UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

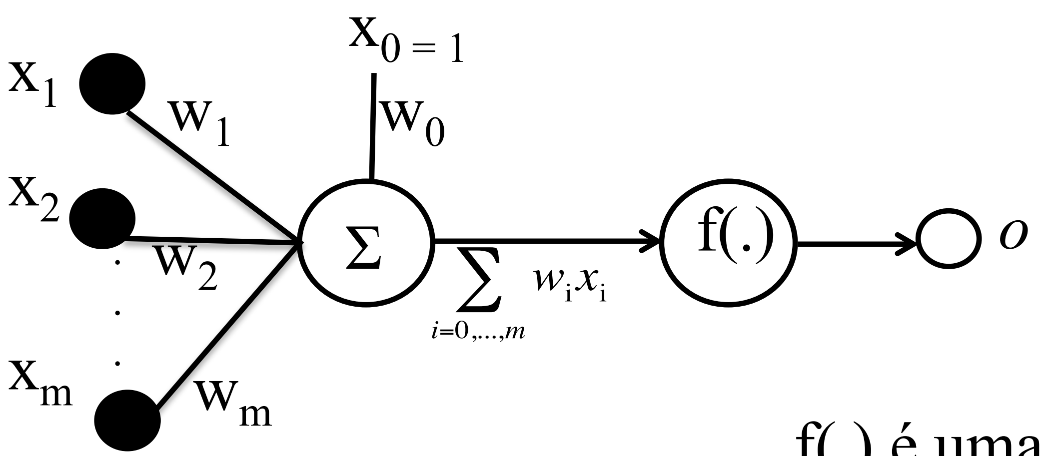
Redes Perceptron de Múltiplas Camadas

Gisele L. Pappa

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Perceptron de uma Camada

- Primeiro modelo para aprendizado supervisionado
- Padrões linearmente separáveis



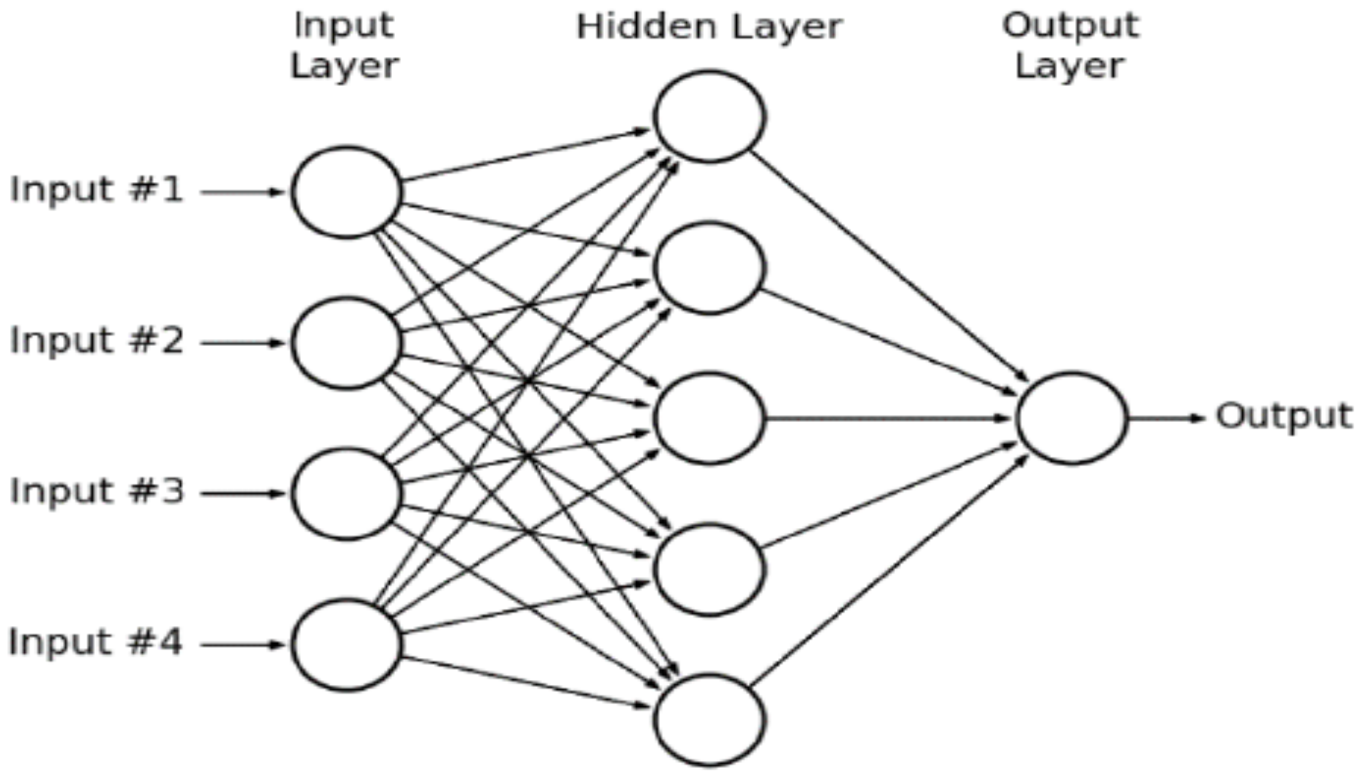
f(.) é uma função linear

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

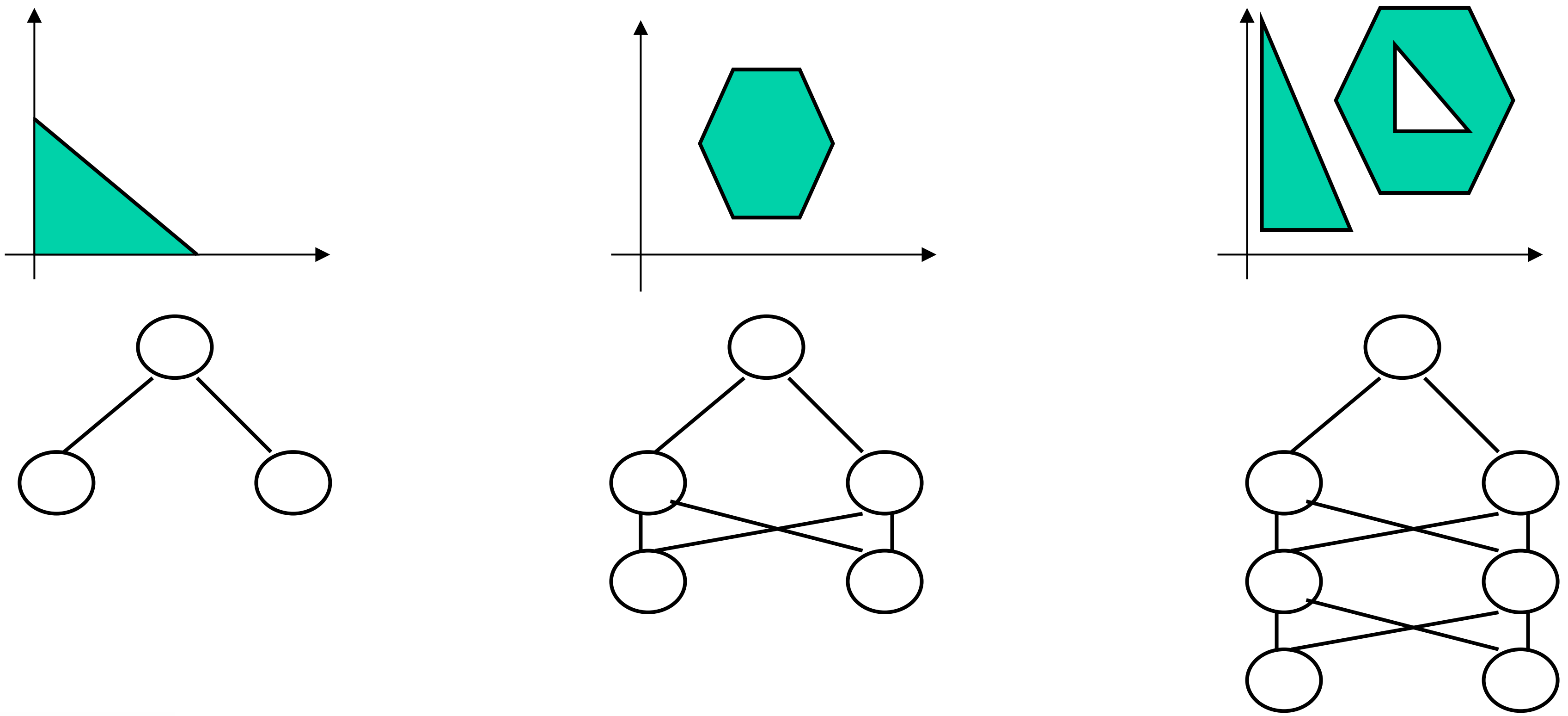
Multi-layer Perceptron (MLP)



DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

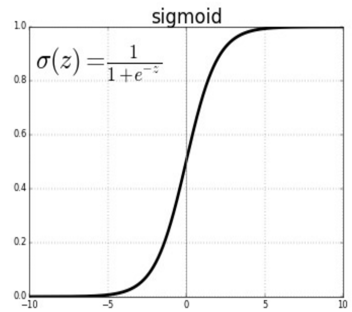
O papel de cada camada escondida

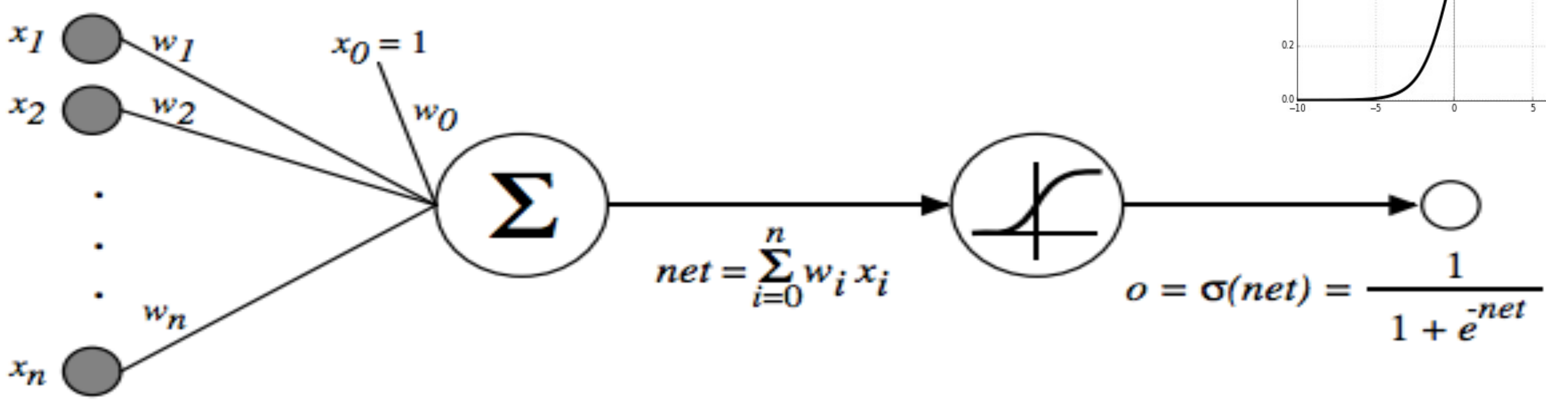


DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

Perceptron com sigmoide





$\sigma(x)$ representa a função sigmoide ou logística

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Arquitetura

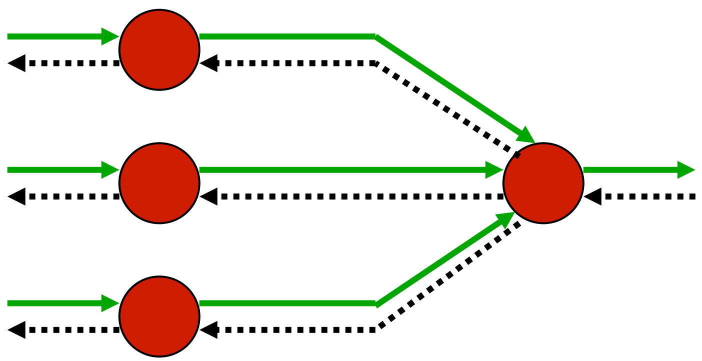
- Não existem:
  - Conexões entre neurônios da mesma camada
  - Conexões diretas entre as camadas de entrada e saída
- Número de neurônios:
  - Camadas de entrada e saída: dependem da modelagem do problema
  - Camadas escondidas: parâmetro, pode ser maior ou menor que o número dos neurônios das camadas de entrada e saída

Arquitetura

- Uma rede com uma camada escondida e uma função sigmoide, é capaz de aproximar qualquer função contínua $f : [-1,+1]^k \rightarrow [0,1]$
- Porém, o número de neurônios na camada escondida pode ser muito grande

Algoritmo de Aprendizagem

- *Back-propagation*



Os sinais das funções são enviados em uma direção

Os sinais de erro são enviados na direção contrária

- Ajusta o peso da rede visando minimizar o erro médio quadrático.

Back Propagation /Propagação pra frente

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

Inicialize os pesos aleatoriamente, e escolha uma taxa de aprendizado

Enquanto (critério de parada não satisfeito)

Para cada exemplo de treinamento

1. Passe o exemplo pela rede para produzir as saídas

Para cada neurônio da rede, da primeira para última camada

- Entradas são multiplicadas pelos pesos
- Somadas
- Passam pela sigmoide
- Saem e servem de entrada para os neurônios da próxima camada

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

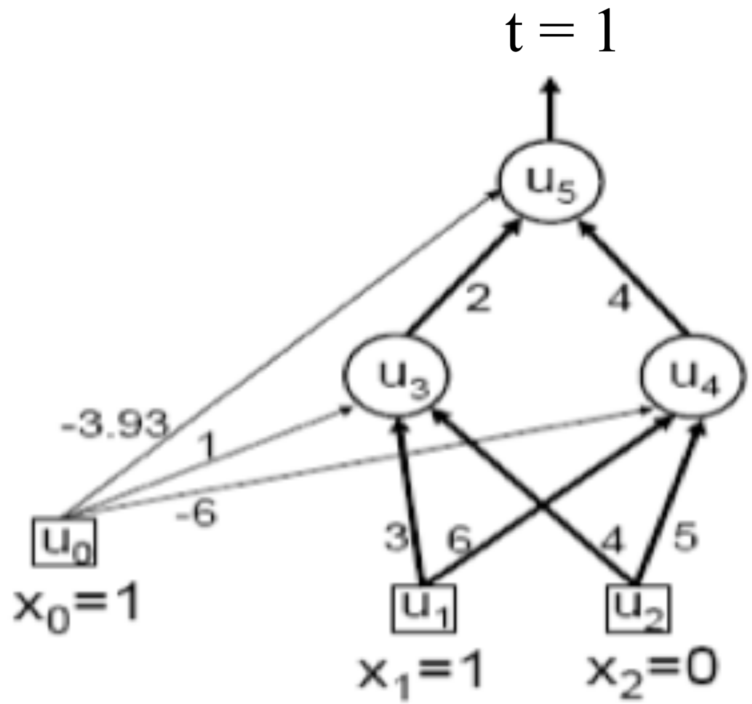
MLP/BP: Exemplo

- Estado atual

(u_0 corresponde ao bias)

- Exemplo de treinamento:

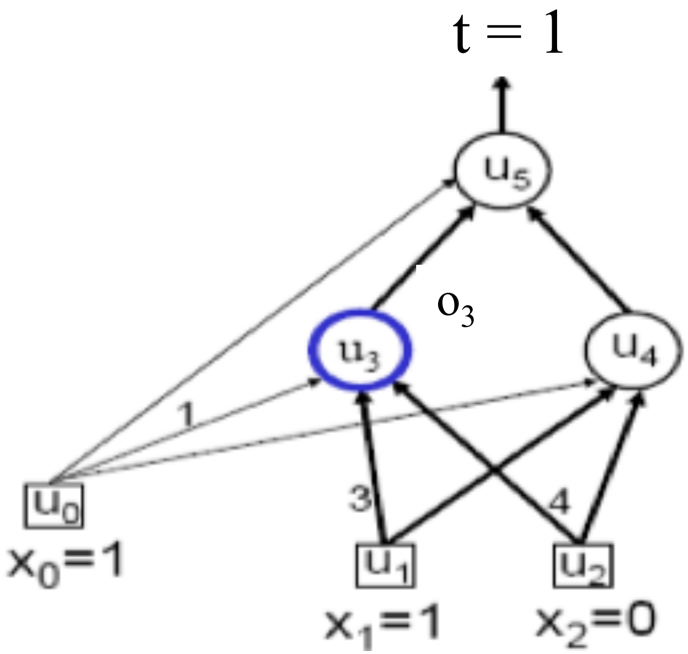
  $x_1=1$, $x_2=0$, $t = 1$



t = 1

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

MLP/BP: Exemplo



$$net_j = \sum_{i=0,\ldots,m} w_{ij} x_t$$

$$o_j = f(net_j) = \frac{1}{1+e^{-net_j}}$$

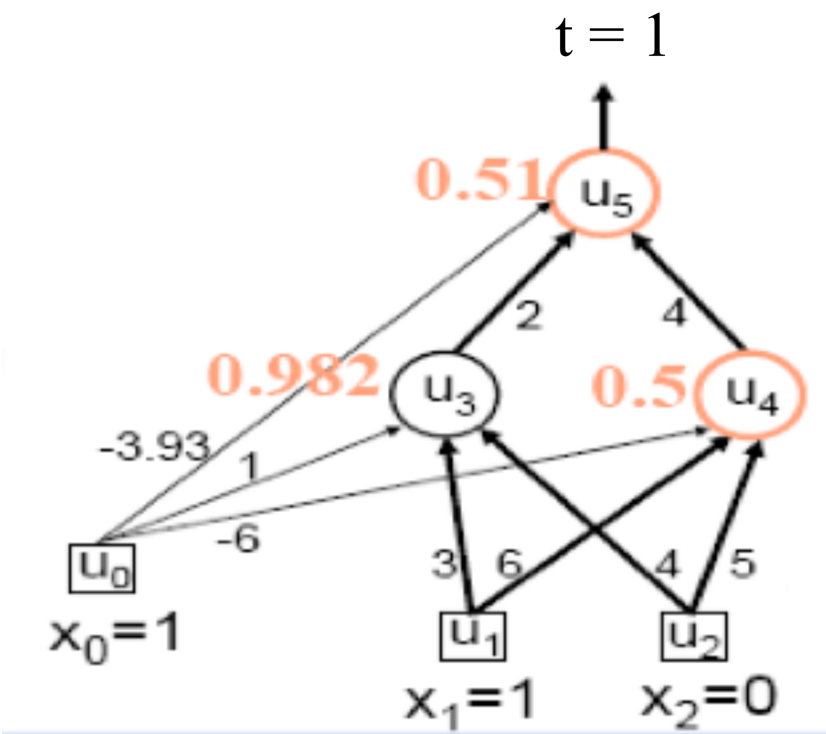
<u>Saída do neurônio u_3</u>

$$net_3 = 1x1 + 3x1 + 4x0 = 4$$

$$o_3 = f(4) = \frac{1}{1+e^{-4}} = 0.982$$

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

MLP/BP: Exemplo



Neurônio	Somatório (net_j)	Saída (o_j)
u_3	4	0.982
u_4	0	0.5
u_5	0.04	0.51

$$\text{Erro} = (o_{target} - o_5) = (1 - 0.51) = 0.49$$

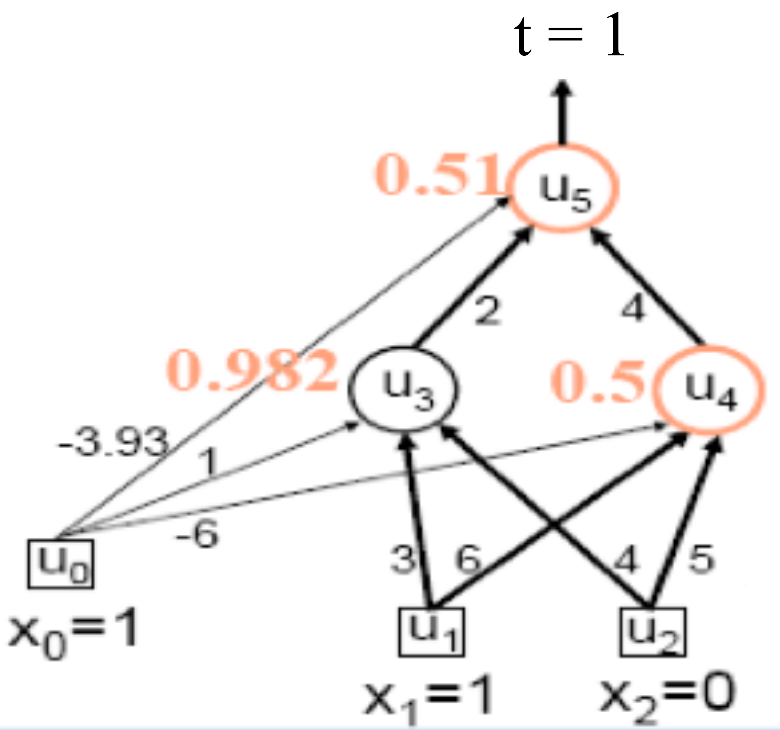
Descida do Gradiente

- Inicialize os pesos da rede aleatoriamente
- Enquanto (não converge)
  - Compute o gradiente da função de erro considerando os pesos
  - Atualize os pesos
- Retorne os pesos

Backpropagation: Calcular a derivada do erro em uma camada e propagar para a anterior

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

MLP/BP: Exemplo



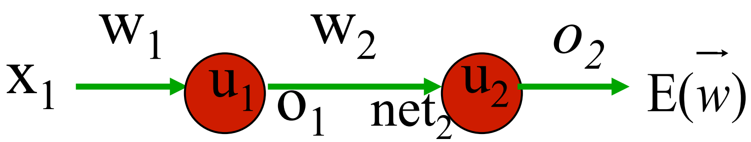
Neurônio	Somatório (net_j)	Saída (o_j)
u_3	4	0.982
u_4	0	0.5
u_5	0.04	0.51

Erro = $(o_{target} - o_5) = (1 - 0.51) = 0.49$

o_5 é dado por uma composição de funções

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

MLP – Backpropagation/ Pra frente



$$o_1 = f(x_1 w_1) = \frac{1}{1+e^{-x_1 w_1}}$$

$$net_2 = o_1 w_2$$

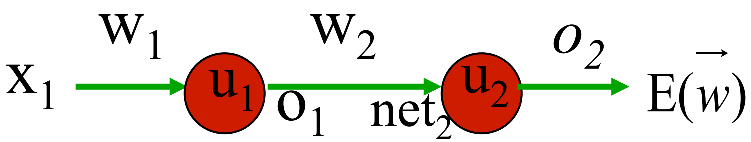
$$o_2 = f(net_2) = \frac{1}{1+e^{-net_2}}$$

Propagação para trás

- Como a função de erro é dada por uma composição de funções, o backpropagation usa a regra da cadeia para calcular as derivadas

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

MLP – Backpropagation – Pra trás



$$\frac{\partial E(\vec{w})}{\partial w_2} = \frac{\partial E(\vec{w})}{\partial o_2} \times \frac{\partial o_2}{\partial \boxed{w_2}}$$

Regra da cadeia

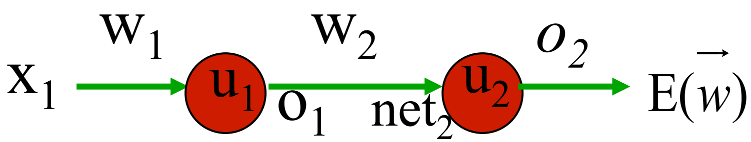
$$\boxed{net_2 = o_1 w_2}$$

$$\boxed{o_2 = f(net_2) = \frac{1}{1 + e^{-net_2}}}$$

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

MLP - Backpropagation



$$\frac{\partial E(\vec{w})}{\partial w_1} = \frac{\partial E(\vec{w})}{\partial o_2} \times \frac{\partial o_2}{\partial \boxed{w_1}}$$

Regra da cadeia

$$\frac{\partial E(\vec{w})}{\partial w_1} = \frac{\partial E(\vec{w})}{\partial o_2} \times \frac{\partial o_2}{\partial net_2} \times \frac{\partial net_2}{\partial w_1}$$

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Definição do erro

$$\boxed{E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in output} (t_{kd} - o_{kd})^2}$$

D: número de exemplos de treinamento

output: conjunto de neurônios da camada de saída

- Em redes de múltiplas camadas, a superfície de erro pode ter vários ótimos locais
  - Descida do gradiente não tem garantia de convergência para ótimo global.

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Back Propagation: Propagação pra trás

2. Compute o erro para cada neurônio da camada de saída

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. Compute o erro para cada neurônio das camadas escondidas

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Atualize todos os pesos da rede

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

onde

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

DCC
DEPARTAM
CIÊNCIA D

Back Propagation: Propagação pra trás

Derivada da sigmoide $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

2. Compute o erro para cada neurônio da camada de saída

$$\delta_k \leftarrow \boxed{o_k(1 - o_k)}\boxed{(t_k - o_k)}$$

Erro em relação a saída esperada

3.Compute o erro para cada neurônio das camadas escondidas

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k} \delta_k$$

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Back Propagation: Propagação pra trás

2. Compute o erro para cada neurônio da camada de saída

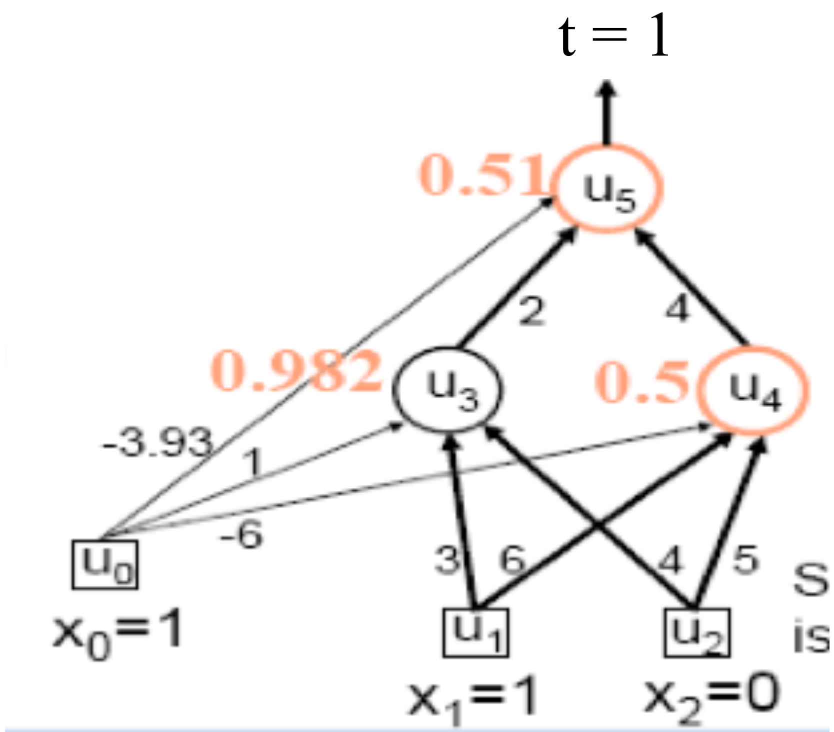
$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3.Compute o erro para cada neurônio das camadas escondidas

$$\delta_h \leftarrow o_h(1 - o_h)\boxed{\sum_{k \in outputs} w_{h,k}\delta_k}$$

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

MLP/BP: Exemplo

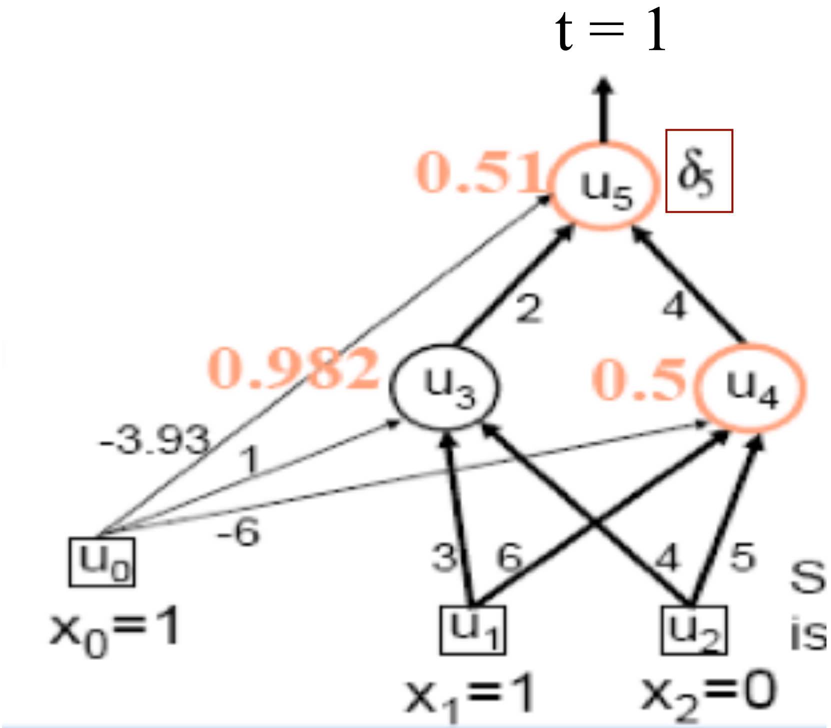


Neurônio	Somatório (net_j)	Saída (o_j)
u_3	4	0.982
u_4	0	0.5
u_5	0.04	0.51

$$\text{Erro} = (o_{target} - o_5) = (1 - 0.51) = 0.49$$

o_5 é dado por uma composição de funções

MLP/BP: Exemplo

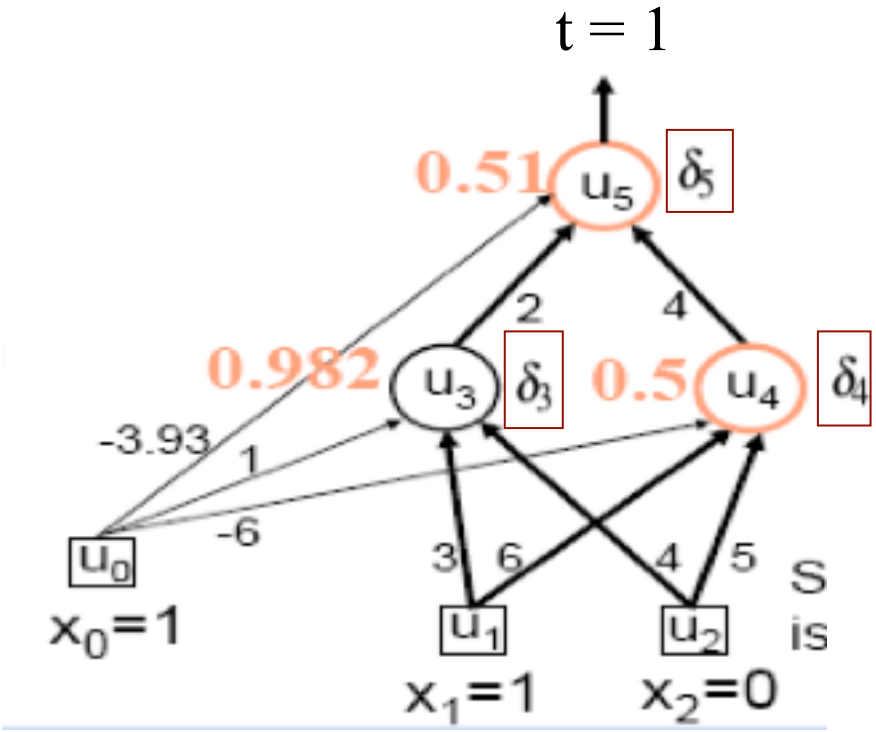


Cálculo dos valores de delta, começando pela camada de saída

$$\delta_5 = o_5(1-o_5)(t-o_5)$$
$$= 0.51(1-0.51)\times 0.49$$
$$= 0.1225$$

MLP/BP: Exemplo

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS



Camada Escondida

$$\delta_4 = o_4(1-o_4)(w_{45}\delta_5)$$
$$= 0.5(1-0.5)\times 4\times 0.1225$$
$$= 0.1225$$

$$\delta_3 = o_3(1-o_3)(w_{35}\delta_5)$$
$$= 0.982(1-0.982)\times 2\times 0.1225$$
$$= 0.0043$$

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

Back Propagation
Propagação pra trás

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

2. Compute o erro para cada neurônio da camada de saída

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. Compute o erro para cada neurônio das camadas escondidas

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Atualize todos os pesos da rede

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

onde

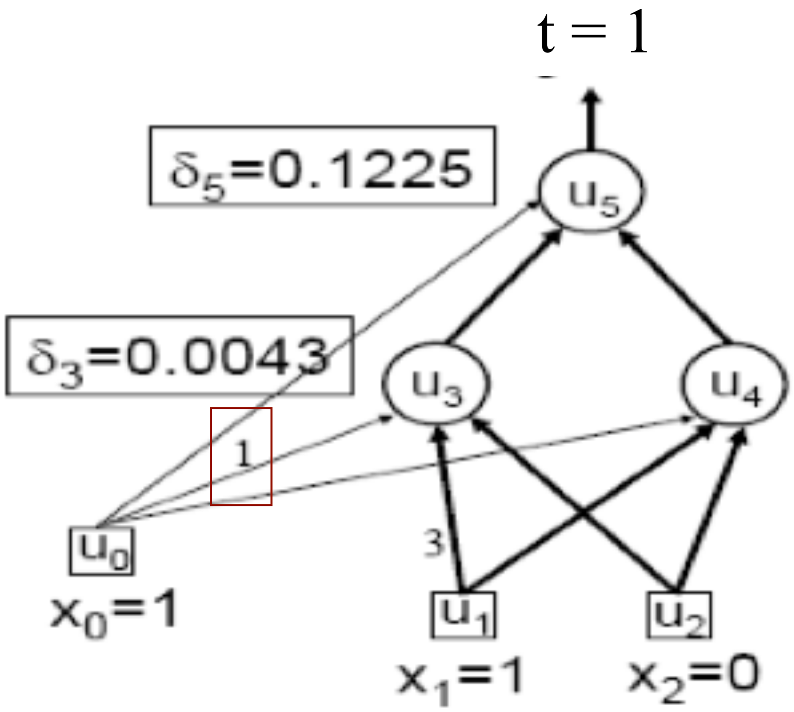
$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

DCC
DEPARTAMENTO I
CIÊNCIA DA COMF

MLP/BP: Exemplo

U F *m* G
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

- Atualização de pesos pela regra delta



0.1

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

- Atualização do peso do bias em u_3

$$\Delta_{w_{03}} = \eta \delta_3 x_0$$
$$= 0.1 \times 0.0043 \times 1$$
$$= 0.004$$
$$w_{03} = w_{03} + \Delta_{w_{03}}$$
$$= 1 + 0.0004 = 1.0004$$

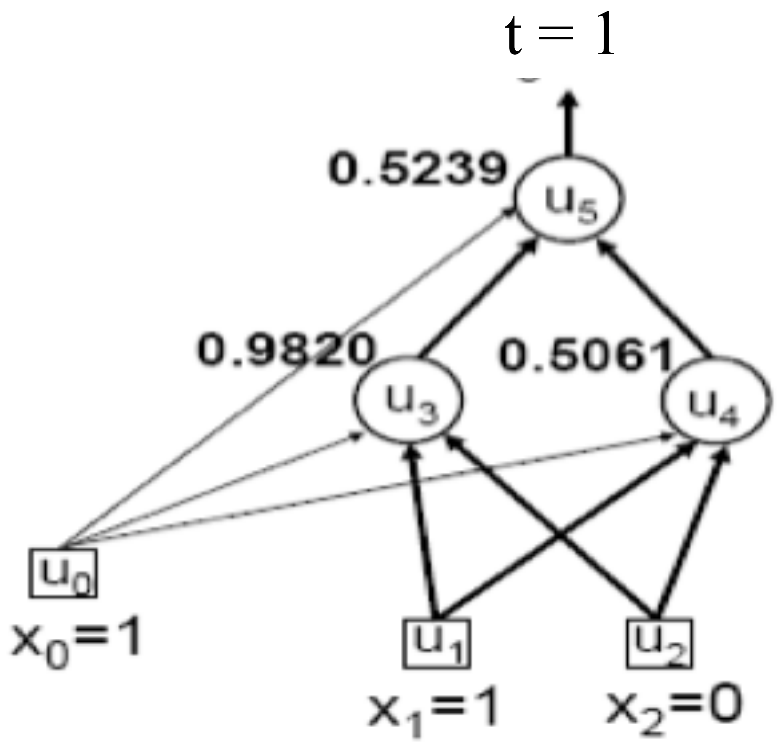
DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Mesmo procedimento para todos w_{ij}

i	j	w_{ij}	δ_j	o_i	Peso atualizado
0	3	**1**	0.0043	1.0	**1.0004**
1	3	**3**	0.0043	1.0	**3.0004**
2	3	**4**	0.0043	0.0	**4.0000**
0	4	**-6**	0.1225	1.0	**-5.9878**
1	4	**6**	0.1225	1.0	**6.0123**
2	4	**5**	0.1225	0.0	**5.0000**
0	5	**-3.92**	0.1225	1.0	**-3.9078**
3	5	**2**	0.1225	0.9820	**2.0120**
4	5	**4**	0.1225	0.5	**4.0061**

Próximo passo pra frente



$o_3 = f(4.0008) = 0.9820$

$o_4 = f(0.0245) = 0.5061$

$o_5 = f(0.0955) = 0.5239$

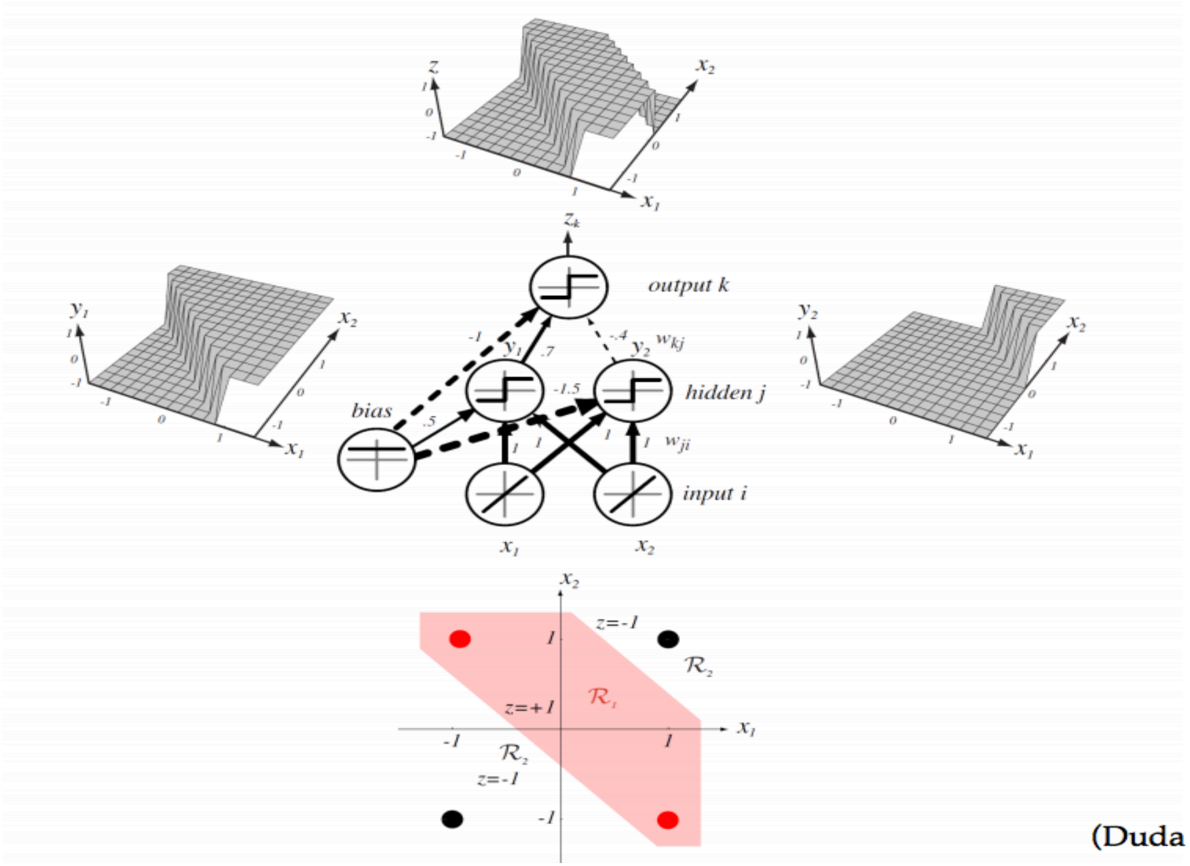
Erro = 1 − 0.5239 = 0.476

Erro no passo anterior = 0.49

Redução do erro em 0.014

Referência: "Neural Network Learning & Expert Systems, Stephen Gallant

O MLP resolve o problema do XOR?



(Duda et al)

Leitura Recomendada

- Livro Online, A Brief Introduction to Neural Networks, http://www.dkriesel.com/en/science/neural_networks, Part2 II

- http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Derivação do erro para sigmoide

$$
\left.\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \boxed{o_d})
\end{aligned}\right\} \text{Igual para o perceptron de uma camada}
$$

$$
\begin{aligned}
&= \sum_d (t_d - o_d) \left(- \frac{\partial o_d}{\partial w_i} \right) \\
&= - \sum_d (t_d - o_d) \frac{\partial o_d}{\partial net_d} \frac{\partial net_d}{\partial w_i}
\end{aligned}
$$

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

Derivação do erro para sigmoide

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&\quad 2(t_d - o_d)\,\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\left(-\frac{\partial o_d}{\partial w_i}\right) \\
&= -\sum_d (t_d - o_d)\,\frac{\partial o_d}{\partial net_d}\,\frac{\partial net_d}{\partial w_i}
\end{aligned}$$

Igual para o perceptron de uma camada

$$\boxed{o_j = f(net_j) = \frac{1}{1+e^{-net_j}}}$$

$$\boxed{net_j = \sum_{i=0,\ldots,m} w_{ij}x_t}$$

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

Derivação do erro para sigmoide

Derivada da sigmoide

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

Sabemos que:

Sigmoide

Termo 1

$$\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d)$$

Termo 2

$$\frac{\partial net_d}{\partial w_i} = \frac{\partial(\vec{w} \cdot \vec{x}_d)}{\partial w_i} = x_{i,d}$$

Então:

$$\boxed{\frac{\partial E}{\partial w_i} = -\sum_{d \in D}(t_d - o_d)o_d(1 - o_d)x_{i,d}}$$

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO