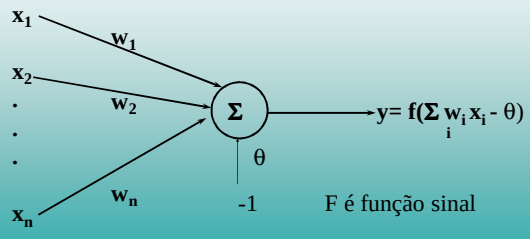
LABIC

Modelos de Neurônios

- **Características Básicas**
- **Modelo de Neurônio**
- **Estrutura da Rede**
- **Algoritmo de Aprendizado**

LABIC

I - PERCEPTRON



LABIC

Perceptron

- O algoritmo usado para ajustar os param. Livres desta rede apareceu num proc. De aprendizado desenvolvido por Rosemlbatt (1958, 1962).
- Ele provou que se os padrões usados para treinar são Linearmente separáveis, então o alg. Converge e a superf. de decisão tem a forma de um hiperplano entre duas classes.

LABIC

Perceptron

- É constituído de apenas 01 neuônio e como tal só consegue classificar padrões envolvendo apenas 2 classes, que devem ser linearmente separáveis.
- A regra de decisão é designar x a classe C1 se a saída y = +1 e a classe C2 se a saída é -1.

LABIC

Perceptron

- Existem duas regiões separadas pelo hiperplano: $\sum wixi - \ominus = 0$
- Se o espaço for o R^2, a região. de separação é uma reta

LABIC

Teorema de Convergência

- Seja o vetor de entrada:

$X(n) = [-1, x_1(n), x_2(n), \ldots, x_p(n)]^T$

E seu correspondente vetor peso:

$W(n) = [\ominus(n), w_1(n), w_2(n), \ldots, w_p(n)]^T$

A saída pode ser descrita na sua forma compacta:

$v(n) = w^T(n).\ X(n)$

A equação $w^T x = 0$, plotada num espaço p-dimensional define um hiperplano entre duas classes diferentes.

LABIC

Teorema de Convergência

- Seja X1=$\{x^1(1), x^1(2), \ldots.\}$ ser oconjunto de vetores de treinamento que pertecem a classe C1
- Seja X2=$\{x^2(1), x^2(2), \ldots.\}$ ser oconjunto de vetores de treinamento que pertecem a classe C2
- X = X1 U X2

LABIC

Teorema de Convergência

- X1 e X2 para treinar o classificador → ajuste do vetor peso W, de tal forma que as classes C1 e C2 fiquem separadas.
- Então duas classes são ditas linearmente separáveis se existe um vetor peso W.
- Analogamente, se as 2 classes sao Linearmente separáveis, então ∃ um vetor w tal que: $w^T x \geq 0\ , \forall\ x, x \in C1$ (X)

$w^T x < 0, \forall\ x, x \in C2$

Teorema de Convergência

- Portanto, dados os subconj. X1 e X2, o probl. De treinamento do Perceptron consiste em encontrar um vetor W que satisfaz as duas desigualdades (X).
- O alg. para atualização do vetor W pode ser formulado:
- Se x(n) é classif. Corretamente então:

$w(n+1) = w(n)$ se $w^T x \geq 0$, $x \in C1$

$w(n+1) = w(n)$ se $w^T x < 0$, $x \in C2$

Algoritmo

- Caso contrário, o vetor peso é atualizado:

$W(n+1) = w(n) - \eta(n)\, x(n)$, se $w^T x \geq 0$, $x \in C2$

$W(n+1) = w(n) + \eta(n)\, x(n)$, se $w^T x < 0$, $x \in C1$

onde $\eta(n)$ é o param. Veloc. aprendizado

Se $\eta(n) = \eta$ – param veloc. é fixo

Convergencia

- A convergencia será provada com $\eta=1$

$W(0) = 0$; suponha que $w^T(n)x(n)<0$ para $n=1,2,\ldots$ e que um vetor de entrada $x(n) \in$ ao conjunto X1, isto é,

A segunda condição de (X) é verdadeira.

Então, a correção deve ser realizada de acordo com:

$w(n+1) = w(n) + x(n)$

Convergência

- Usando a condição inicial $w(1) = 0$, nós podemos iterativamente resolver esta eq. p/ $w(n+1) = x(1) + x(2) + \ldots + x(n)$ (1)
- Desde que as classes C1 e C2 são assumidas L.S., $\exists$ 1 solução w_0 / $w_0^T x(n)>0$, $n=1,2,\ldots$

$\in$ X1.

Seja $d = \min_{x(n) \in X1} w_0^T x(n)$

LABIC

Convergência

- Multiplicando ambos os membros de (1) por w_0^T obtemos:

$$w_0^T w(n+1) = w_0^T x(1) + w_0^T x(2) + \ldots + w_0^T x(n) \geq n\,d$$

Usando a desigualdade de Cauchy, tem-se:

$$n^2 d^2 \leq [w_0^T w(n+1)]^2 \leq \|w_0\|^2 \|w(n+1)\|^2$$

Ou $\|w(n+1)\|^2 \geq n^2 d^2 / \|w_0\|^2$ (2)

LABIC

Convergência

- Considerando a equação (X):

$\|w(k+1)\|^2 = \|w(k)\|^2 + \|x(k)\|^2 + 2w^T(k).x(k)$
≤ 0

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

Ou, equivalente:

$\|w(k+1)\|^2 - \|w(k)\|^2 \leq \|x(k)\|^2$, k= 1, 2, ..., n

Adicionando estas desigualdades para k= 1, 2, ..., n e w(0)=0,

$\|w(n+1)\|^2 \leq \sum_k \|x(k)\|^2 \leq n\,\beta$ (3)

Onde $\beta = \max \|x(k)\|^2$ x(k) pertencente ao X1.

LABIC

Convergência

- A eq. (3) coloca que a norma eucl. ao quadr. do vetor peso w(n+1) cresce linearmente com o no. de iterações.

Isto entra em conflito com a eq. (2).

N não pode ser maior que n_{max} para os quais as eq. (2) e (3) valem com o sinal de igualdade:

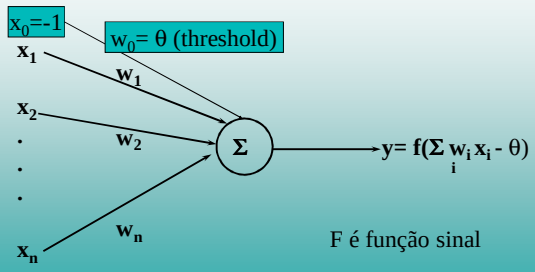
$n_{max}^2\, d^2 / \|w_0\|^2 = n_{max}\, \beta$

- $n_{max} = \beta \|w_0\|^2 / d^2$

Portanto, supondo que w_0 existe a regra de atualiz. deve terminar em n_{max} iterações

LABIC

Modelos de Neurônios

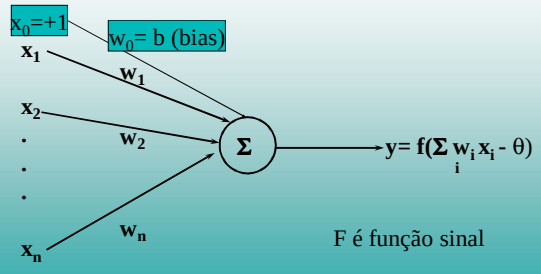


F é função sinal

4

LABIC

Modelos de Neurônios



F é função sinal

LABIC

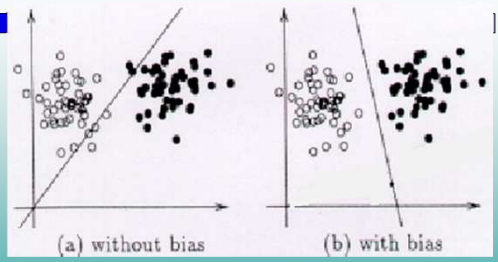
CARACTERISTICAS BASICAS

$$y_j = \text{sgn}(\sum_i x_i w_{ij})$$

- Regra de propagação
- 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)
- Valor de Entrada/Saída: Binários t= 1 ou -1

LABIC

Finalidade do Termo *Bias*:



$\sum_i x_i w_{ij} = 0$ Define um hiperplano passando pela origem

$\sum_i x_i w_{ij} + \theta_i = 0$ Desloca-se o hiperplano da origem

LABIC

ALGORITMO DE APRENDIZADO
Regra Delta - LMS

1) iniciar os pesos sinápticos com valores randomicos e pequenos ou iguais a zero;
2) aplicar um padrão com seu respectivo valor desejado de saída (t_j) e verificar a saída da rede (y_j);
3) calcula o erro na saída $E_j = t_j - y_j$;
4) se $E_j = 0$, volta ao passo 2;
   se $E_j \neq 0$, atualiza os pesos: $\Delta w_{ij} = \eta x_i E_j$;
5) volta ao passo 2.

5

LABIC

ALGORITMO DE APRENDIZADO

IMPORTANTE

- não ocorre variação no peso se a saída estiver correta;
- caso contrário, cada peso é incrementado de η quando a saída é menor que o valor-alvo e decrementado de η quando a saída é maior que o valor-alvo.

$$\Delta w_{ij} = \eta\, x_i\, e_j$$

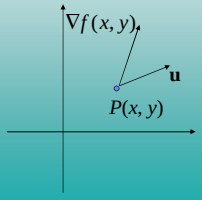
LABIC

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:

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



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

LABIC

Gradiente de uma função

Teorema do gradiente: Seja f uma função de duas variáveis, diferenciáveis no ponto $P(x, y)$.

i) O máximo de $D_{\mathbf{u}}f(x, y)$ em $P(x, y)$ é $\|\nabla f(x, y)\|$.

ii) 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 $P(x, y)$.

i) O mínimo de $D_{\mathbf{u}}f(x, y)$ em $P(x, y)$ é $-\|\nabla f(x, y)\|$.

ii) O máximo da taxa de decrescimento de $f(x, y)$ em $P(x, y)$ ocorre na direção de $-\nabla f(x, y)$.

LABIC

LABIC

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.

LABIC

Logo:

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

$$E_j = \frac{1}{2}(t_j - s_j)^2 \qquad s_j = k\Sigma x_i . w_{ij} + \theta_j$$

$$\Delta w_{ij} = -\eta . [2 . \frac{1}{2}(t_j - s_j).(-1)] . x_i$$

$$= -\eta . [-(t_j - s_j)] . x_i = \eta x_i (t_j - s_j)$$

LABIC

EXEMPLO

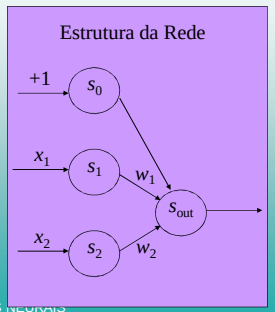
Simulação do Operador Lógico AND

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: $\eta = 0.5$

Estrutura da Rede



LABIC

EXEMPLO

1º Ciclo

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

$= f(0\times1+0\times0+0\times0) = f(0) = 0 \longrightarrow s_{out} = t$

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

$= f(0\times1+0\times1+0\times0) = f(0) = 0 \longrightarrow s_{out} = t$

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

$= f(0\times1+0\times0+0\times1) = f(0) = 0 \longrightarrow s_{out} = t$

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

$= f(0\times1+0\times1+0\times1) = f(0) = 0 \longrightarrow s_{out} \neq t$

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

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

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

7

EXEMPLO

LABIC **2º Ciclo**

Entrada1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$ ⟶

$= f(0.5\times1+0.5\times0+0.5\times0) = f(0.5) = 1$ $s_{out} \neq t$

$w_0 = w_0+(t-s_{out})x_0 = 0.5 + 0.5 \times(0-1) \times1 = 0$

$w_1 = w_1+(t-s_{out})x_1 = 0.5 + 0.5 \times(0-1) \times0 = 0.5$

$w_2 = w_2+(t-s_{out})x_2 = 0.5 + 0.5 \times(0-1) \times0 = 0.5$

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

$= f(0\times1+0.5\times0+0.5\times1) = f(0.5) = 1$ $s_{out} \neq t$

$w_0 = w_0+(t-s_{out})x_0 = 0 + 0.5 \times(0-1) \times1 = -0.5$

$w_1 = w_1+(t-s_{out})x_1 = 0.5 + 0.5 \times(0-1) \times0 = 0.5$

$w_2 = w_2+(t-s_{out})x_2 = 0.5 + 0.5 \times(0-1) \times1 = 0$

EXEMPLO

LABIC

2º Ciclo

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

$= f(-0.5\times1+0.5\times1+0\times0) = f(0) = 0$ ⟶ $s_{out} = t$

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

$= f(-0.5\times1+0.5\times1+0\times1) = f(0) = 0$ ⟶ $s_{out} \neq t$

$w_0 = w_0+(t-s_{out})x_0 = -0.5 + 0.5 \times(1-0) \times1 = 0$

$w_1 = w_1+(t-s_{out})x_1 = 0.5 + 0.5 \times(1-0) \times1 = 1$

$w_2 = w_2+(t-s_{out})x_2 = 0 + 0.5 \times(1-0) \times1 = 0.5$

EXEMPLO

LABIC

3º Ciclo

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

$= f(0\times1+1\times0+0.5\times0) = f(0) = 0$ ⟶ $s_{out} = t$

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

$= f(0\times1+1\times0+0.5\times1) = f(0.5) = 1$ $s_{out} \neq t$

$w_0 = w_0+(t-s_{out})x_0 = -0.5 + 0.5 \times(0-1) \times1 = -1$

$w_1 = w_1+(t-s_{out})x_1 = 1 + 0.5 \times(0-1) \times0 = 1$

$w_2 = w_2+(t-s_{out})x_2 = 0.5 + 0.5 \times(0-1) \times1 = 0$

EXEMPLO

LABIC

3º Ciclo

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

$= f(-1\times1+1\times1+0\times0) = f(0) = 0$ $s_{out} = t$

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

$= f(-1\times1+1\times1+0\times1) = f(0) = 0$ $s_{out} \neq t$

$w_0 = w_0+(t-s_{out})x_0 = -1 + 0.5 \times(1-0) \times1 = -0.5$

$w_1 = w_1+(t-s_{out})x_1 = 1 + 0.5 \times(1-0) \times1 = 1.5$

$w_2 = w_2+(t-s_{out})x_2 = 0 + 0.5 \times(1-0) \times1 = 0.5$

EXEMPLO

LABIC **4º Ciclo** Entrada 1: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$= f(-0.5\times1+1.5\times0+0.5\times0) = f(-0.5) = 0 \longrightarrow s_{out} = t$

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

$= f(-0.5\times1+1.5\times0+0.5\times1) = f(0) = 0 \longrightarrow s_{out} = t$

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

$= f(-0.5\times1+1.5\times1+0.5\times0) = f(1) = 1 \longrightarrow s_{out} \neq t$

$w_0 = w_0+(t-s_{out})x_0 = -0.5 + 0.5\times(0-1)\times1 = -1$

$w_1 = w_1+(t-s_{out})x_1 = 1.5 + 0.5\times(0-1)\times1 = 1$

$w_2 = w_2+(t-s_{out})x_2 = 0.5 + 0.5\times(0-1)\times0 = 0.5$

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

$= f(-1\times1+1\times1+0.5\times1) = f(0.5) = 1 \longrightarrow s_{out} = t$

EXEMPLO

LABIC

5º Ciclo $\longrightarrow$

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

$= f(-1\times1+1\times0+0.5\times0) = f(-1) = 0 \longrightarrow s_{out} = t$

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

$= f(-1\times1+1\times0+0.5\times1) = f(-0.5) = 0 \longrightarrow s_{out} = t$

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

$= f(-1\times1+1\times1+0.5\times0) = f(0) = 0 \longrightarrow s_{out} = t$

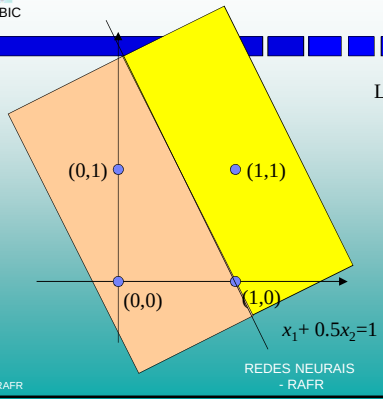
Entrada 4: $s_{out} = f(w_0x_0 + w_1x_1 + w_2x_2)$

$= f(-1\times1+1\times1+0.5\times1) = f(0.5) = 1 \quad s_{out} = t$

$w_0 = -1,\ w_1 = 1,\ w_2 = 0.5$

INTERPRETAÇÃO GEOMÉTRICA

LABIC



Linha de Decisão:

$x_1w_1+x_2w_2 = -\theta$

$x_1 + 0.5\,x_2 = 1$

Perceptron

LABIC

- **OBS:** A classe de funções representadas por Perceptrons é Limitada

2) Perceptrons Não-Lineares

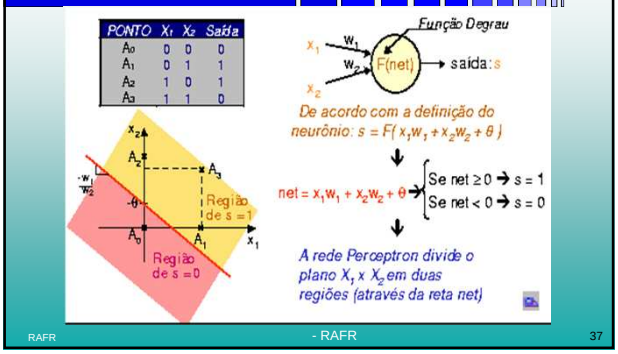
$\text{Out}(\underline{x}) = g(\underline{w}^t\,\underline{x}) \dashrightarrow y$

$\text{Out}(\underline{x}) = g(\underline{w}^T\,\underline{x}) = g(\sum_i w_i\,x_i)$ onde

g é uma função não-linear (SIGMOID)

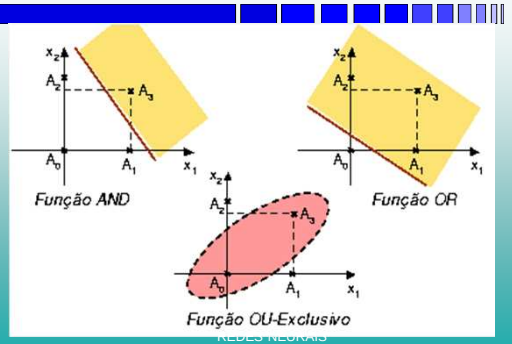
LABIC

O PROBLEMA DO OU-EXCLUSIVO (XOR)



LABIC

O PROBLEMA DO OU-EXCLUSIVO (XOR)



LABIC

O PROBLEMA DO OU-EXCLUSIVO (XOR)

Conclusão

- mudando-se os valores de w_1, w_2 e θ, muda-se a inclinação e a posição da reta;
- entretanto é impossível achar uma reta que divide o plano de forma separar os pontos A_1 e A_2 de um lado e A_0 e A_3 de outro
- redes de 1 única camada só representam **funções linearmente separáveis**

LABIC

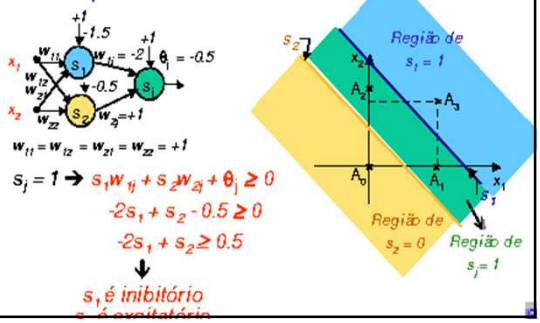
O PROBLEMA DO OU-EXCLUSIVO (XOR)

Minsky & Papert provaram que este problema pode ser solucionado adicionando-se uma outra camada intermediaria de processadores- Multi-Layer Perceptron (MLP)

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

Exemplo:



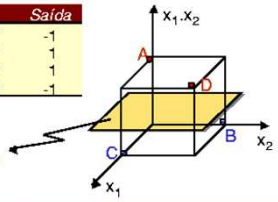
RAFR

O PROBLEMA DO OU-EXCLUSIVO (XOR)

LABIC

- Exemplo do OU-EXCLUSIVO:
  - J = 2 (número de entradas originais - x_1,x_2)
  - H = 1 (número de entradas adicionais - x_1.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

RAFR

REDES NEURAIS - RAFR

42

MULTI-LAYER PERCEPTRON

LABIC

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

RAFR

REDES NEURAIS - RAFR

43

Algoritmos de Aprendizado

LABIC

- **Problema**

$\underline{\mathbf{x}}_1$ y_1
$\underline{\mathbf{x}}_2$ y_2
$\underline{\mathbf{x}}_3$ y_3

$\underline{\mathbf{x}}_N$ y_N

$\underline{\mathbf{x}}$ denota um vetor de p componentes

RAFR

REDES NEURAIS - RAFR

44

LABIC

- Perceptrons Lineares: eles são modelos multivariados lineares:

$$Out(\underline{x}) = \underline{w}^T \underline{x} \rightarrow y$$

- e o treinamento consiste de minimizar a soma dos quadrados do resíduo pelo método do "gradiente descent".
- Regra Gradiente Descent:

$$w \leftarrow w - \eta \, \partial f(w) / \partial w$$

LABIC

Adaline

- O objetivo do processo adaptativo do Adaline consiste em utilizar a função de ativação "hard limiter (saída +1 ou –1)" e minimizar os pesos e θ usando o algoritmo LMS.
- Como a saída desejada pode ter apenas valores 1 ou –1, o erro pode ser +2, 0, ou –2.

LABIC

Exercícios

- Pensar em uma rede de perceptrons constituída de uma camada intermediaria constituída de um único neurônio, para implementar a função OU-EXCLUSIVO.
- Implementar uma ADALINE para reconhecer a letra A e a letra A invertida.