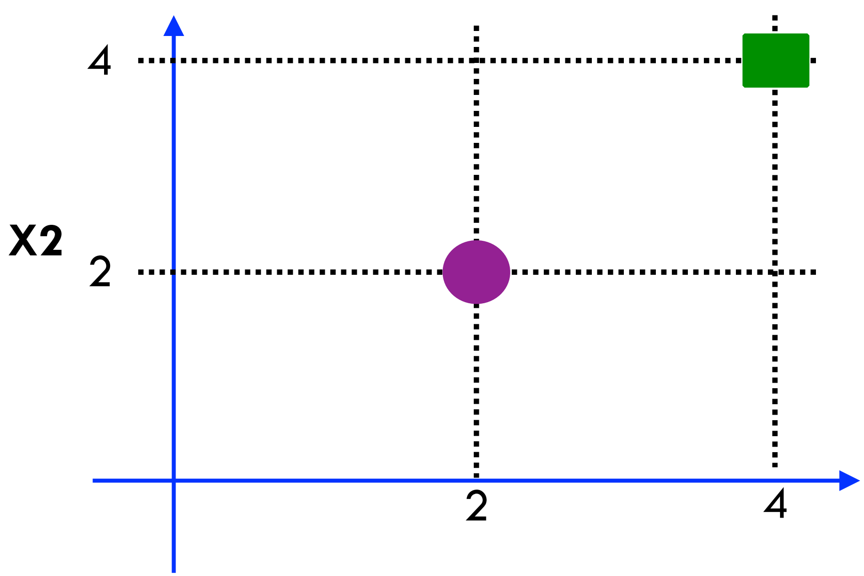
Roteiro

1. Introdução
2. Perceptron
3. Teorema de Convergência
4. Algoritmo de Treinamento Perceptron
5. Exemplo / Exercício
6. Síntese / Próximas Aulas
7. Referências

Exemplo

- Treinar o perceptron para o problema abaixo:
  - w0 = -0.5441, w1 = 0.5562, w2 = 0.4074
  - bias = -1
  - $\eta = 0.1$

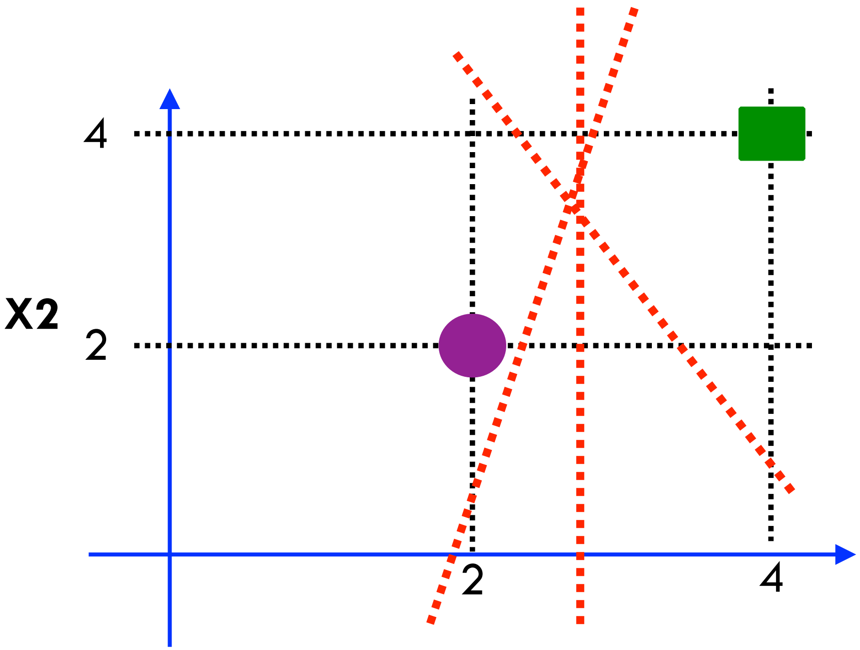
Exemplo	*X1*	*X2*	*Classe*
E1	***2***	***2***	***1***
E2	***4***	***4***	***0***



Exemplo

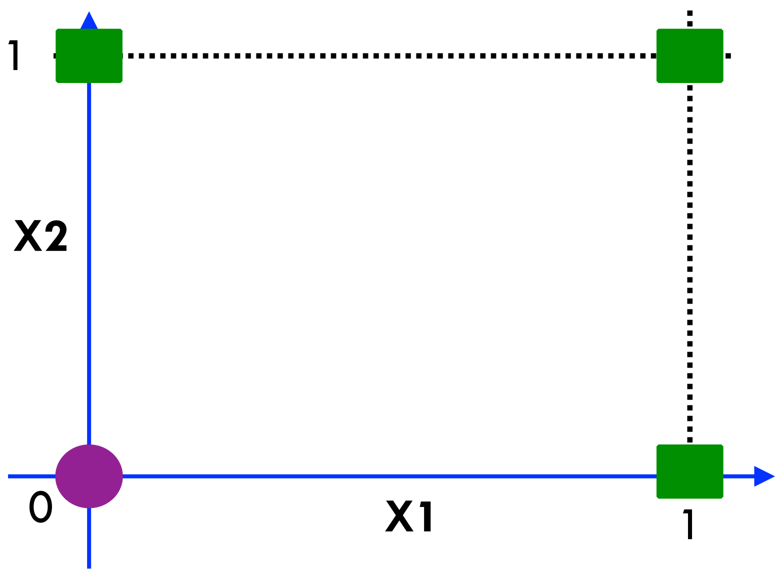
- Treinar o perceptron para o problema abaixo:
  - w0 = -0.5441, w1 = 0.5562, w2 = 0.4074
  - bias = -1
  - η = 0.1

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



Exercício

- Treinar o perceptron para reconhecer o problema lógico OR. Dados:
  - w0 = w1 = w2 = 0.5
  - bias = +1
  - $\eta = 0.1$



X1	*X2*	*D*
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*1*

Exercício

- Treinar o perceptron para reconhecer o problema lógico OR. Dados:
  - w0 = w1 = w2 = 0.5
  - bias = +1
  - $\eta = 0.1$

Hands on :)

Vamos **codificar** !

Roteiro

Síntese/Revisão

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

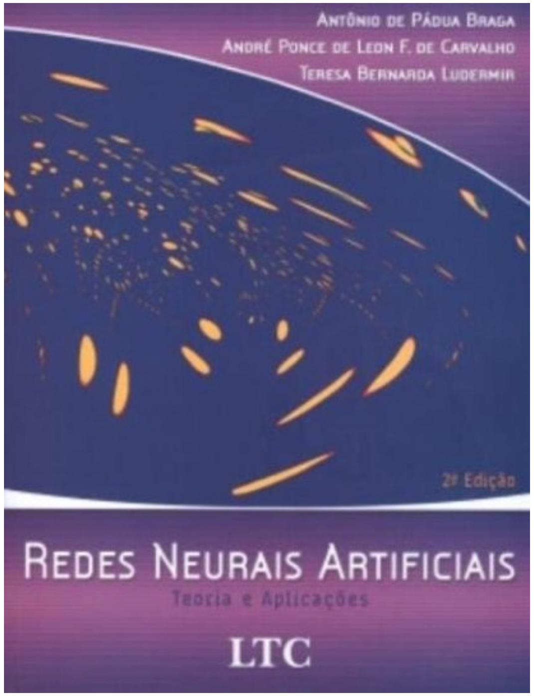
- Próximo conteúdo: MLP

Roteiro

1. Introdução
2. Perceptron
3. Teorema de Convergência
4. Algoritmo de Treinamento Perceptron
5. Exemplo / Exercício
6. Síntese / Próximas Aulas
7. Referências

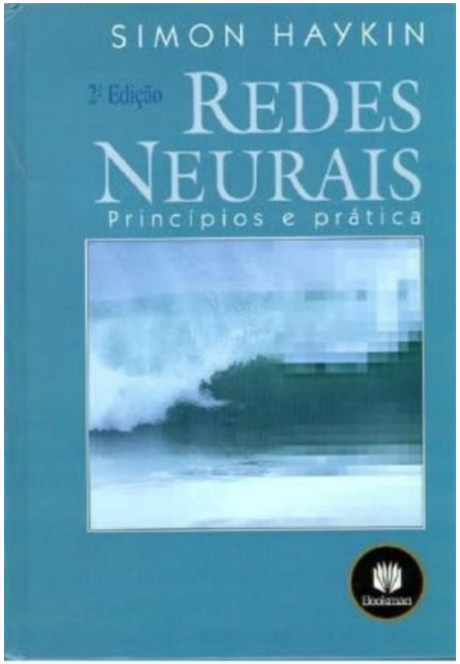
Literatura Sugerida

[Faceli et al, 2011]

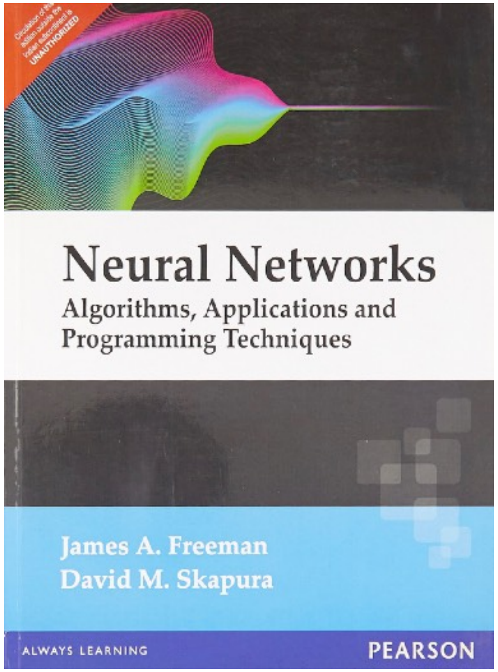


[Braga et al, 2007]

Literatura Sugerida



(Haykin, 1999)



(Freeman & Skapura, 1991)

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 w0 sendo o peso do bias:
    - slope (m) = - (w0/w2) / (w0/w1)
    - y-intercept(b) = -w0/w2