SICO70
SISTEMAS INTELIGENTES 2

Aula 03 - ADALINE

Prof. Rafael G. **Mantovani**

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Apucarana - PR, Brasil

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação

Roteiro

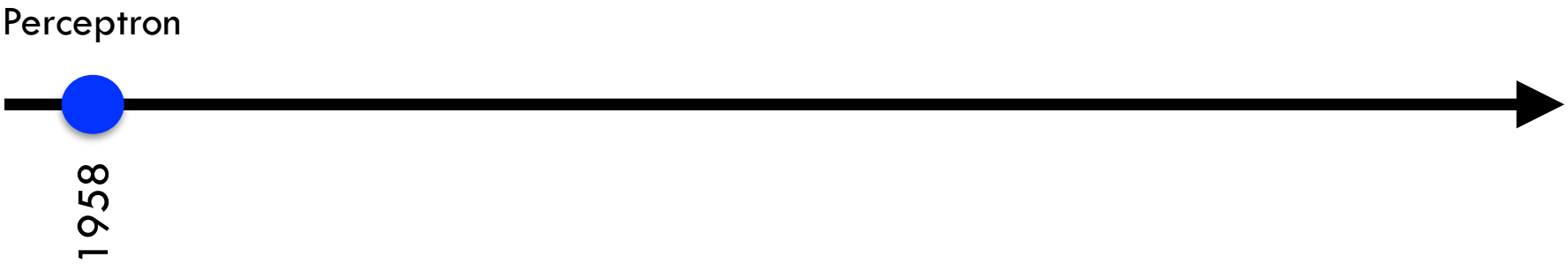
1. Introdução
2. ADALINE
3. Algoritmo de Treinamento para ADALINE
4. Exemplo / Exercício
5. Referências

Roteiro

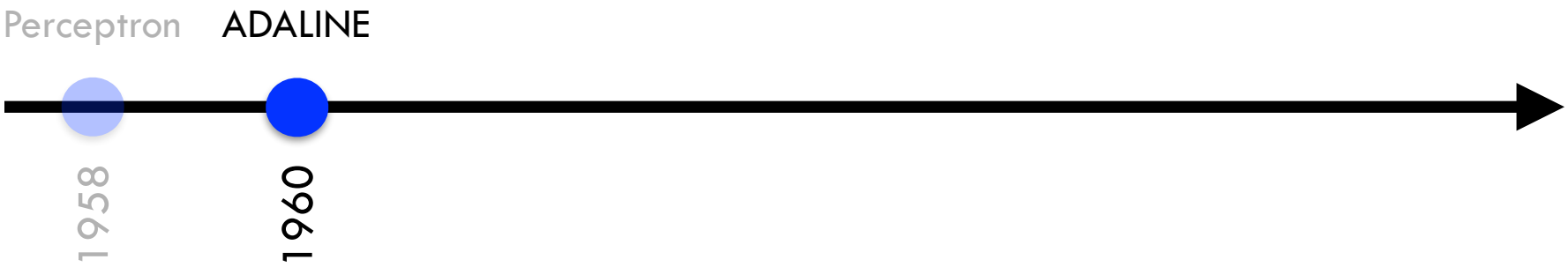
1. **Introdução**
2. **ADALINE**
3. **Algoritmo de Treinamento para ADALINE**
4. **Exemplo / Exercício**
5. **Referências**

Introdução

Introdução

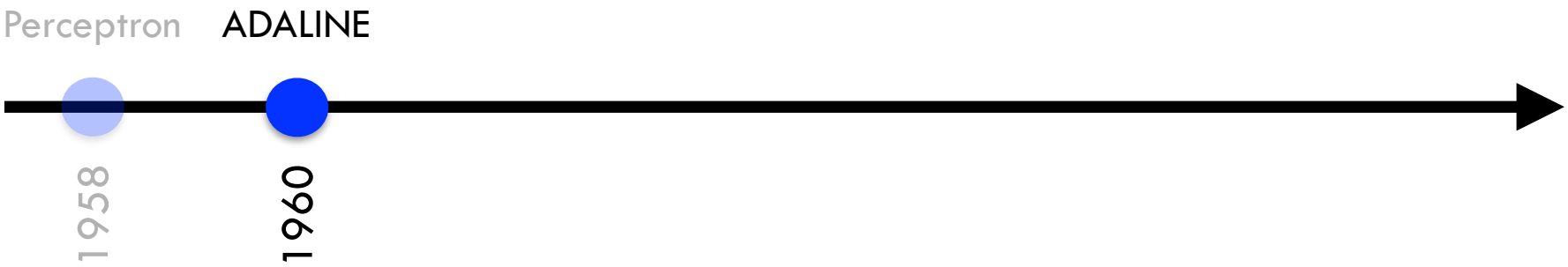


Introdução



Introdução

- **ADALINE** (Widrow & Hoff)
  - **ADA**ptive **LINE**ar neuron/element
  - Aplicações voltadas a filtros lineares



Roteiro

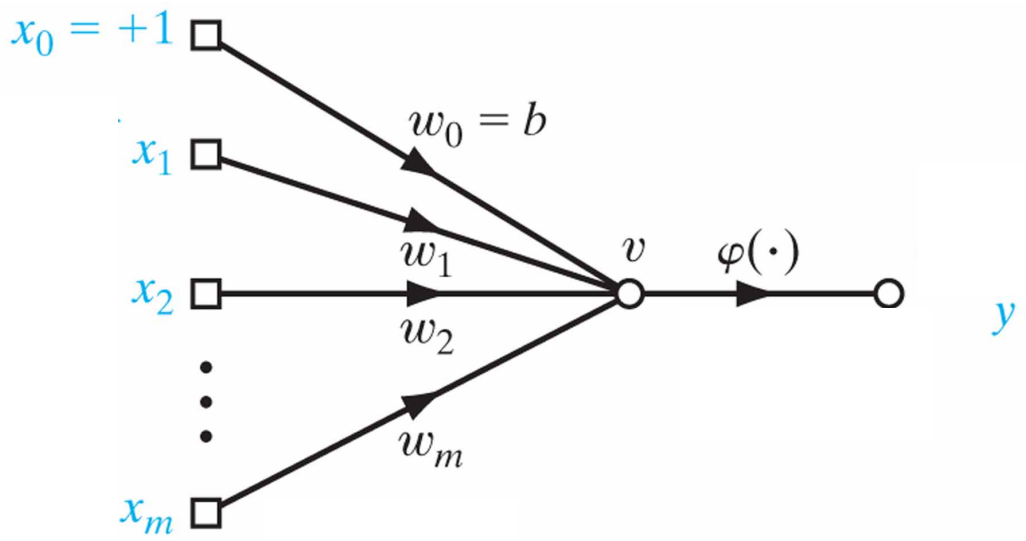
1. Introdução
2. **ADALINE**
3. Algoritmo de Treinamento para ADALINE
4. Exemplo / Exercício
5. Referências

ADALINE

- **ADALINE** → Neurônio de *McCulloch-Pitts*

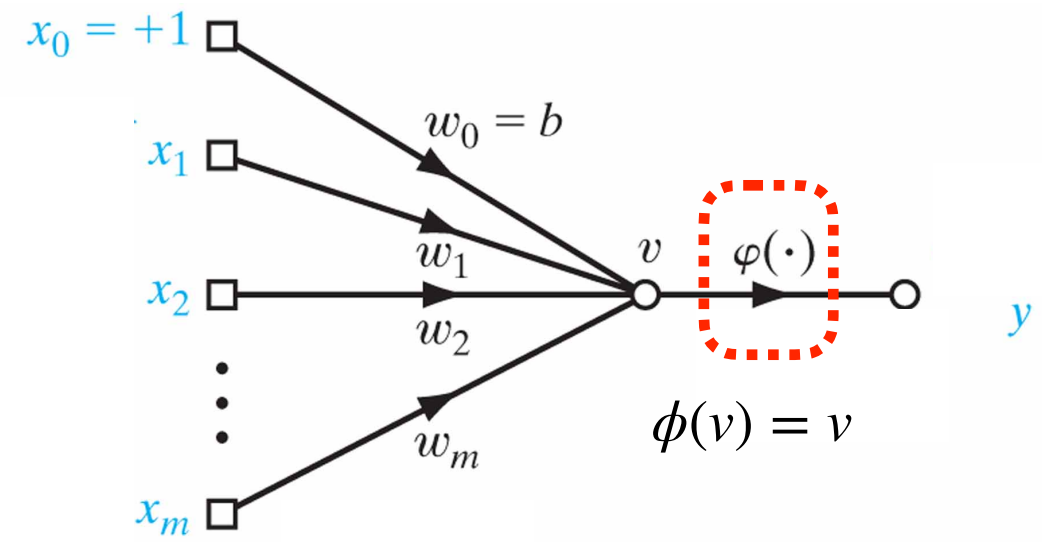
ADALINE

- **ADALINE** → Neurônio de *McCulloch-Pitts*



ADALINE

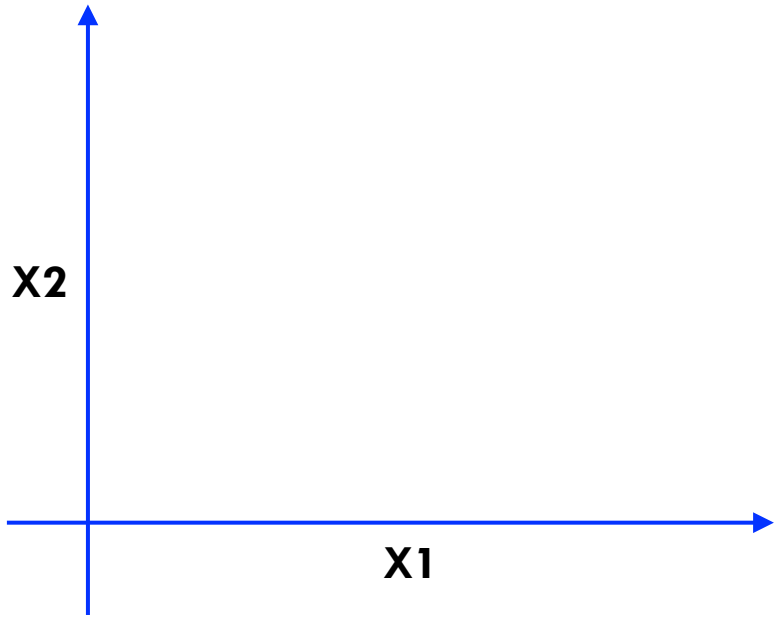
- **ADALINE** → Neurônio de *McCulloch-Pitts*



Função de ativação **linear**

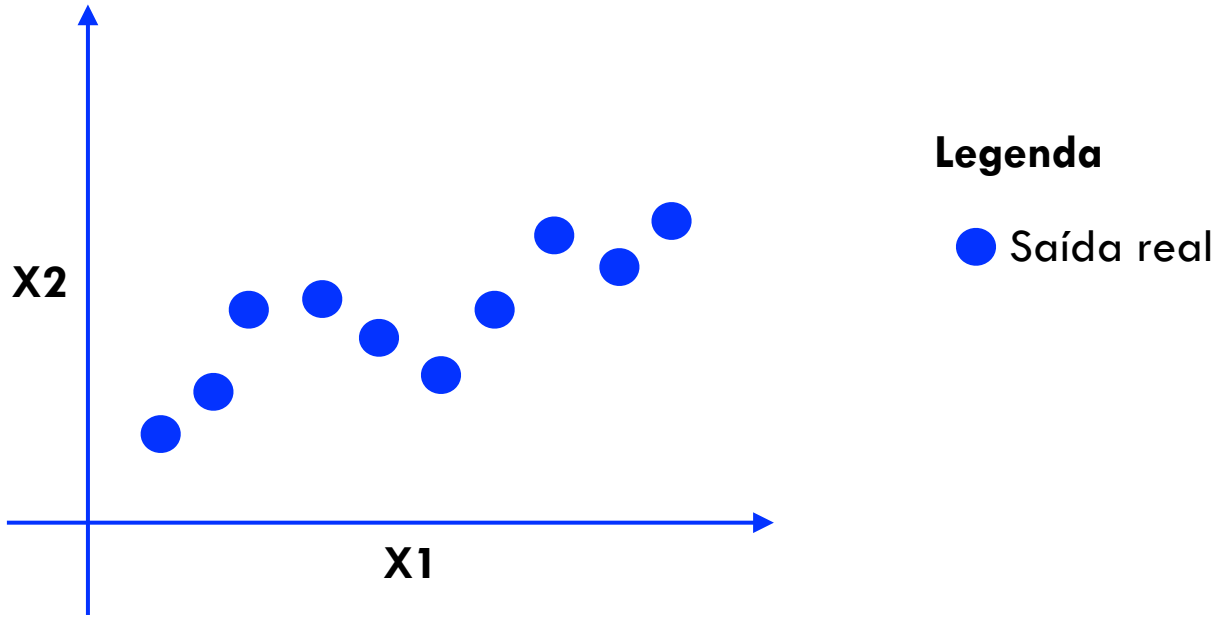
ADALINE

- **Objetivo:** Aproximação de funções (regressão)



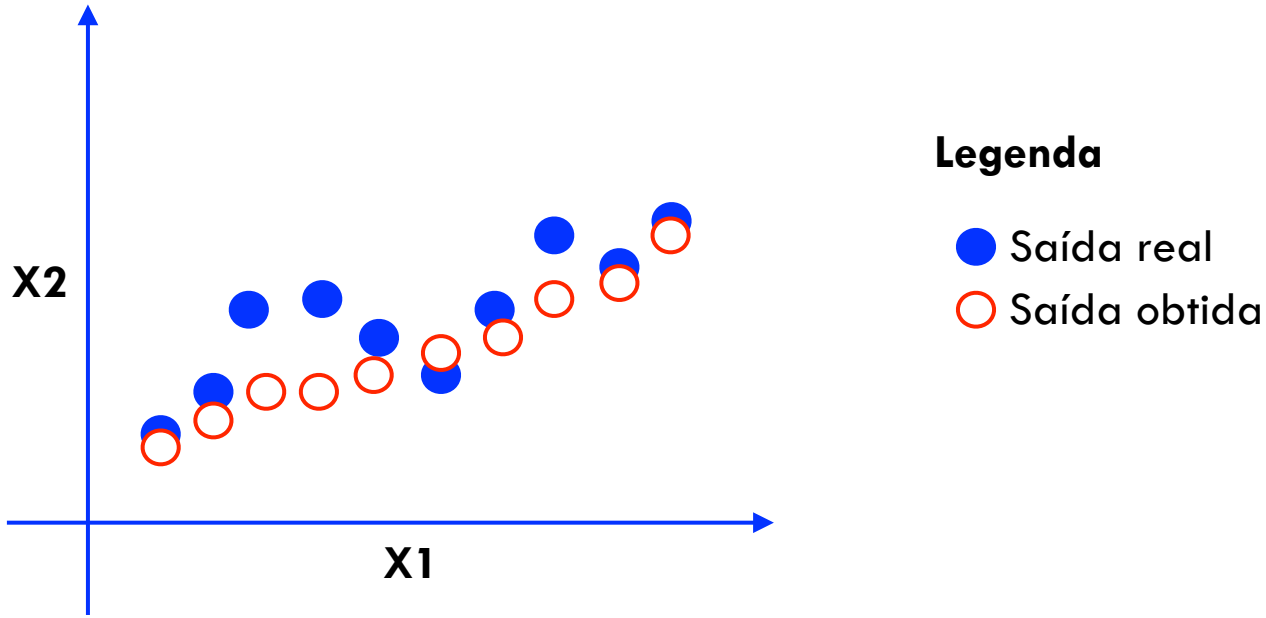
ADALINE

- **Objetivo:** Aproximação de funções (regressão)



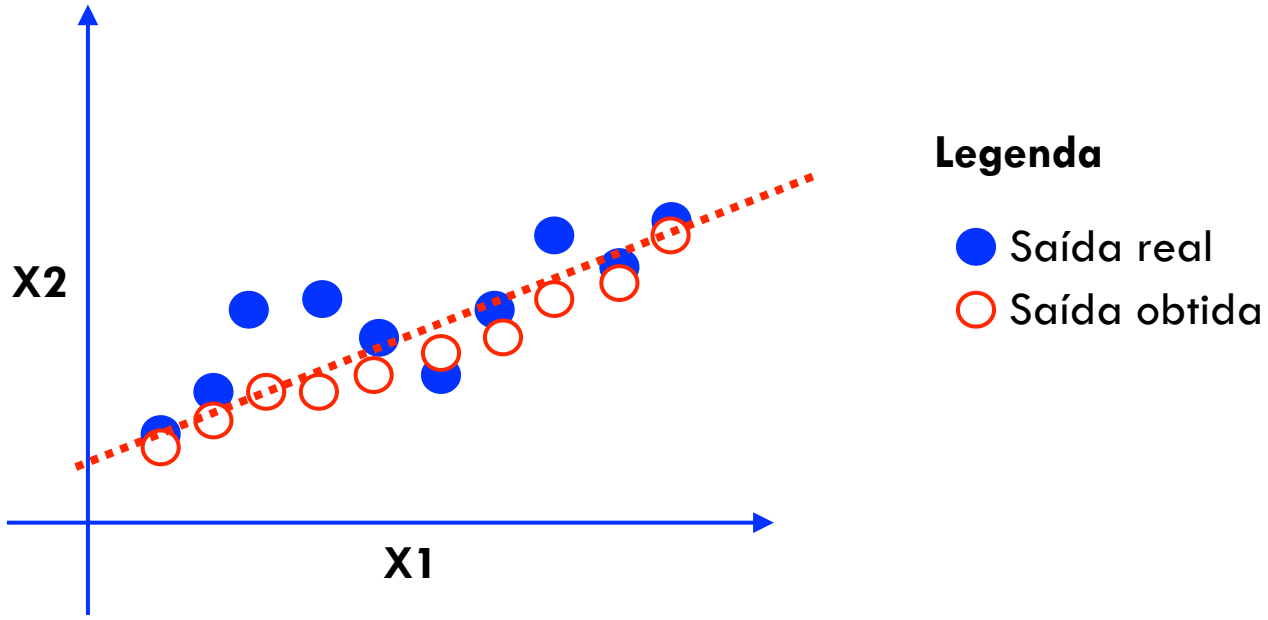
ADALINE

- **Objetivo:** Aproximação de funções (regressão)



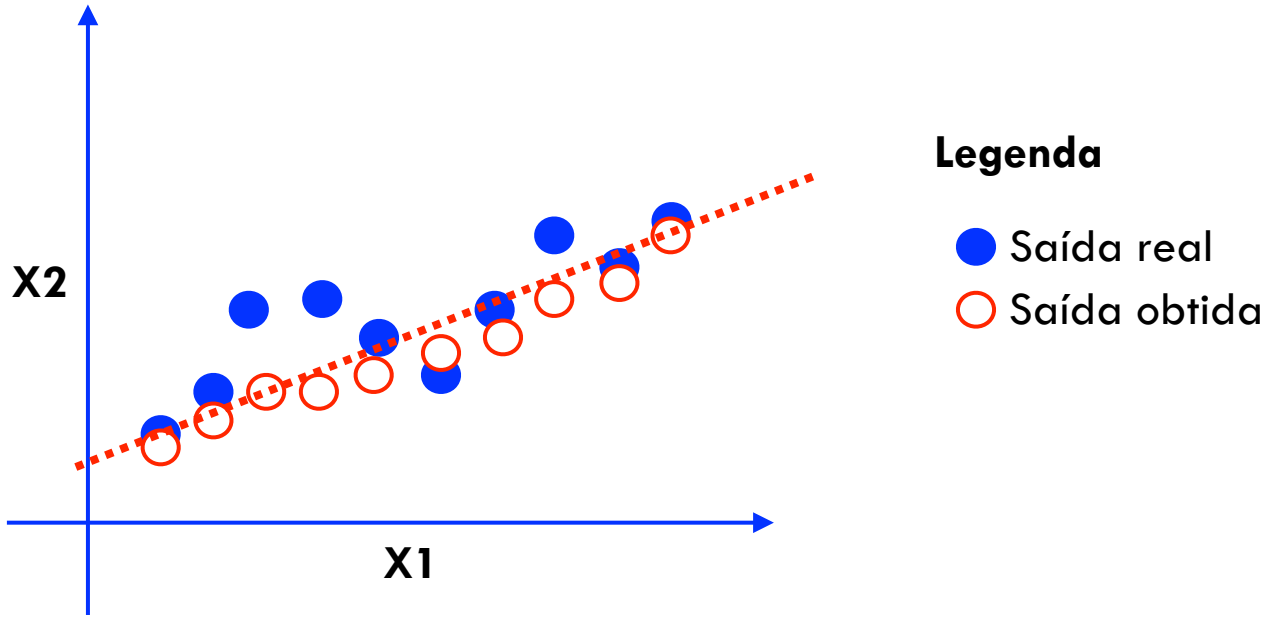
ADALINE

- **Objetivo:** Aproximação de funções (regressão)



ADALINE

- **Objetivo:** Aproximação de funções (regressão)



ADALINE **aproxima** uma função por meio de um **hiperplano**

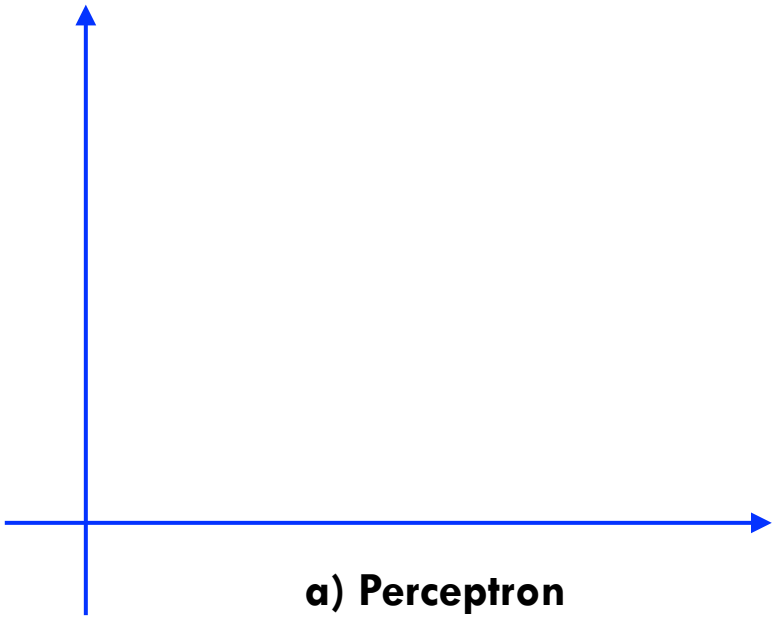
ADALINE

- Pode ser adaptada para classificação

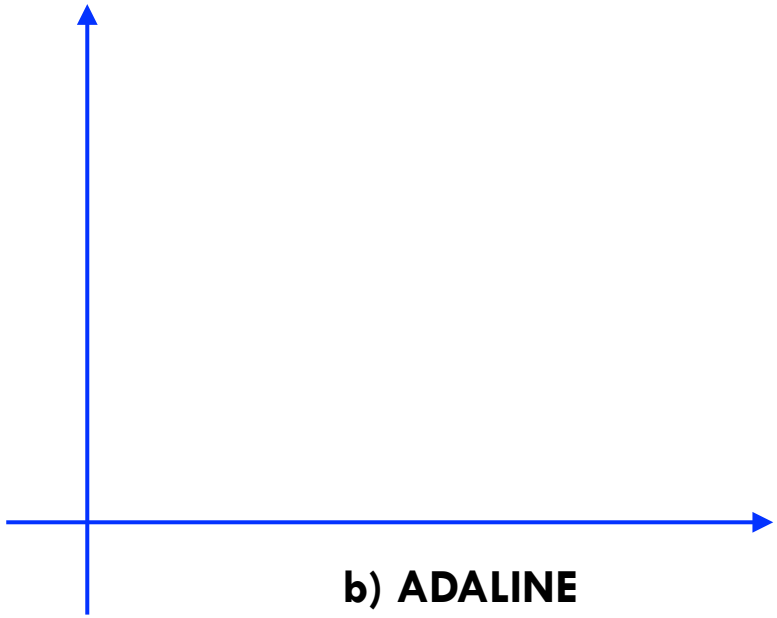
ADALINE

- Pode ser adaptada para classificação
  - **Função de ativação:**
    - Treinamento (linear): $\phi(v) = v$
    - Teste (degrau): $\phi(v) = \begin{cases} 1, v \geq 0 \\ 0, v < 0 \end{cases}$

ADALINE

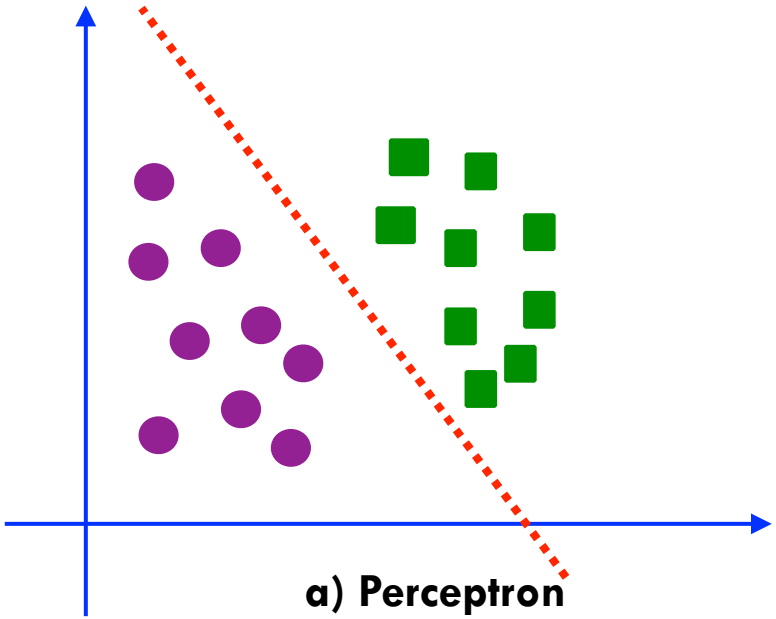


a) Perceptron

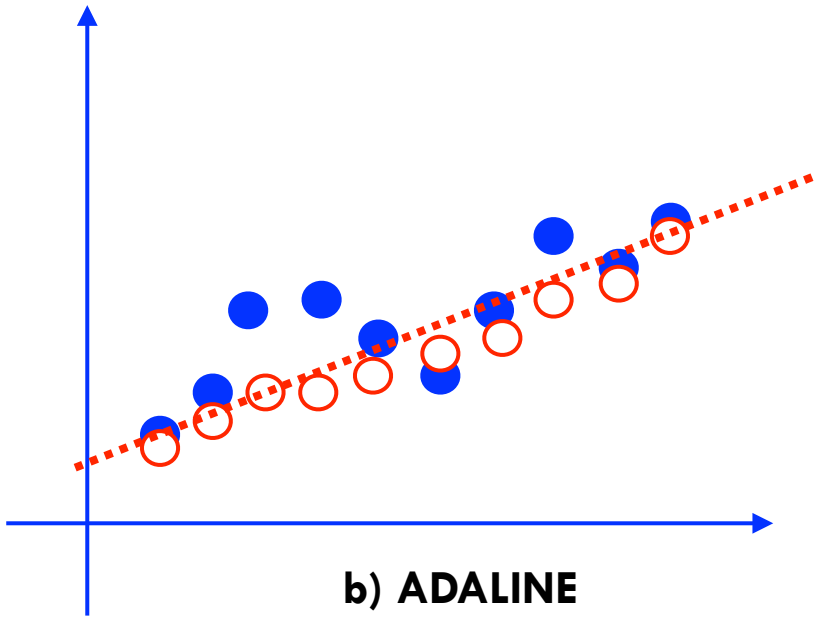


b) ADALINE

ADALINE

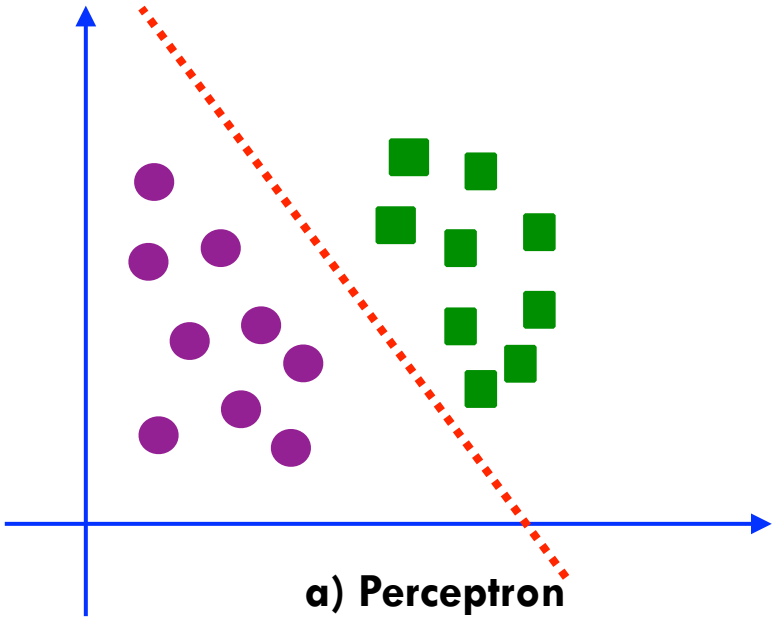


a) Perceptron

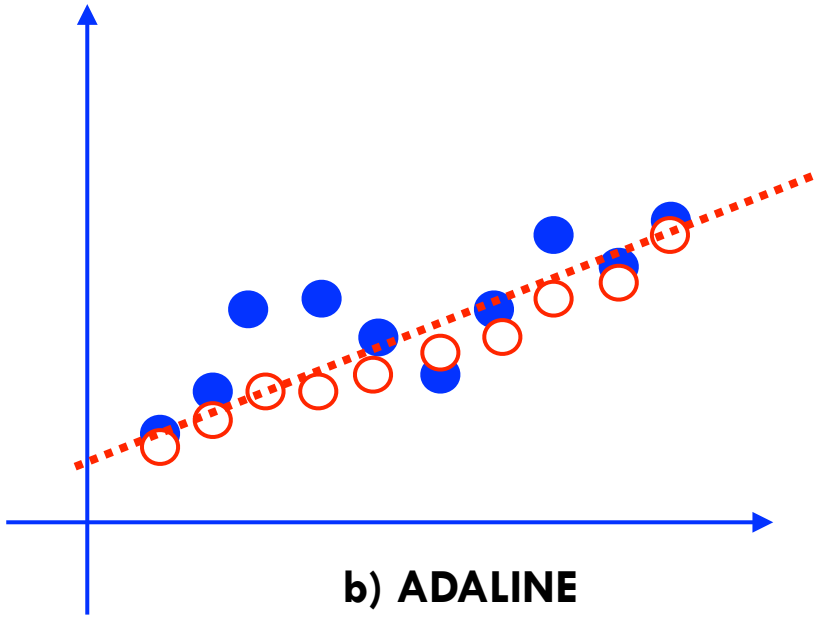


b) ADALINE

ADALINE



a) Perceptron

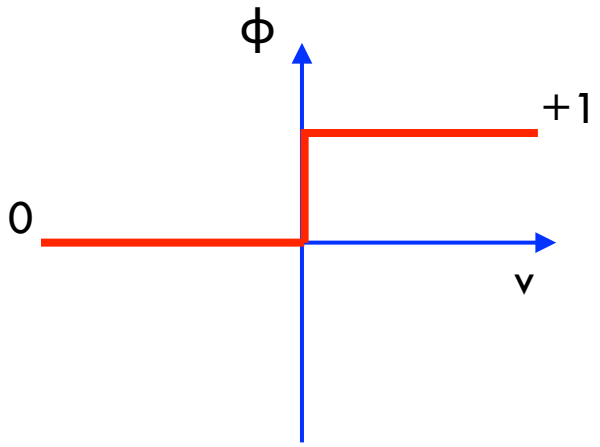


b) ADALINE

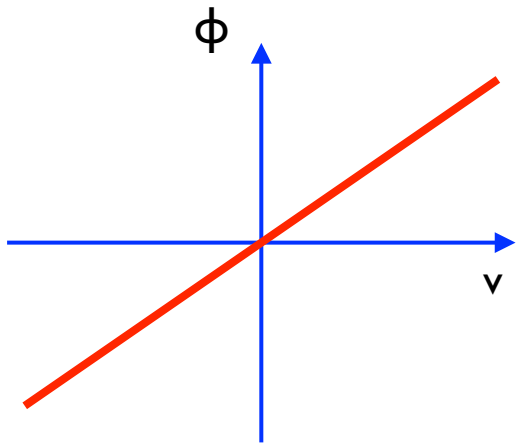
Perceptron é um separador linear (**classificação**)
ADALINE é um aproximador linear de funções (**regressão**)

ADALINE

- **Função de ativação:**



a) Perceptron



b) ADALINE

Roteiro

1. Introdução
2. ADALINE
3. **Algoritmo de Treinamento para ADALINE**
4. Exemplo / Exercício
5. Referências

Treinamento

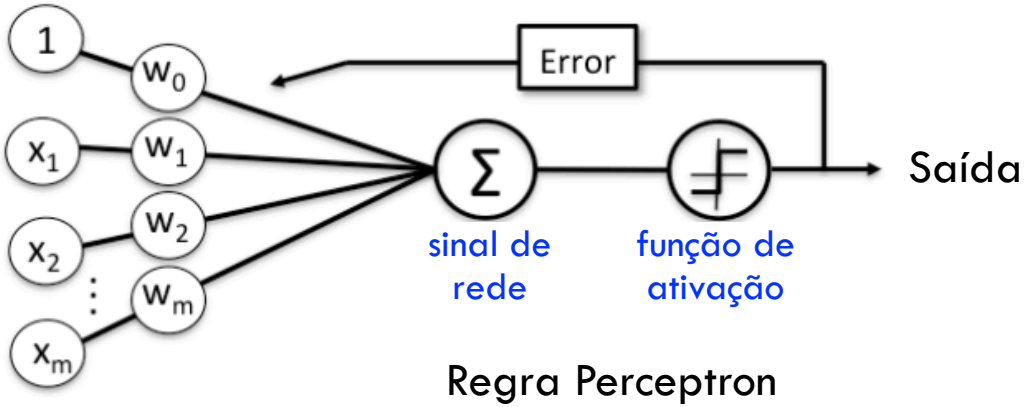
- **Aprendizado:**
  - **Estocástico:** atualiza W sempre (para todos os exemplos)
  - **Batch:** atualiza W considerando todos os exemplos de uma vez (cálculo matricial)

Treinamento

- **Aprendizado:**
  - **Estocástico:** atualiza W sempre (para todos os exemplos)
  - **Batch:** atualiza W considerando todos os exemplos de uma vez (cálculo matricial)

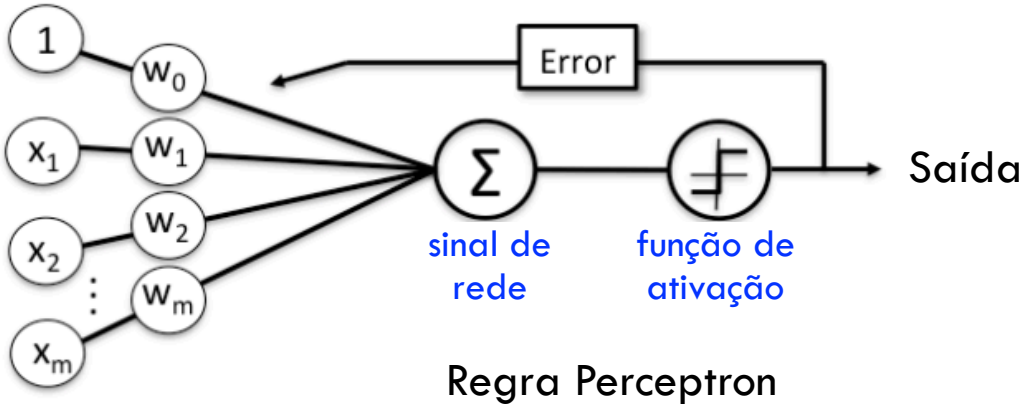
Perceptron atualiza W quando um exemplo está errado.

Treinamento

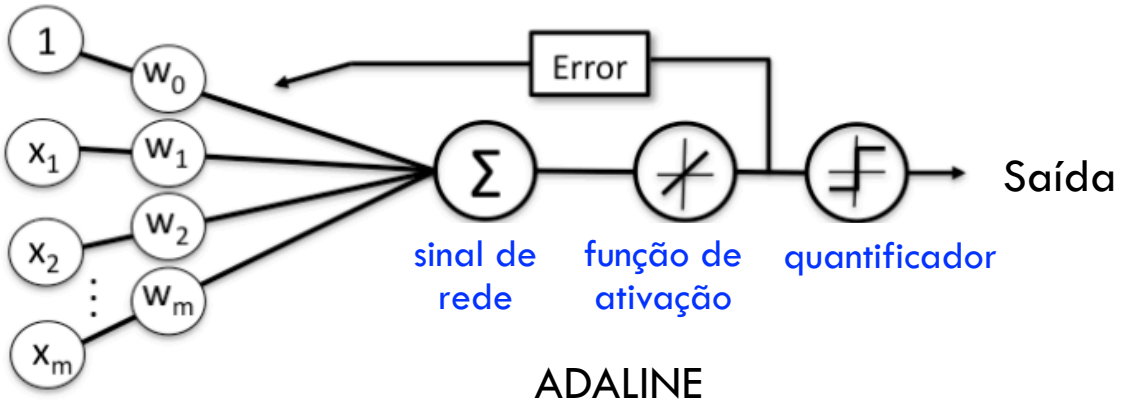


Regra Perceptron

Treinamento

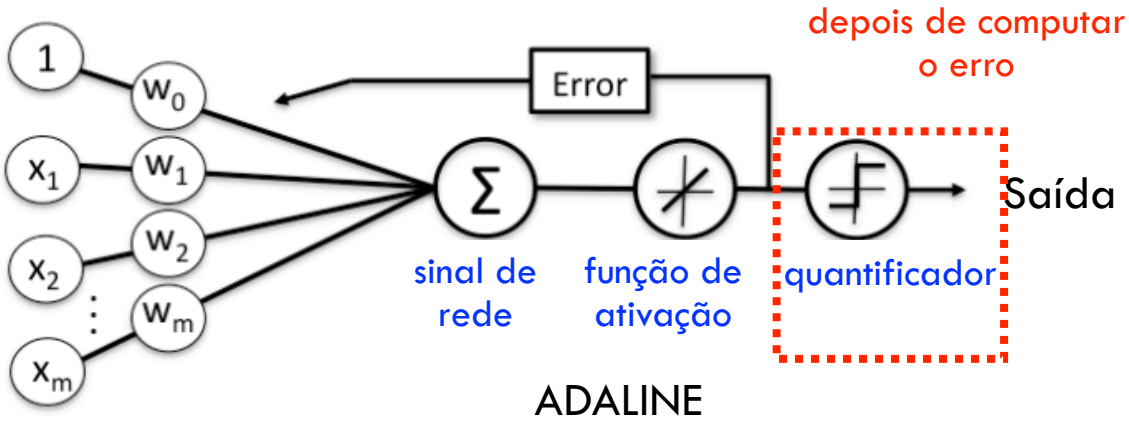


Regra Perceptron



ADALINE

Treinamento



Treinamento

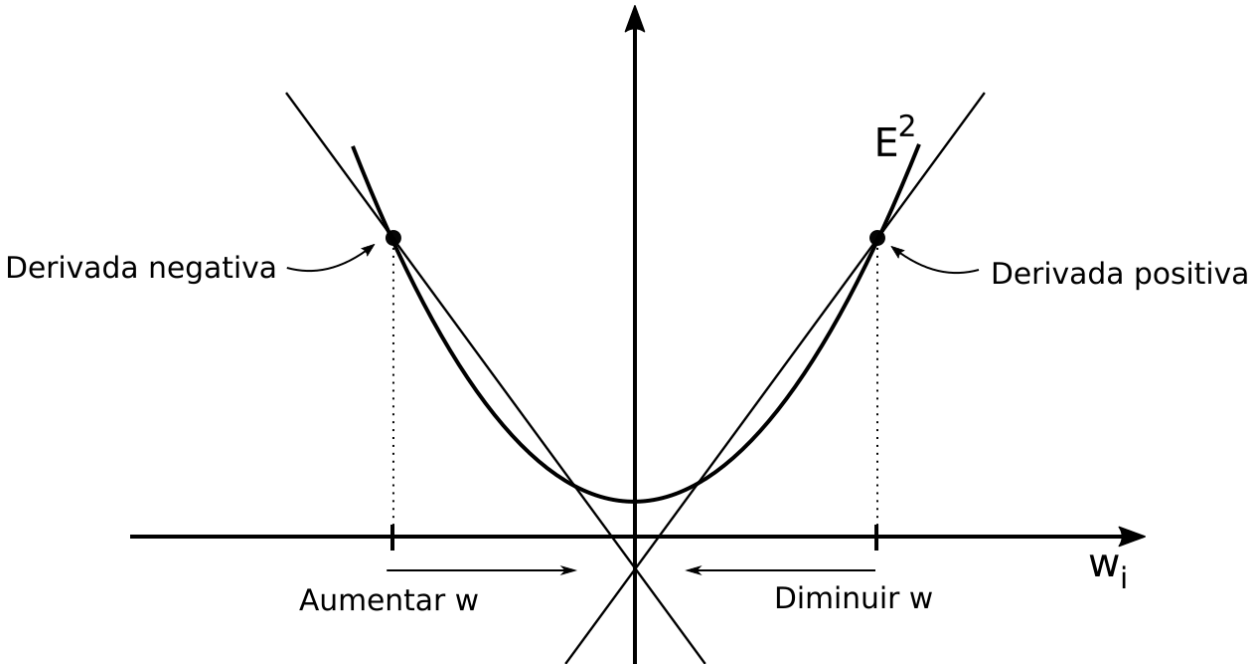
- Gradiente Descendente

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

Treinamento

- Gradiente Descendente

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

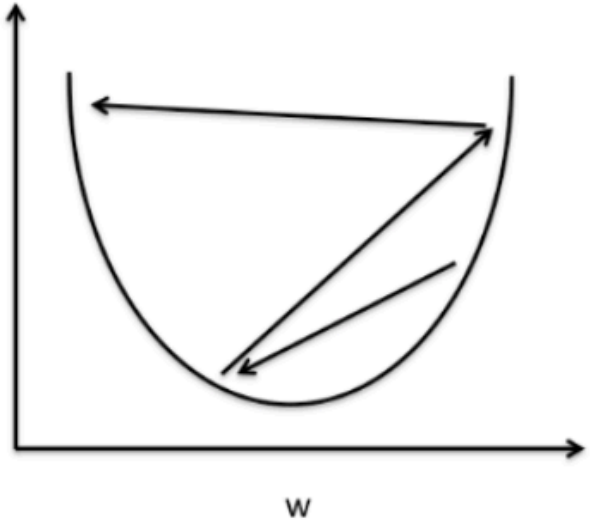


Treinamento

- Extremamente sensível ao valor de η

Treinamento

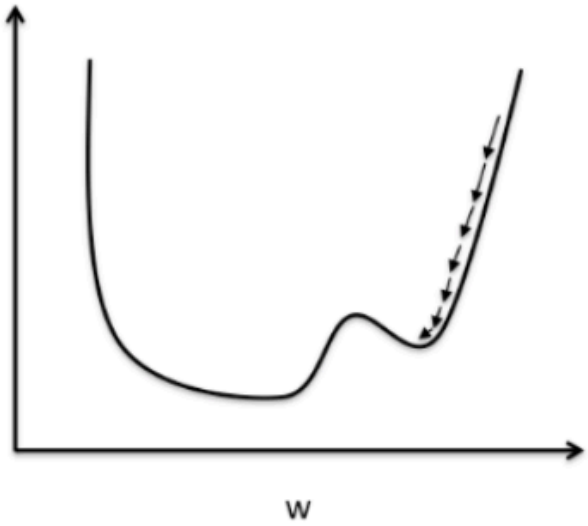
- Extremamente sensível ao valor de η



a) taxa de aprendizado alta: gradiente se perde

Treinamento

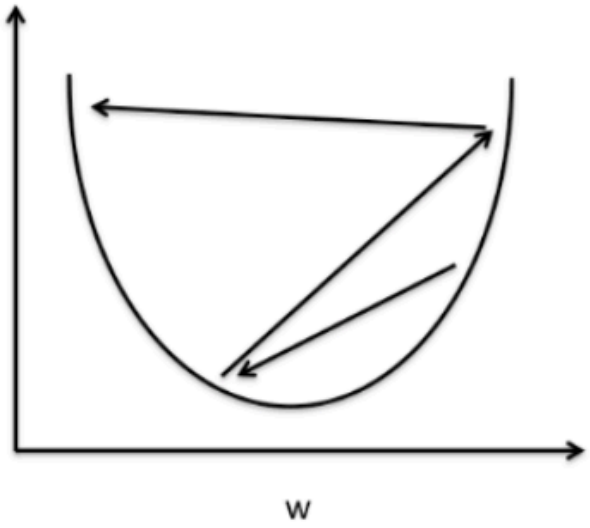
- Extremamente sensível ao valor de η



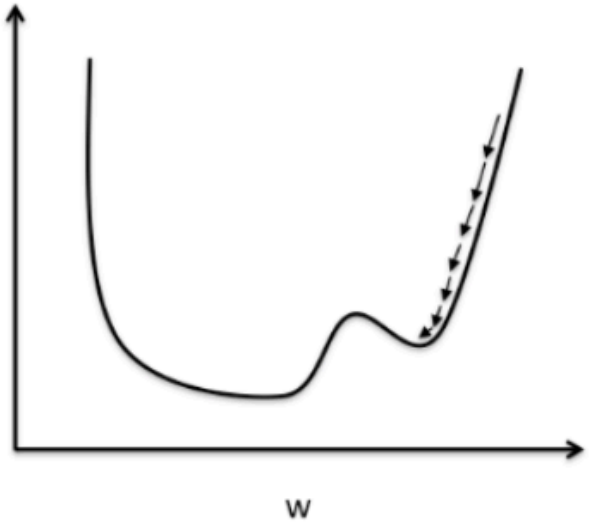
b) taxa de aprendizado pequena: demora muito, pode cair em mínimos locais

Treinamento

- Extremamente sensível ao valor de η



a) taxa de aprendizado alta: gradiente se perde



b) taxa de aprendizado pequena: demora muito, pode cair em mínimos locais

Treinamento

- Solução (convergência) → normalização dos dados

Treinamento

- Solução (convergência) → normalização dos dados

$$X_{N,std} = (X_N - \mu_N)/\sigma_N$$

Treinamento

- Solução (convergência) → normalização dos dados

$$X_{N,std} = (X_N - \mu_N)/\sigma_N$$

XN: coluna / atributo do conjunto de treinamento

Treinamento

- Solução (convergência) → normalização dos dados

$$X_{N,std} = (X_N - \mu_N)/\sigma_N$$

XN: coluna / atributo do conjunto de treinamento

uN: média dos valores do atributo N

Treinamento

- Solução (convergência) → normalização dos dados

$$X_{N,std} = (X_N - \mu_N)/\sigma_N$$

XN: coluna / atributo do conjunto de treinamento

uN: média dos valores do atributo N

σN: desvio padrão dos valores do atributo N

Treinamento

- Embora similar ao algoritmo de atualização de pesos do Perceptron, possui duas diferenças:

Treinamento

- Embora similar ao algoritmo de atualização de pesos do Perceptron, possui duas diferenças:

1. a saída obtida pela rede é um **número real,** não uma classe (categoria)

2. o ajuste sináptico é calculado baseado em **todas as amostras** de treinamento

Treinamento

- Embora similar ao algoritmo de atualização de pesos do Perceptron, possui duas diferenças:

1. a saída obtida pela rede é um **número real,** não uma classe (categoria)

2. o ajuste sináptico é calculado baseado em **todas as amostras** de treinamento

Algoritmo Online: Gradiente Descendente Estocástico (SGD)

Algoritmo Batch: Gradiente Descendente (GD)

Algoritmo de Treinamento

- **Entradas:**

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 (**max.epochs**)
  - tolerância do erro quadrático médio (**error.tol**)

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 (**max.epochs**)
  - tolerância do erro quadrático médio (**error.tol**)

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

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

```
// Inicialização
```

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

// Inicialização

1. Iniciar o vetor **W** com valores aleatórios pequenos, se **W** não for fornecido
2. Iniciar o contador de número de épocas (**épocas** ← **0**)
3. Normalizar os atributos de X

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

// Inicialização

1. Iniciar o vetor **W** com valores aleatórios pequenos, se **W** não for fornecido
2. Iniciar o contador de número de épocas (**épocas** ← **0**)
3. Normalizar os atributos de X
4. **Repetir** enquanto (error < **tol.error** & epoca < **n.Iter**)
5. **Para** todas as amostras de treinamento em **X (em ordem aleatória)**, fazer:

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

```
   // Inicialização
1. Iniciar o vetor W com valores aleatórios pequenos, se W não for fornecido
2. Iniciar o contador de número de épocas (épocas ← 0)
3. Normalizar os atributos de X
4. Repetir enquanto (error < tol.error & epoca < n.Iter)
5.     Para todas as amostras de treinamento em X (em ordem aleatória), fazer:
6.         V = W' * X  //Calcular o sinal do neurônio (spike)
7.         Y = phi(V)  // Ativação Linear
```

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

```
   // Inicialização
1. Iniciar o vetor W com valores aleatórios pequenos, se W não for fornecido
2. Iniciar o contador de número de épocas (épocas ← 0)
3. Normalizar os atributos de X
4. Repetir enquanto (error < tol.error & epoca < n.Iter)
5.     Para todas as amostras de treinamento em X (em ordem aleatória), fazer:
6.         V = W' * X  //Calcular o sinal do neurônio (spike)
7.         Y = phi(V)  // Ativação Linear
8.         // Atualiza os pesos sinápticos da rede para todos os exemplos (SEMPRE)
9.         W = W + η * (D - Y) * X
10.    Fim Para.
```

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

11. epocas ← epocas + 1 // *Incrementar o contador do numero de épocas*

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

```
11.   epocas ← epocas + 1 // Incrementar o contador do numero de épocas
12.   // Computar o erro quadrático médio da época (MSE) usando todas instâncias
13.   error ← (1/N) * Σ (D - X * W)²
```

Pseudocódigo

ADALINE_Online(X, W, tol.error, max.epochs):

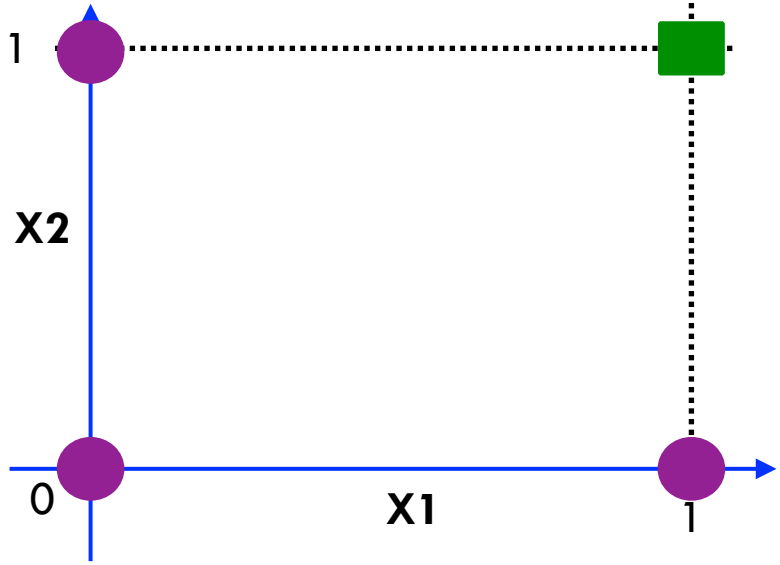
```
11.     epocas ← epocas + 1 // Incrementar o contador do numero de épocas
12.     // Computar o erro quadrático médio da época (MSE) usando todas instâncias
13.     error ← (1/N) * Σ (D - X * W)²
14.   Fim Repita.
15. Fim Pseudocódigo.
```

Roteiro

1. Introdução
2. ADALINE
3. Algoritmo de Treinamento para ADALINE
4. Exemplo / Exercício
5. Referências

Exercício

- Treinar o ADALINE para reconhecer o problema lógico AND usando Gradiente Descendente Estocástico. Dados:
  - w0 = w1 = w2 = 0.5
  - bias = +1
  - η = 0.1



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

Exercício

- Treinar o ADALINE para reconhecer o problema lógico AND usando Gradiente Descendente Estocástico. Dados:
  - w0 = w1 = w2 = 0.5
  - bias = +1
  - η = 0.1

Hands On! Hora de codar!

Algoritmo Online: Gradiente Descendente Estocástico (SGD)

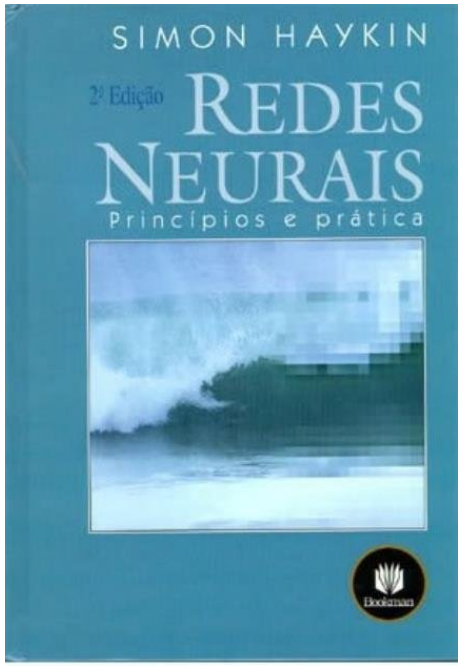
Síntese/Revisão

- ADALINE
  - um neurônio de McCulloch Pitts
  - bias
  - função de ativação **linear**
- Gradiente Descendente (Estocástico)

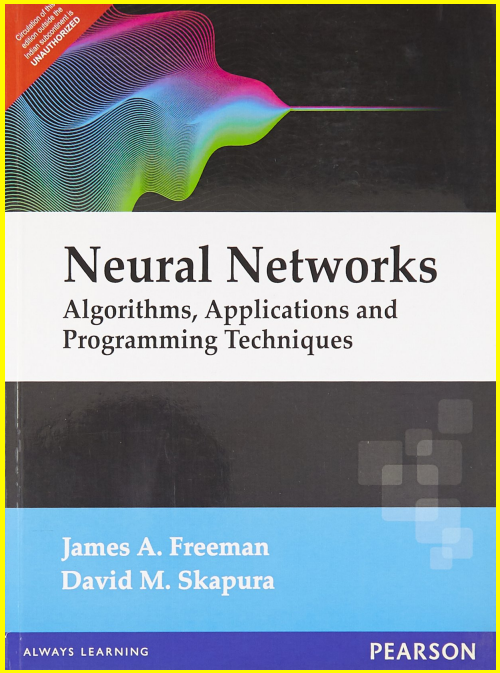
Roteiro

1. Introdução
2. ADALINE
3. Algoritmo de Treinamento para ADALINE
4. Exemplo / Exercício
5. Referências

Literatura Sugerida



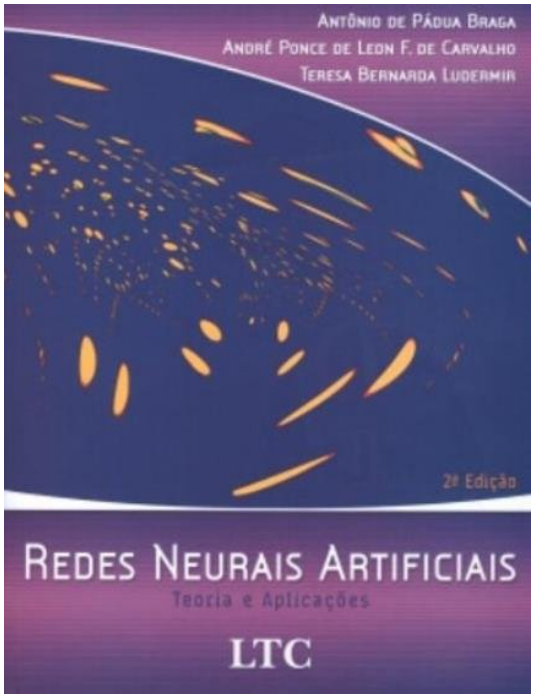
(Haykin, 1999)



(Freeman & Skapura, 1991)

Literatura Sugerida

[Faceli et al, 2011]



[Braga et al, 2007]

Perguntas?

Prof. Rafael G. Mantovani

rgmantovani@gmail.com