

SCC0270/SCC5809 - Redes Neurais

Aula 1 - Parte II - *Perceptron*

Profa. Dra. Roseli Aparecida Francelin Romero
SCC - ICMC - USP

2018

Sumário

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)

Perceptron

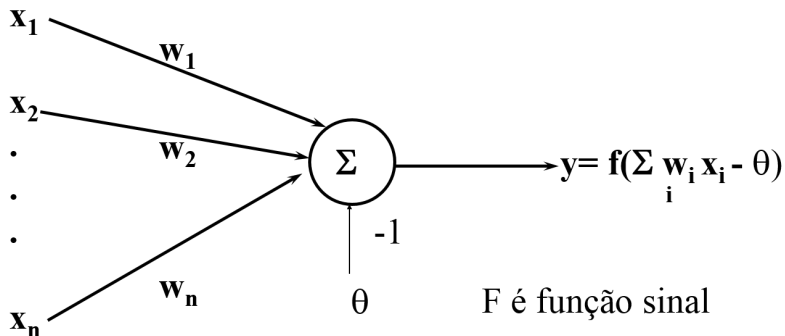


Figura 1: Modelo para representação do *Perceptron*.

Perceptron

- O algoritmo usado para ajustar os parâmetros livres desta rede apareceu num processo de aprendizado desenvolvido por Rosenblatt (1958, 1962).
- Ele provou que se os padrões usados para treinar são **linearmente separáveis**, então o algoritmo converge e a superfície de decisão tem a forma de um hiperplano entre duas classes.

Perceptron

- É constituído de apenas 1 neurônio e, como tal, limita-se a classificar padrões envolvendo apenas 2 classes, que devem ser linearmente separáveis.
- A regra de decisão é designar x à classe \mathcal{C}_1 , se a saída é $y = +1$, ou à classe \mathcal{C}_2 , se a saída é $y = -1$.
- Existem duas regiões separadas pelo hiperplano:
 - $\sum w_i x_i - \theta = 0$
- Se o espaço for o \mathbb{R}^2 , a região de separação é uma reta.

Sumário

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)

Diagram illustrating a single neuron model:

- Inputs: $x_0, x_1, x_2, \dots, x_n$
- Weights: $w_0, w_1, w_2, \dots, w_n$
- Summation node: Σ
- Output: $y = f(\Sigma w_i x_i - \theta)$
- Threshold: θ (labeled as $w_0 = \theta$ (threshold))
- Function: F é função sinal (F is a sign function)

8 / 54

Modelos de neurônios

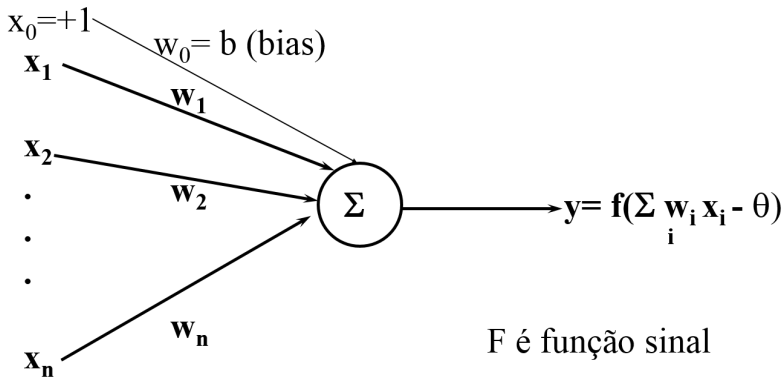


Figura 3: Modelo de um neurônio com $x_0 = +1$.

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)



Características básicas

- **Regra de propagação:** $y_j = \text{sgn}(\sum_i x_i w_{ij})$
- **Função de ativação:** função sinal
- **Topologia:** uma única camada de processadores.
- **Algoritmo de aprendizado:** $\Delta w_{ij} = \eta x_i (t_j - y_j)$
 - (é do tipo supervisionado)
- **Valores de entrada/saída:** binários $\rightarrow t = 1$ ou $t = -1$

Finalidade do termo *bias*

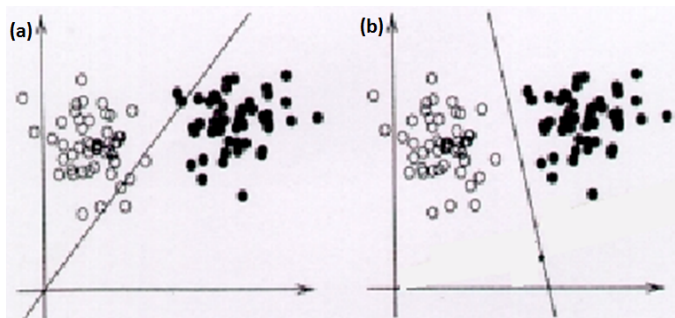


Figura 4: Hiperplano obtido: (a) sem *bias*; (b) com *bias*.

- $\sum_i x_i w_{ij} = 0 \rightarrow$ define um hiperplano passando pela origem.
- $\sum_i x_i w_{ij} + \theta_i = 0 \rightarrow$ desloca o hiperplano da origem.

Sumário

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)

Adaline

- O processo adaptativo do Adaline consiste em utilizar a função de ativação *hard limiter* (saída $+1$ ou -1) e minimizar os pesos usando o algoritmo LMS.

Regra Delta - LMS

- 1 Iniciar os pesos sinápticos com valores randômicos pequenos ou iguais a zero.
- 2 Aplicar um padrão com seu respectivo valor esperado de saída (t_j) e verificar a saída da rede (y_j).
- 3 Calcular o erro na saída: $E_j = t_j - y_j$
- 4 Se $E_j = 0$, voltar ao passo 2
Se $E_j \neq 0$, atualizar os pesos: $\Delta w_{ij} = \eta x_i E_j$
- 5 Voltar ao passo 2.

Regra Delta - LMS

- **Importante:**

- Não ocorre variação no peso se a saída estiver correta.
- Caso contrário, cada peso é incrementado de η quando a saída é maior que o valor-alvo.

$$\Delta w_{ij} = \eta x_i e_j \quad (1)$$

Sumário

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)

Gradiente de uma função

- **Gradiente:**

$$\nabla f(x, y) = \left(\frac{\partial}{\partial x} f(x, y), \frac{\partial}{\partial y} f(x, y) \right)$$

- **Derivada direcional:**

$$\begin{aligned} D_{\mathbf{u}} f(x, y) &= \nabla f(x, y) \cdot \mathbf{u} \\ &= \|\nabla f(x, y)\| \cdot \|\mathbf{u}\| \cos \gamma \\ &= \|\nabla f(x, y)\| \cdot \|\cos \gamma\| \end{aligned}$$

Gradiente de uma função

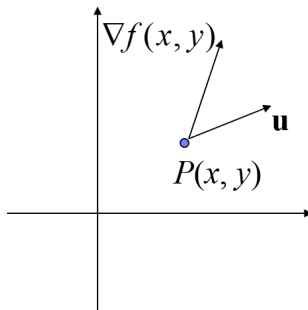


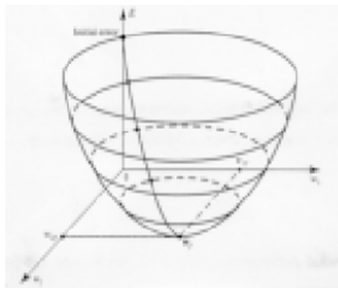
Figura 5: $D_{\mathbf{u}}f(x, y)$ é a taxa de variação de $f(x, y)$ na direção definida por \mathbf{u} .

Gradiente de uma função

- **Teorema do gradiente:** seja f uma função de duas variáveis, diferenciáveis no ponto $P(x, y)$.
 - O máximo de $D_{\mathbf{u}}f(x, y)$ em $P(x, y)$ é $\|\nabla f(x, y)\|$.
 - O máximo da taxa de crescimento de $f(x, y)$ em $P(x, y)$ ocorre na direção de $\nabla f(x, y)$.
- **Corolário:** seja f uma função de duas variáveis, diferenciáveis no ponto $P(x, y)$.
 - O mínimo de $D_{\mathbf{u}}f(x, y)$ em $P(x, y)$ é $-\|\nabla f(x, y)\|$.
 - O máximo da taxa de *decrescimento* de $f(x, y)$ em $P(x, y)$ ocorre na direção de $-\nabla f(x, y)$.

Gradiente de uma função

Superfície de Erro



Processo de Minimização

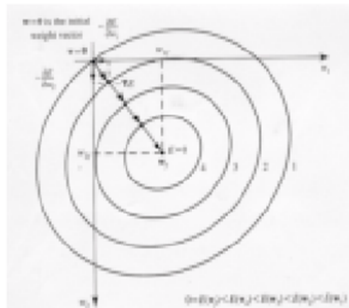


Figura 6: A direção do gradiente negativo é a de descida mais íngreme (*steepest descent*).

Método do gradiente descendente (GD)

$$\Delta w_{ij} = -\eta \frac{\delta E_j}{\delta w_{ij}}$$

- Cada *peso sináptico* i do elemento processador j é atualizado proporcionalmente ao *negativo da derivada parcial do erro* deste processador com relação ao peso.

Método do gradiente descendente (GD)

$$\Delta w_{ij} = -\eta \frac{\delta E_j}{\delta w_{ij}} = -\eta \frac{\delta E_j}{\delta y_j} \frac{\delta y_j}{\delta w_{ij}}$$

$$E_j = \frac{1}{2}(t_j - y_j)^2$$

$$y_j = \sum x_i w_{ij} + \theta_j$$

$$\Delta w_{ij} = -\eta \cdot \left[2 \cdot \frac{1}{2} (t_j - y_j) \cdot (-1) \right] \cdot x_i$$

$$= -\eta \cdot [-(t_j - y_j)] \cdot x_i = \eta (t_j - y_j) x_i$$

Sumário

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)

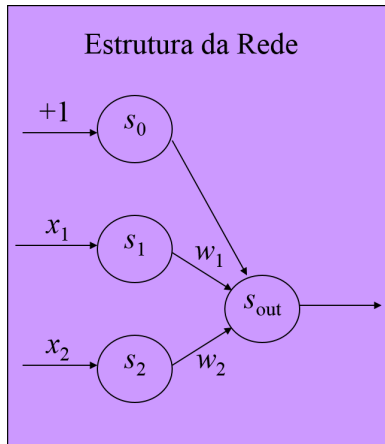
Exemplo

Exemplo: simulação do operador lógico AND

AND	x_0	x_1	x_2	t
Entrada 1:	1	0	0	0
Entrada 2:	1	0	1	0
Entrada 3:	1	1	0	0
Entrada 4:	1	1	1	1

Peso inicial: $w_0 = 0$, $w_1 = 0$, $w_2 = 0$

Taxa de aprendizado: $\eta = 0.5$



Exemplo: simulação do operador lógico AND

• 1º ciclo

- Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.0 \times 1 + 0.0 \times 0 + 0.0 \times 0) = f(0) = 0 \rightarrow s_{out} = t$
- Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.0 \times 1 + 0.0 \times 0 + 0.0 \times 1) = f(0) = 0 \rightarrow s_{out} = t$
- Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.0 \times 1 + 0.0 \times 1 + 0.0 \times 0) = f(0) = 0 \rightarrow s_{out} = t$
- Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.0 \times 1 + 0.0 \times 1 + 0.0 \times 1) = f(0) = 0 \rightarrow s_{out} \neq t$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = 0.0 + 0.5 \times (1 - 0) \times 1 = 0.5$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 0.0 + 0.5 \times (1 - 0) \times 1 = 0.5$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.0 + 0.5 \times (1 - 0) \times 1 = 0.5$$

Exemplo: simulação do operador lógico AND

• 2º ciclo

- Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.5 \times 1 + 0.5 \times 0 + 0.5 \times 0) = f(0.5) = 1 \rightarrow s_{out} \neq t$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = 0.5 + 0.5 \times (0 - 1) \times 1 = 0.0$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$$

- Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.0 \times 1 + 0.5 \times 0 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} \neq t$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = 0 + 0.5 \times (0 - 1) \times 1 = -0.5$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.5 + 0.5 \times (0 - 1) \times 1 = 0.0$$

Exemplo: simulação do operador lógico AND

• 2º ciclo

- Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-0.5 \times 1 + 0.5 \times 1 + 0.0 \times 0) = f(0) = 0 \rightarrow s_{out} = t$
- Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-0.5 \times 1 + 0.5 \times 1 + 0.0 \times 1) = f(0) = 0 \rightarrow s_{out} \neq t$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = -0.5 + 0.5 \times (1 - 0) \times 1 = 0.0$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 0.5 + 0.5 \times (1 - 0) \times 1 = 1.0$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.0 + 0.5 \times (1 - 0) \times 1 = 0.5$$

Exemplo: simulação do operador lógico AND

• 3º ciclo

- Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.0 \times 1 + 1.0 \times 0 + 0.5 \times 0) = f(0) = 0 \rightarrow s_{out} = t$
- Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(0.0 \times 1 + 1.0 \times 0 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} \neq t$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = 0.0 + 0.5 \times (0 - 1) \times 1 = -0.5$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 1.0 + 0.5 \times (0 - 1) \times 0 = 1.0$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.5 + 0.5 \times (0 - 1) \times 1 = 0.0$$

Exemplo: simulação do operador lógico AND

• 3º ciclo

- Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-0.5 \times 1 + 1 \times 1 + 0.0 \times 0) = f(0.5) = 1 \rightarrow s_{out} \neq t$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = -0.5 + 0.5 \times (0 - 1) \times 1 = -1.0$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 1.0 + 0.5 \times (0 - 1) \times 1 = 0.5$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.0 + 0.5 \times (0 - 1) \times 0 = 0.0$$

- Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-1.0 \times 1 + 0.5 \times 1 + 0.0 \times 1) = f(-0.5) = 0 \rightarrow s_{out} \neq t$$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = -1.0 + 0.5 \times (1 - 0) \times 1 = -0.5$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 0.5 + 0.5 \times (1 - 0) \times 1 = 1.0$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.0 + 0.5 \times (1 - 0) \times 1 = 0.5$$

Exemplo: simulação do operador lógico AND

- 4º ciclo

- Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-0.5 \times 1 + 1.0 \times 0 + 0.5 \times 0) = f(-0.5) = 0 \rightarrow s_{out} = t$$

- Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-0.5 \times 1 + 1.0 \times 0 + 0.5 \times 1) = f(0) = 0 \rightarrow s_{out} = t$$

Exemplo: simulação do operador lógico AND

• 4º ciclo

- Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-0.5 \times 1 + 1.0 \times 1 + 0.5 \times 0) = f(0.5) = 1 \rightarrow s_{out} \neq t$$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = -0.5 + 0.5 \times (0 - 1) \times 1 = -1.0$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 1.0 + 0.5 \times (0 - 1) \times 1 = 0.5$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.5 + 0.5 \times (0 - 1) \times 0 = 0.5$$

- Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-1.0 \times 1 + 0.5 \times 1 + 0.5 \times 1) = f(0) = 0 \rightarrow s_{out} \neq t$$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = -1.0 + 0.5 \times (1 - 0) \times 1 = -0.5$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 0.5 + 0.5 \times (1 - 0) \times 1 = 1.0$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 0.5 + 0.5 \times (1 - 0) \times 1 = 1.0$$

Exemplo: simulação do operador lógico AND

● 5º ciclo

- Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-0.5 \times 1 + 1.0 \times 0 + 1.0 \times 0) = f(-0.5) = 0 \rightarrow s_{out} = t$$

- Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-0.5 \times 1 + 1.0 \times 0 + 1.0 \times 1) = f(0.5) = 1 \rightarrow s_{out} \neq t$$

$$w_0 = w_0 + \eta(t - s_{out})x_0 = -0.5 + 0.5 \times (0 - 1) \times 1 = -1.0$$

$$w_1 = w_1 + \eta(t - s_{out})x_1 = 1.0 + 0.5 \times (0 - 1) \times 0 = 1.0$$

$$w_2 = w_2 + \eta(t - s_{out})x_2 = 1.0 + 0.5 \times (0 - 1) \times 1 = 0.5$$

Exemplo: simulação do operador lógico AND

- 5º ciclo

- Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1.0 \times 1 + 1.0 \times 1 + 0.5 \times 0) = f(0) = 0 \rightarrow s_{out} = t$

- Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$
 $= f(-1.0 \times 1 + 1.0 \times 1 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} = t$

Exemplo: simulação do operador lógico AND

● 6º ciclo

- Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-1.0 \times 1 + 1.0 \times 0 + 0.5 \times 0) = f(-1) = 0 \rightarrow s_{out} = t$$

- Entrada 2: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-1.0 \times 1 + 1.0 \times 0 + 0.5 \times 1) = f(-0.5) = 0 \rightarrow s_{out} = t$$

- Entrada 3: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

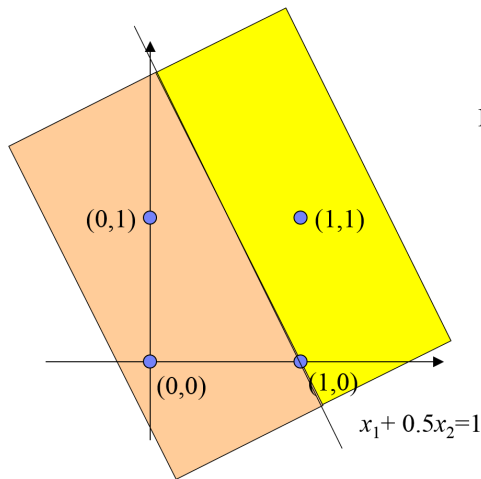
$$= f(-1.0 \times 1 + 1.0 \times 1 + 0.5 \times 0) = f(0) = 0 \rightarrow s_{out} = t$$

- Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$$= f(-1.0 \times 1 + 1.0 \times 1 + 0.5 \times 1) = f(0.5) = 1 \rightarrow s_{out} = t$$

- $w_0 = -1.0, w_1 = 1.0, w_2 = 0.5$

Interpretação geométrica



Linha de Decisão:

$$x_1 w_1 + x_2 w_2 = -\theta$$



$$x_1 + 0.5 x_2 = 1$$

Sumário

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)

Teorema de convergência do Perceptron

- Seja o vetor de entrada:
 - $\mathbf{x}(n) = [-1, x_1(n), x_2(n), \dots, x_p(n)]^T$
- E seu correspondente vetor peso:
 - $\mathbf{w}(n) = [\theta(n), w_1(n), w_2(n), \dots, w_p(n)]^T$
- A saída pode ser descrita na sua forma compacta:
 - $\mathbf{v}(n) = \mathbf{w}_T(n) \cdot \mathbf{x}(n)$
- A equação $\mathbf{w}^T \mathbf{x}$, plotada em um plano p-dimensional, define um hiperplano entre duas classes diferentes.

Teorema de convergência do Perceptron

- Seja $X_1 = \{\mathbf{x}_1(1), \mathbf{x}_1(2), \dots\}$ o conjunto de vetores de treinamento pertencentes à classe \mathcal{C}_1
- Seja $X_2 = \{\mathbf{x}_2(1), \mathbf{x}_2(2), \dots\}$ o conjunto de vetores de treinamento pertencentes à classe \mathcal{C}_2
- $X = X_1 \cup X_2$

Teorema de convergência do Perceptron

- Usando X_1 e X_2 para treinar o classificador \rightarrow ajuste do vetor peso \mathbf{w} , de tal forma que as classes \mathcal{C}_1 e \mathcal{C}_2 fiquem separadas.
- Então, duas classes são ditas linearmente separáveis se existe um vetor peso \mathbf{w} .
- Analogamente, se as duas classes são linearmente separáveis, então \exists um vetor \mathbf{w} tal que:

$$\begin{cases} \mathbf{w}^T \mathbf{x} \geq 0, \forall \mathbf{x} \in \mathcal{C}_1 \\ \mathbf{w}^T \mathbf{x} < 0, \forall \mathbf{x} \in \mathcal{C}_2 \end{cases} \quad (2)$$

Teorema de convergência do Perceptron

- Portanto, dados os subconjuntos X_1 e X_2 , o problema de treinamento do Perceptron consiste em encontrar um vetor \mathbf{w} que satisfaça às duas desigualdades apresentadas em (2).
- O algoritmo para atualização do vetor \mathbf{w} pode ser formulado conforme segue:

Algoritmo

- Se $\mathbf{x}(n)$ é classificado corretamente, então:

$$\begin{cases} \mathbf{w}(n+1) = \mathbf{w}(n), & \text{se } \mathbf{w}^T \mathbf{x} \geq 0, x \in \mathcal{C}_1 \\ \mathbf{w}(n+1) = \mathbf{w}(n), & \text{se } \mathbf{w}^T \mathbf{x} < 0, x \in \mathcal{C}_2 \end{cases}$$

- Caso contrário, o vetor peso é atualizado:

$$\begin{cases} \mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n), & \text{se } \mathbf{w}^T \mathbf{x} \geq 0, x \in \mathcal{C}_2 \\ \mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n), & \text{se } \mathbf{w}^T \mathbf{x} < 0, x \in \mathcal{C}_1 \end{cases}$$

- Onde $\eta(n)$ é o parâmetro *velocidade de aprendizado*.
 - Se $\eta(n) = \eta$, então o parâmetro velocidade é fixo.

Convergência

- A convergência será provada com $\eta = 1$ e $\mathbf{w}(0) = \mathbf{0}$.
- Suponha que $\mathbf{w}^T(n)\mathbf{x}(n) < 0$ para $n = 1, 2, \dots$ e que o vetor de entrada $\mathbf{x}_k(n) \in X_1$, isto é, a segunda condição de (2) é verdadeira.
- Então, a correção deve ser realizada de acordo com:
 - $\mathbf{w}(n+1) = \mathbf{w}(n) + \mathbf{x}(n)$
- Usando a condição inicial $\mathbf{w}(0) = \mathbf{0}$, pode-se resolver iterativamente esta equação para:

$$\mathbf{w}(n+1) = \mathbf{x}(1) + \mathbf{x}(2) + \dots + \mathbf{x}(n) \quad (3)$$

Convergência

- Contanto que as classes \mathcal{C}_1 e \mathcal{C}_2 sejam assumidas linearmente separáveis, existe uma solução \mathbf{w}_0 tal que $\mathbf{w}_0^T \mathbf{x}(n) > 0, \quad n = 1, 2, \dots$
- Seja $d = \min\{\mathbf{w}_0^T \mathbf{x}(n)\}, \mathbf{x}(n) \in X_1$
- Multiplicando ambos os termos de (3) por \mathbf{w}_0^T , obtemos:

$$\mathbf{w}_0^T \mathbf{w}(n+1) = \mathbf{w}_0^T \mathbf{x}(1) + \mathbf{w}_0^T \mathbf{x}(2) + \dots + \mathbf{w}_0^T \mathbf{x}(n) \geq n \cdot d$$

- Usando a desigualdade de Cauchy, tem-se:

$$n^2 d^2 \leq [\mathbf{w}_0^T \mathbf{w}(n+1)]^2 \leq \|\mathbf{w}_0\|^2 \|\mathbf{w}(n+1)\|^2 \implies$$

$$\|\mathbf{w}(n+1)\|^2 \geq \frac{n^2 d^2}{\|\mathbf{w}_0\|^2} \quad (4)$$

Convergência

- Considerando a equação (2):

$$\|\mathbf{w}(k+1)\|^2 = \|\mathbf{w}(k)\|^2 + \|\mathbf{x}(k)\|^2 + 2\mathbf{w}^T(k) \cdot \mathbf{x}(k) \leq 0$$

$$\|\mathbf{w}(k+1)\|^2 \leq \|\mathbf{w}(k)\|^2 + \|\mathbf{x}(k)\|^2$$

- Ou, equivalentemente:

$$\|\mathbf{w}(k+1)\|^2 - \|\mathbf{w}(k)\|^2 \leq \|\mathbf{x}(k)\|^2, \quad k = 1, 2, \dots, n$$

- Somando estas desigualdades para $k = 1, 2, \dots, n$ e $\mathbf{w}(0) = 0$:

$$\|\mathbf{w}(n+1)\|^2 \leq \sum_k \|\mathbf{x}(k)\|^2 \leq n \cdot \beta \quad (5)$$

- Onde $\beta = \max\{\|\mathbf{x}(k)\|^2\}$, $\mathbf{x}(k) \in X_1$

Convergência

- A equação (5) diz que a norma euclidiana ao quadrado do vetor peso $\mathbf{w}(n+1)$ cresce linearmente com o número de iterações.
- Isso entra em conflito com a equação (4).
- n não pode ser maior que n_{max} , para o qual as equações (4) e (5) valem com sinal de igualdade:

$$\frac{n_{max}^2 d^2}{\|\mathbf{w}_0\|^2} = n_{max} \beta \implies n_{max} = \frac{\beta \|\mathbf{w}_0\|^2}{d^2}$$

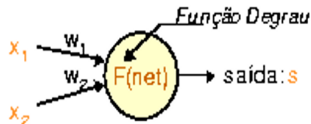
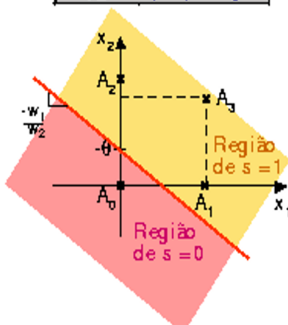
- Portanto, supondo que \mathbf{w}_0 existe, a regra de atualização deve terminar em n_{max} iterações.

Sumário

- 1 Introdução
- 2 Algoritmo de aprendizado
 - Modelos de neurônios
 - Características básicas
 - Regra Delta - LMS
 - Gradiente de uma função
 - Exemplo
- 3 Teorema de convergência
- 4 O problema do OU exclusivo (XOR)

O problema do OU exclusivo (XOR)

PONTO	x_1	x_2	Saída
A_0	0	0	0
A_1	0	1	1
A_2	1	0	1
A_3	1	1	0



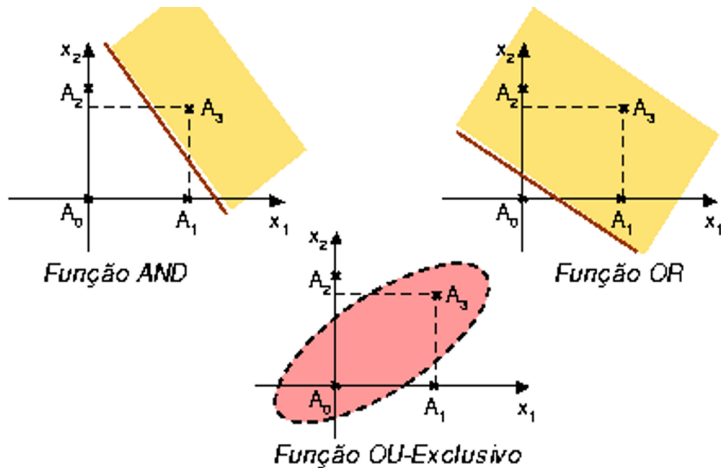
De acordo com a definição do neurônio: $s = F(x_1 w_1 + x_2 w_2 + \theta)$

$$\text{net} = x_1 w_1 + x_2 w_2 + \theta \rightarrow \begin{cases} \text{Se net} \geq 0 \rightarrow s = 1 \\ \text{Se net} < 0 \rightarrow s = 0 \end{cases}$$

A rede Perceptron divide o plano $x_1 \times x_2$ em duas regiões (através da reta net)



O problema do OU exclusivo (XOR)



O problema do OU exclusivo (XOR)

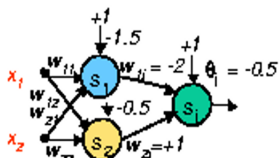
- Mudando-se os valores de w_1 , w_2 e θ , muda-se a inclinação da reta.
- Entretanto, é impossível achar uma reta que divida o plano de forma a separar os pontos A_1 e A_2 de um lado e A_0 e A_3 de outro.
- Redes de uma única camada só representam **funções linearmente separáveis**.

O problema do OU exclusivo (XOR)

- Minsky & Papert provaram que esse problema pode ser solucionado adicionando-se uma outra camada intermediária de processadores → Multi-Layer Perceptron (MLP).

O problema do OU exclusivo (XOR)

Exemplo:



$$w_{11} = w_{12} = w_{21} = w_{22} = +1$$

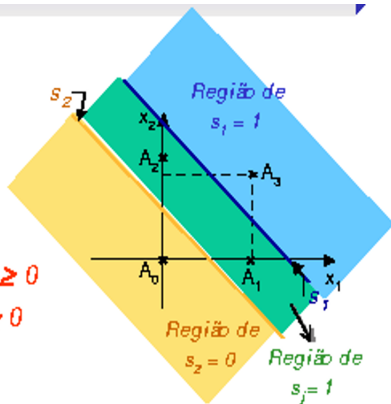
$$s_j = 1 \rightarrow s_1 w_{1j} + s_2 w_{2j} + \theta_j \geq 0$$

$$-2s_1 + s_2 - 0.5 \geq 0$$

$$-2s_1 + s_2 \geq 0.5$$



s_1 é inibitório
 s_2 é excitatório



O problema do OU exclusivo (XOR)

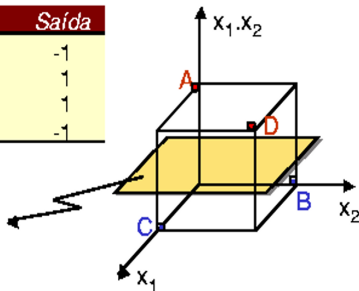
- Exemplo do OU-EXCLUSIVO:

- $J = 2$ (número de entradas originais - x_1, x_2)

- $H = 1$ (número de entradas adicionais - $x_1 \cdot x_2$)

Pontos	Entradas				Saída
A	-1	-1	1	⇒	-1
B	-1	1	-1	⇒	1
C	1	-1	-1	⇒	1
D	1	1	1	⇒	-1

Problema
Linearmente
Separável



Multi-Layer Perceptron

- Redes de apenas uma camada só representam funções linearmente separáveis.
- Redes de múltiplas camadas solucionam essa restrição.
- O desenvolvimento do algoritmo *backpropagation* foi um dos motivos para o ressurgimento da área de redes neurais.