Ciência de Dados

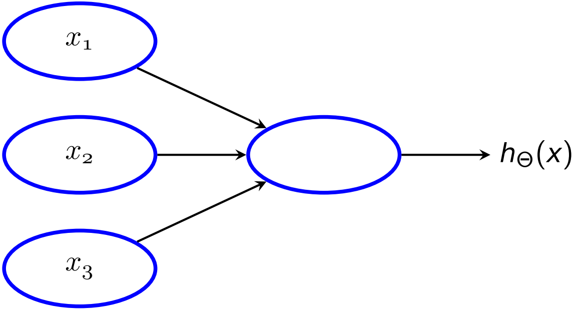
Redes Neurais Multicamadas

Renato Moraes Silva

renato.silva@facens.br

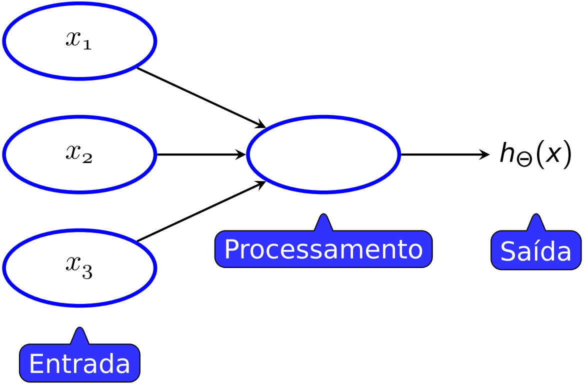
Representação

- Modelo neural: unidade logística



Representação

- Modelo neural: unidade logística

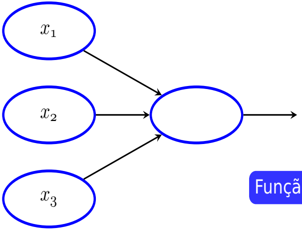


Representação

- Modelo neural: unidade logística

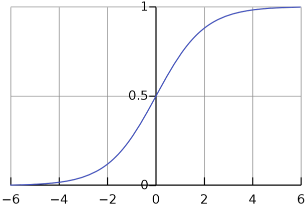
$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

Pesos



$$h_\theta(x) = g\left(\theta^T x\right) = \frac{1}{1 + e^{-\theta^T x}}$$

Função de ativação sigmoidal (logística)

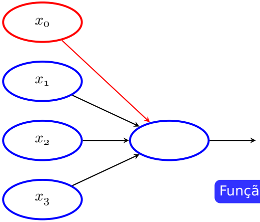


Representação

- Modelo neural: unidade logística

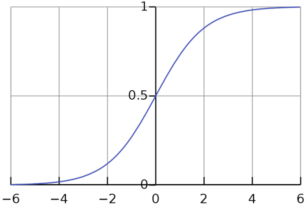
$$x = \begin{bmatrix} \color{red}{x_0} \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \color{red}{\theta_0} \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

unidade de viés ($x_0 = 1$)

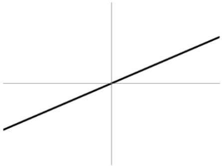


$$h_\theta(x) = g\left(\theta^T x\right) = \frac{1}{1 + e^{-\theta^T x}}$$

Função de ativação sigmoidal (logística)

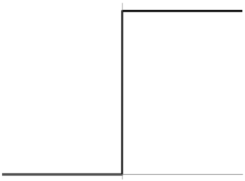


Funções de ativação



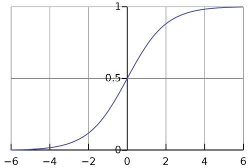
(a) Linear

$$f(x) = x$$



(b) Limiar

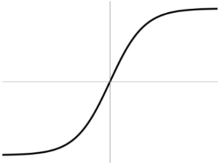
$$y = \varphi(u) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



(c) Sigmoidal

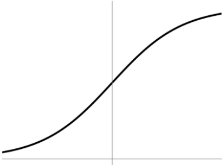
$$f(x) = \frac{1}{1 + e^{-x}}$$

Funções de ativação



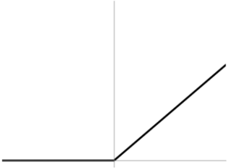
(d) Tangente hiperbólica

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



(e) Gaussiana

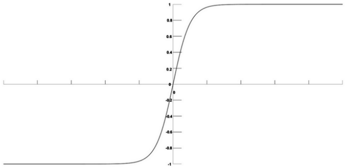
$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$



(f) Linear retificada (ReLU)

$$f(x) = \max(0, x)$$

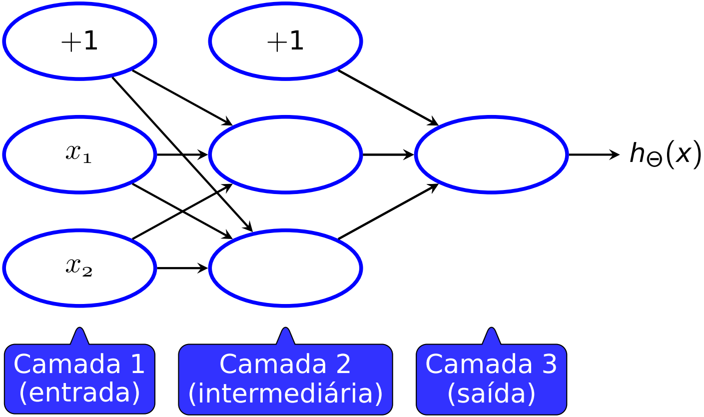
Funções de ativação



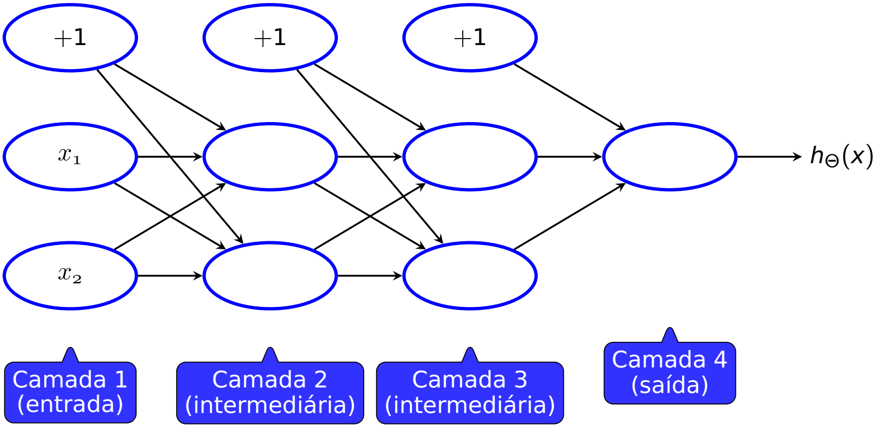
(g) Softmax

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

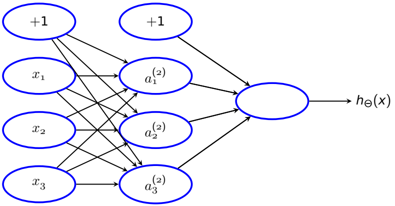
Arquitetura (1)



Arquitetura (2)



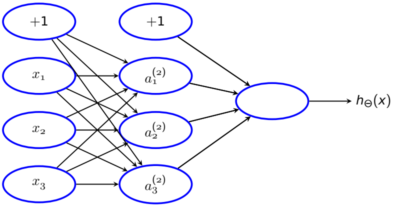
Representação



$a_i^{(j)}$ = valor de ativação do neurônio *i* na camada *j*

$\Theta^{(j)}$ = matriz de pesos de controle da função de mapeamento da camada *j* para a camada $j+1$

Representação

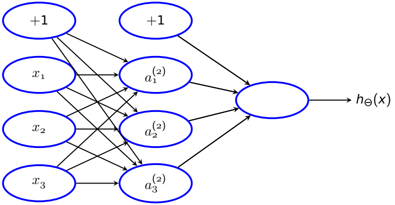


$a_i^{(j)}$ = valor de ativação do neurônio *i* na camada *j*

$\Theta^{(j)}$ = matriz de pesos de controle da função de mapeamento da camada *j* para a camada $j+1$

$$
\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) = g\left(\Theta_{10}^{(1)} * x_0 + \Theta_{11}^{(1)} * x_1 + \Theta_{12}^{(1)} * x_2 + \Theta_{13}^{(1)} * x_3\right) \\
a_2^{(2)} &= g\left(z_2^{(2)}\right) = g\left(\Theta_{20}^{(1)} * x_0 + \Theta_{21}^{(1)} * x_1 + \Theta_{22}^{(1)} * x_2 + \Theta_{23}^{(1)} * x_3\right) \\
a_3^{(2)} &= g\left(z_3^{(2)}\right) = g\left(\Theta_{30}^{(1)} * x_0 + \Theta_{31}^{(1)} * x_1 + \Theta_{32}^{(1)} * x_2 + \Theta_{33}^{(1)} * x_3\right) \\
h_\Theta(x) = a_1^{(3)} &= g\left(z_1^{(3)}\right) = g\left(\Theta_{10}^{(2)} * a_0^{(2)} + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)} + \Theta_{13}^{(2)} * a_3^{(2)}\right)
\end{aligned}
$$

Representação

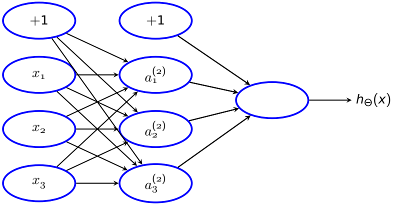


$a_i^{(j)}$ = valor de ativação do neurônio *i* na camada *j*

$\Theta^{(j)}$ = matriz de pesos de controle da função de mapeamento da camada *j* para a camada $j+1$

$$
\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) = g\left(\Theta_{10}^{(1)} * x_0 + \Theta_{11}^{(1)} * x_1 + \Theta_{12}^{(1)} * x_2 + \Theta_{13}^{(1)} * x_3\right)\\
a_2^{(2)} &= g\left(z_2^{(2)}\right) = g\left(\Theta_{20}^{(1)} * x_0 + \Theta_{21}^{(1)} * x_1 + \Theta_{22}^{(1)} * x_2 + \Theta_{23}^{(1)} * x_3\right)\\
a_3^{(2)} &= g\left(z_3^{(2)}\right) = g\left(\Theta_{30}^{(1)} * x_0 + \Theta_{31}^{(1)} * x_1 + \Theta_{32}^{(1)} * x_2 + \Theta_{33}^{(1)} * x_3\right)\\
h_\Theta(x) = a_1^{(3)} &= g\left(z_1^{(3)}\right) = g\left(\Theta_{10}^{(2)} * a_0^{(2)} + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)} + \Theta_{13}^{(2)} * a_3^{(2)}\right)
\end{aligned}
$$

Representação



$a_i^{(j)}$ = valor de ativação do neurônio *i* na camada *j*

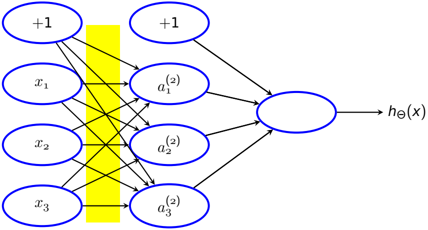
$\Theta^{(j)}$ = matriz de pesos de controle da função de mapeamento da camada *j* para a camada $j+1$

$$
\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) = g\left(\Theta_{10}^{(1)} * x_0 + \Theta_{11}^{(1)} * x_1 + \Theta_{12}^{(1)} * x_2 + \Theta_{13}^{(1)} * x_3\right)\\
a_2^{(2)} &= g\left(z_2^{(2)}\right) = g\left(\Theta_{20}^{(1)} * x_0 + \Theta_{21}^{(1)} * x_1 + \Theta_{22}^{(1)} * x_2 + \Theta_{23}^{(1)} * x_3\right)\\
a_3^{(2)} &= g\left(z_3^{(2)}\right) = g\left(\Theta_{30}^{(1)} * x_0 + \Theta_{31}^{(1)} * x_1 + \Theta_{32}^{(1)} * x_2 + \Theta_{33}^{(1)} * x_3\right)\\
h_\Theta(x) = a_1^{(3)} &= g\left(z_1^{(3)}\right) = g\left(\Theta_{10}^{(2)} * a_0^{(2)} + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)} + \Theta_{13}^{(2)} * a_3^{(2)}\right)
\end{aligned}
$$

Se a rede contém t_j unidades na camada j, t_{j+1} unidades na camada $j+1$, então $\Theta^{(j)}$ terá dimensão $t_{j+1} \times (t_j + 1)$.
Por exemplo, na rede neural deste slide, a matriz da camada 1 para a 2 ($\Theta^{(1)}$) tem dimensão 3 x 4.

Forward propagation

- Na etapa **forward propagation**, pode ser usada multiplicação matricial

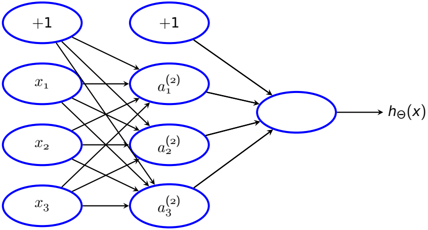


$$
\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) = g\left(\Theta_{10}^{(1)} * x_0 + \Theta_{11}^{(1)} * x_1 + \Theta_{12}^{(1)} * x_2 + \Theta_{13}^{(1)} * x_3\right) \\
a_2^{(2)} &= g\left(z_2^{(2)}\right) = g\left(\Theta_{20}^{(1)} * x_0 + \Theta_{21}^{(1)} * x_1 + \Theta_{22}^{(1)} * x_2 + \Theta_{23}^{(1)} * x_3\right) \\
a_3^{(2)} &= g\left(z_3^{(2)}\right) = g\left(\Theta_{30}^{(1)} * x_0 + \Theta_{31}^{(1)} * x_1 + \Theta_{32}^{(1)} * x_2 + \Theta_{33}^{(1)} * x_3\right) \\
h_\Theta(x) = a_1^{(3)} &= g\left(z_1^{(3)}\right) = g\left(\Theta_{10}^{(2)} * a_0^{(2)} + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)} + \Theta_{13}^{(2)} * a_3^{(2)}\right)
\end{aligned}
$$

$$
x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}
$$

Forward propagation

- Na etapa **forward propagation**, pode ser usada multiplicação matricial



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

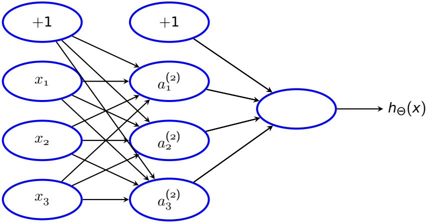
$$z^{(2)} = \Theta^{(1)} x$$

$$a^{(2)} = g\left(z^{(2)}\right)$$

$$\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) = g\left(\Theta_{10}^{(1)} * x_0 + \Theta_{11}^{(1)} * x_1 + \Theta_{12}^{(1)} * x_2 + \Theta_{13}^{(1)} * x_3\right) \\
a_2^{(2)} &= g\left(z_2^{(2)}\right) = g\left(\Theta_{20}^{(1)} * x_0 + \Theta_{21}^{(1)} * x_1 + \Theta_{22}^{(1)} * x_2 + \Theta_{23}^{(1)} * x_3\right) \\
a_3^{(2)} &= g\left(z_3^{(2)}\right) = g\left(\Theta_{30}^{(1)} * x_0 + \Theta_{31}^{(1)} * x_1 + \Theta_{32}^{(1)} * x_2 + \Theta_{33}^{(1)} * x_3\right) \\
h_\Theta(x) = a_1^{(3)} &= g\left(z_1^{(3)}\right) = g\left(\Theta_{10}^{(2)} * a_0^{(2)} + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)} + \Theta_{13}^{(2)} * a_3^{(2)}\right)
\end{aligned}$$

Forward propagation

- Na etapa **forward propagation**, pode ser usada multiplicação matricial



$$
\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) = g\left(\Theta_{10}^{(1)} * x_0 + \Theta_{11}^{(1)} * x_1 + \Theta_{12}^{(1)} * x_2 + \Theta_{13}^{(1)} * x_3\right) \\
a_2^{(2)} &= g\left(z_2^{(2)}\right) = g\left(\Theta_{20}^{(1)} * x_0 + \Theta_{21}^{(1)} * x_1 + \Theta_{22}^{(1)} * x_2 + \Theta_{23}^{(1)} * x_3\right) \\
a_3^{(2)} &= g\left(z_3^{(2)}\right) = g\left(\Theta_{30}^{(1)} * x_0 + \Theta_{31}^{(1)} * x_1 + \Theta_{32}^{(1)} * x_2 + \Theta_{33}^{(1)} * x_3\right) \\
h_\Theta(x) = a_1^{(3)} &= g\left(z_1^{(3)}\right) = g\left(\Theta_{10}^{(2)} * a_0^{(2)} + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)} + \Theta_{13}^{(2)} * a_3^{(2)}\right)
\end{aligned}
$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{(1)} x$$

$$a^{(2)} = g\left(z^{(2)}\right)$$

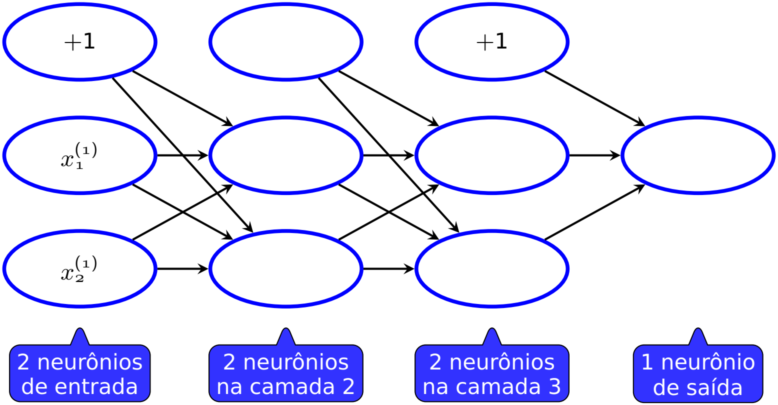
adicionar $a_0^{(2)} = 1$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$h_\Theta(x) = a^{(3)} = g\left(z^{(3)}\right)$$

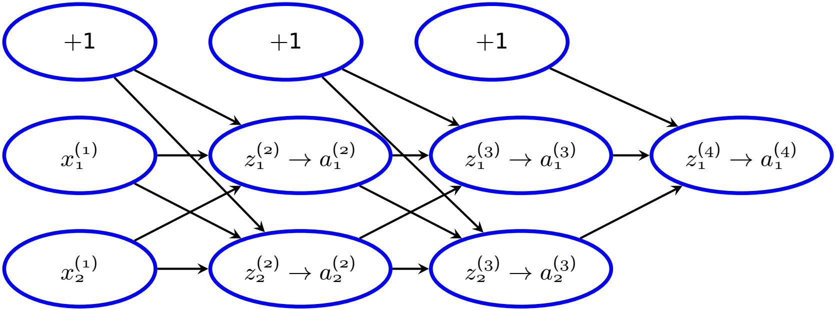
Exemplo

- Forward propagation



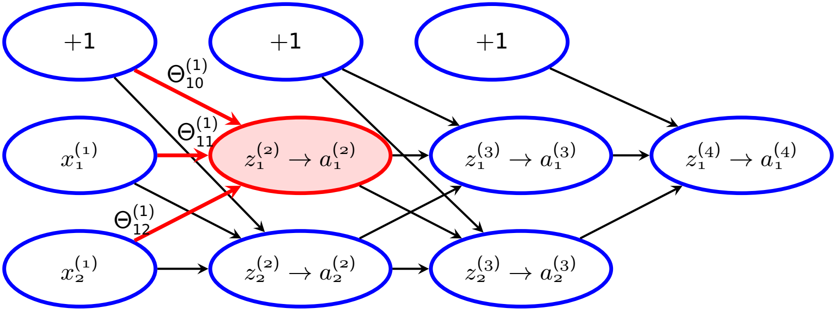
Exemplo

- Forward propagation



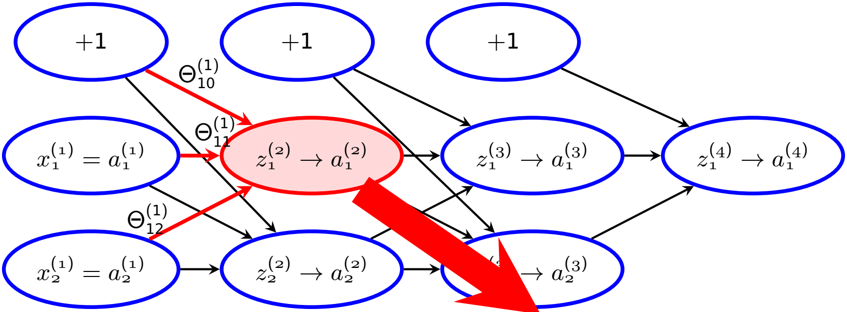
Exemplo

- Forward propagation



Exemplo

- Forward propagation

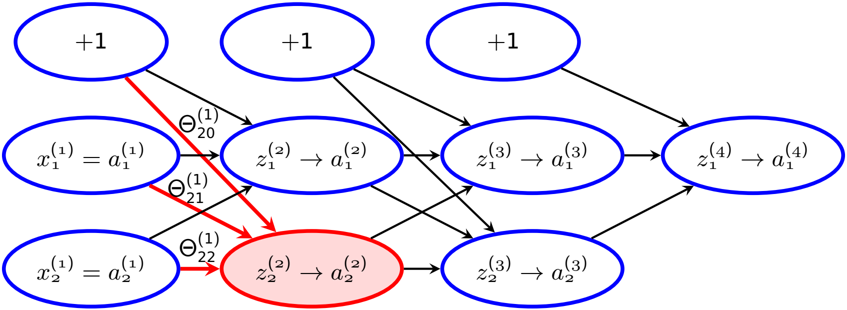


$$z_1^{(2)} = \Theta_{10}^{(1)} * 1 + \Theta_{11}^{(1)} * a_1^{(1)} + \Theta_{12}^{(1)} * a_2^{(1)}$$

$$a_1^{(2)} = g(z_1^{(2)})$$

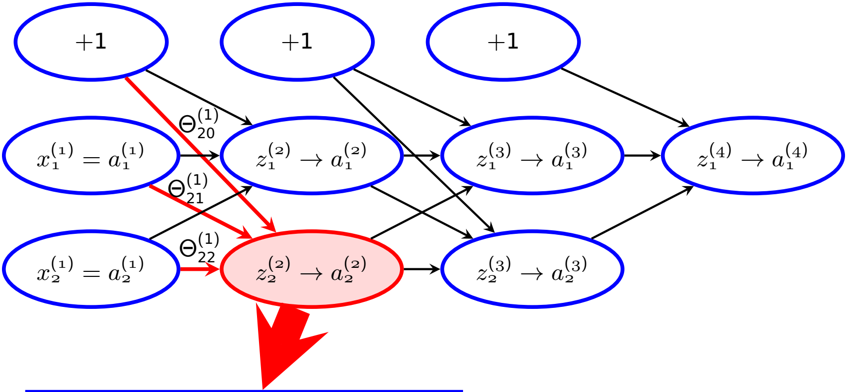
Exemplo

- Forward propagation



Exemplo

- Forward propagation

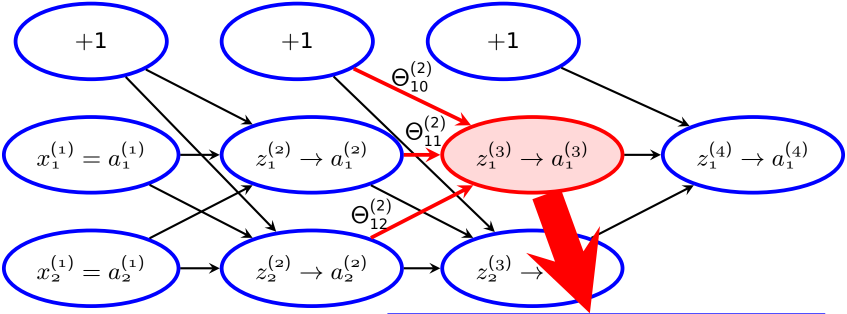


$$z_1^{(2)} = \Theta_{20}^{(1)} * 1 + \Theta_{21}^{(1)} * a_1^{(1)} + \Theta_{22}^{(1)} * a_2^{(1)}$$

$$a_2^{(2)} = g\left(z_2^{(2)}\right)$$

Exemplo

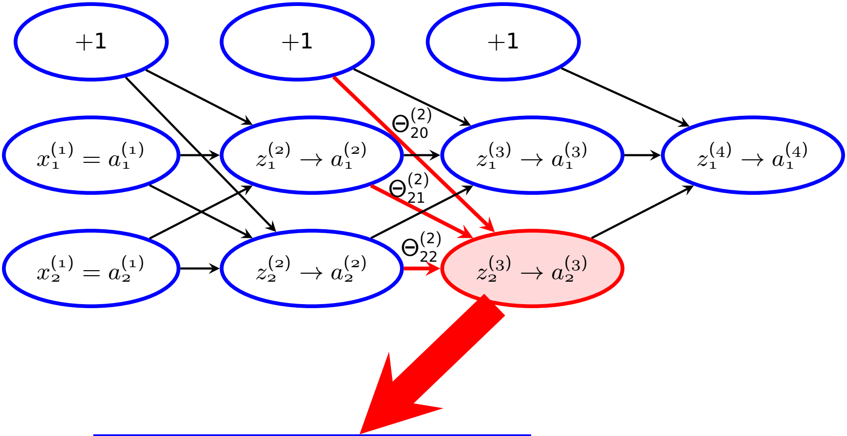
- Forward propagation



$$z_1^{(3)} = \Theta_{10}^{(2)} * 1 + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)}$$
$$a_1^{(3)} = g(z_1^{(3)})$$

Exemplo

- Forward propagation

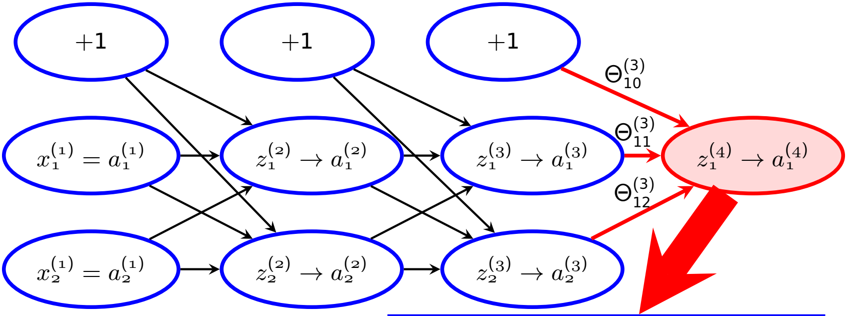


$$z_2^{(3)} = \Theta_{20}^{(2)} * 1 + \Theta_{21}^{(2)} * a_1^{(2)} + \Theta_{22}^{(2)} * a_2^{(3)}$$

$$a_2^{(3)} = g(z_2^{(3)})$$

Exemplo

- Forward propagation

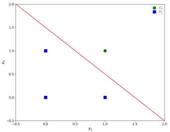


$$z_1^{(4)} = \Theta_{10}^{(3)} * 1 + \Theta_{11}^{(3)} * a_1^{(3)} + \Theta_{12}^{(3)} * a_2^{(2)}$$

$$a_1^{(4)} = h_\Theta(x) = g(z_1^{(4)})$$

Exemplo: função lógica AND

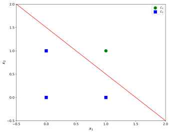
x_1	x_2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

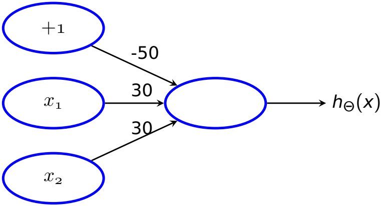


- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Considere os pesos: $\theta_0 = -50, \theta_1 = 30, \theta_2 = 30$.

Exemplo: função lógica AND

x_1	x_2	Classe
0	0	0
0	1	0
1	0	0
1	1	1

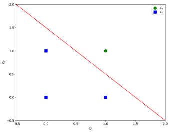


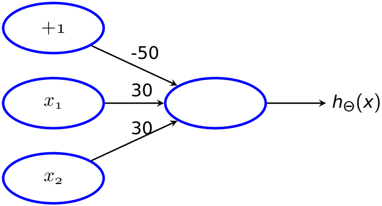


- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Considere os pesos: $\theta_0 = -50, \theta_1 = 30, \theta_2 = 30$.

Exemplo: função lógica AND

x_1	x_2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



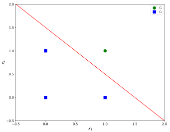


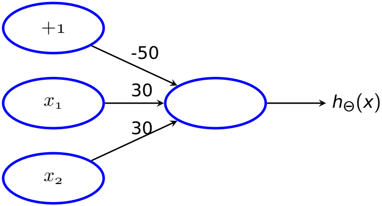
- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Considere os pesos: $\theta_0 = -50, \theta_1 = 30, \theta_2 = 30$.

x_1	x_2	$h_{\theta(x)}$
0	0	$-50 \times 1 + 30 \times 0 + 30 \times 0 = g(-50) \approx 0$

Exemplo: função lógica AND

x_1	x_2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



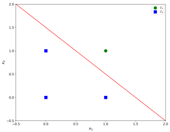


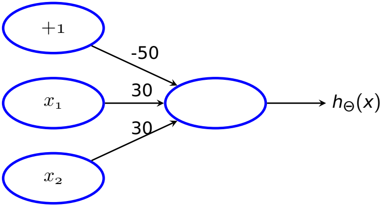
- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Considere os pesos: $\theta_0 = -50, \theta_1 = 30, \theta_2 = 30$.

x_1	x_2	$h_{\theta(x)}$
0	0	$-50 \times 1 + 30 \times 0 + 30 \times 0 = g(-50) \approx 0$
0	1	$-50 \times 1 + 30 \times 0 + 30 \times 1 = g(-20) \approx 0$

Exemplo: função lógica AND

x_1	x_2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



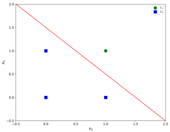


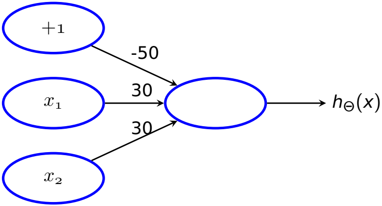
- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Considere os pesos: $\theta_0 = -50, \theta_1 = 30, \theta_2 = 30$.

x_1	x_2	$h_{\theta(x)}$
0	0	$-50 \times 1 + 30 \times 0 + 30 \times 0 = g(-50) \approx 0$
0	1	$-50 \times 1 + 30 \times 0 + 30 \times 1 = g(-20) \approx 0$
1	0	$-50 \times 1 + 30 \times 1 + 30 \times 0 = g(-20) \approx 0$

Exemplo: função lógica AND

x_1	x_2	Classe
0	0	0
0	1	0
1	0	0
1	1	1



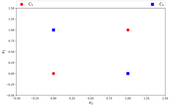


- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Considere os pesos: $\theta_0 = -50, \theta_1 = 30, \theta_2 = 30$.

x_1	x_2	$h_{\theta(x)}$
0	0	$-50 \times 1 + 30 \times 0 + 30 \times 0 = g(-50) \approx 0$
0	1	$-50 \times 1 + 30 \times 0 + 30 \times 1 = g(-20) \approx 0$
1	0	$-50 \times 1 + 30 \times 1 + 30 \times 0 = g(-20) \approx 0$
1	1	$-50 \times 1 + 30 \times 1 + 30 \times 1 = g(10) \approx 1$

Exemplo: função lógica XOR

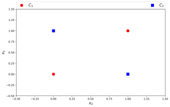
x_1	x_2	Classe
0	0	0
0	1	1
1	0	1
1	1	0

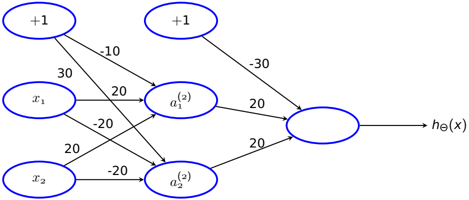


- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Só é possível resolver com pelo menos uma camada intermediária.

Exemplo: função lógica XOR

x_1	x_2	Classe
0	0	0
0	1	1
1	0	1
1	1	0



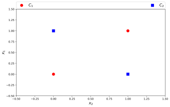


- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Só é possível resolver com pelo menos uma camada intermediária.

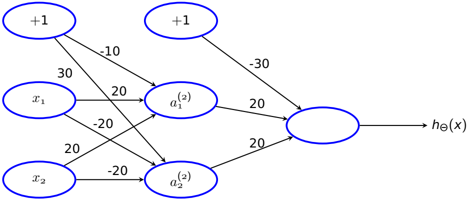
Exemplo: função lógica XOR

Facens

x_1	x_2	Classe
0	0	0
0	1	1
1	0	1
1	1	0



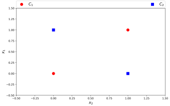
- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Só é possível resolver com pelo menos uma camada intermediária.

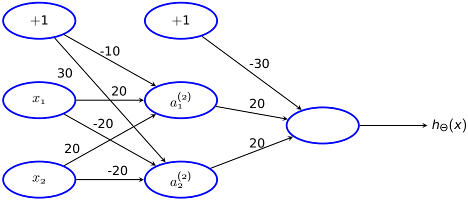


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\theta(x)}$
0	0	$g(-10) \approx 0$	$g(30) \approx 1$	$g(-10) \approx 0$

Exemplo: função lógica XOR

x_1	x_2	Classe
0	0	0
0	1	1
1	0	1
1	1	0



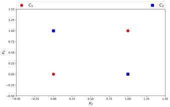


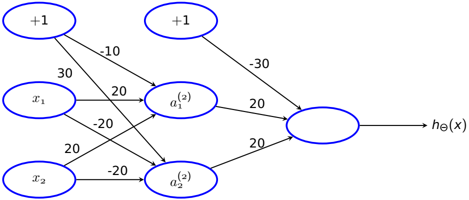
x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\theta(x)}$
0	0	$g(-10) \approx 0$	$g(30) \approx 1$	$g(-10) \approx 0$
0	1	$g(10) \approx 1$	$g(10) \approx 1$	$g(10) \approx 1$

- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Só é possível resolver com pelo menos uma camada intermediária.

Exemplo: função lógica XOR

x_1	x_2	Classe
0	0	0
0	1	1
1	0	1
1	1	0



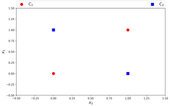


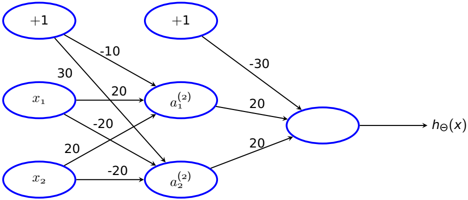
x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\theta(x)}$
0	0	$g(-10) \approx 0$	$g(30) \approx 1$	$g(-10) \approx 0$
0	1	$g(10) \approx 1$	$g(10) \approx 1$	$g(10) \approx 1$
1	0	$g(10) \approx 1$	$g(10) \approx 1$	$g(10) \approx 1$

- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Só é possível resolver com pelo menos uma camada intermediária.

Exemplo: função lógica XOR

x_1	x_2	Classe
0	0	0
0	1	1
1	0	1
1	1	0



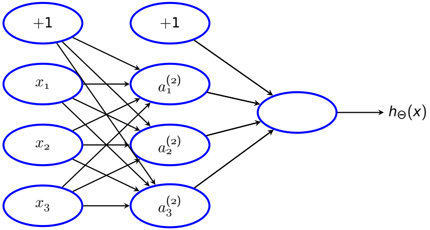


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\theta(x)}$
0	0	$g(-10) \approx 0$	$g(30) \approx 1$	$g(-10) \approx 0$
0	1	$g(10) \approx 1$	$g(10) \approx 1$	$g(10) \approx 1$
1	0	$g(10) \approx 1$	$g(10) \approx 1$	$g(10) \approx 1$
1	1	$g(30) \approx 1$	$g(-10) \approx 0$	$g(-10) \approx 0$

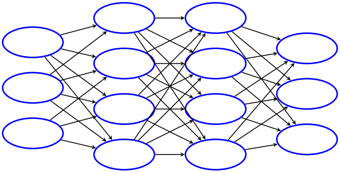
- Deve-se classificar os padrões de entrada em duas classes (1 e 0);
- Cada padrão de entrada tem dois componentes (x_1 e x_2);
- Só é possível resolver com pelo menos uma camada intermediária.

Classificação

- Na classificação **binária**, geralmente usa-se um único neurônio de saída
- Na classificação **multiclasse** ou **multilabel**, se a quantidade de classes é *n*, geralmente usa-se *n* neurônios de saída



Classificação binária ($y = 0$ ou 1)



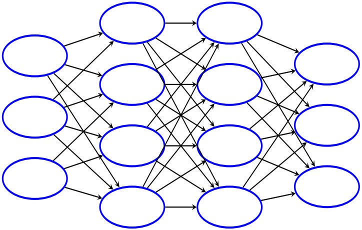
Classificação Muticlasse (K classes)

$y \in \mathbb{R}^K$. Exemplo:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Classe A Classe B Classe C

Notação



L = quantidade de camadas
s_t = número de neurônios,
excluindo o *bias*, na camada l

Na arquitetura ao lado: $L = 4$,
$s_1 = 3$, $s_2 = 4$, $s_3 = 4$, $s_4 = 3$

Função Custo (Loss)

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m} \left[\sum_{i=1}^{m} \sum_{k=1}^{K} -y_k^{(i)} \log \left(h_\Theta(x^{(i)})_k \right) - \left(1 - y_k^{(i)} \right) \log \left(1 - h_\Theta(x^{(i)})_k \right) \right]$$

Função Custo (Loss)

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$

Para cada
dado de entrada

Função Custo (Loss)

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$

Para cada
neurônio de saída

Função Custo (Loss)

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

Se $y = 0$

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} \cancel{-y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right)} - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$

Função Custo (Loss)

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

Se $y = 1$

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \cancel{\left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)}\right]$$

Função Custo (Loss) com regularização

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

Função Custo (Loss) com regularização

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

Resumindo: $+\frac{\lambda}{2m}$ * soma do quadrado de todos os pesos (exceto os pesos bias)

Função Custo (Loss) com regularização

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$

$$\boxed{+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2}$$

Conhecida como regularização L2

Resumindo: $+\frac{\lambda}{2m}$ * soma do quadrado de todos os pesos (exceto os pesos bias)

Função Custo (Loss) com regularização

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

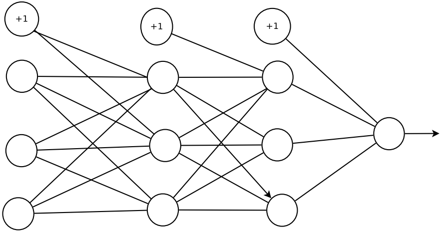
Conhecida como regularização L2

Existe também a regularização L1:

$$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left|\Theta_{ji}^{(l)}\right|$$

Resumindo: $+\frac{\lambda}{2m}$ * soma do quadrado de todos os pesos (exceto os pesos bias)

Dropout



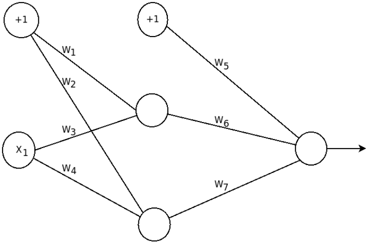
Exercício

- Considere a arquitetura de rede neural artificial mostrada na imagem. Suponha também que a função de ativação é a **sigmoidal**.

- Você possui uma pequena base de dados $X = [0.4; 1.0; -0.7; -0.9]$ com as seguintes classes: $Y = [1; 1; 0; 0]$.

- Os pesos da rede são $W = [0.5; 0.6; 0.7; 0.5; 0.8; 0.0; 0.6]$ (a ordem desses pesos é a mesma dos pesos mostrados na Figura).

- Qual o valor do custo de usar os pesos W como parâmetros da rede neural artificial considerando a base de dados X.
  - **Obs.**: Adicione regularização no cálculo, considerando o parâmetro de regularizacao $\lambda = 0.9$.



Backprojection

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

$$\min_\Theta J(\Theta)$$

Computar :

$J(\Theta)$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$$

Backpropagation

Considere que $h_\Theta(x) \in \mathbb{R}^K$ e que $h_\Theta(x)_i = i$-ésima saída.

$$J(\Theta) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} -y_k^{(i)} \log\left(h_\Theta(x^{(i)})_k\right) - \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})_k\right)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

$\min_\Theta J(\Theta)$

Computar :

$J(\Theta)$

$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Lembre-se da regra da cadeia. $\frac{dt}{dx} = \frac{dt}{dg}\frac{dg}{dx}$.

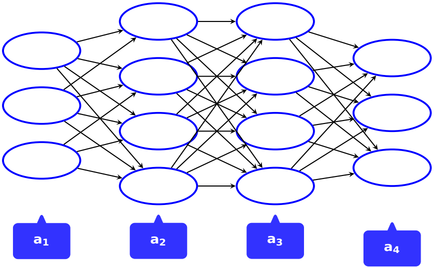
Exemplo: Seja $t = exp(x^2)$.

Para calcular $\frac{dt}{dx}$, podemos fazer $g = x^2$ e usar a regra da cadeia:

$$\frac{dt}{dx} = \frac{dt}{dg}\frac{dg}{dx} = exp(g)2x = exp(x^2)2x$$

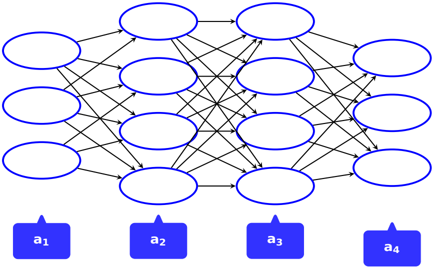
(a derivada de $exp(x)$ é $exp(x)$)

Gradiente descendente



Seja um único exemplo (x, y)

Gradiente descendente



Seja um único exemplo (x, y)

Forward propagation:

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g\left(z^{(2)}\right) \quad \left(\text{adicionar } a_0^{(2)}\right)$$

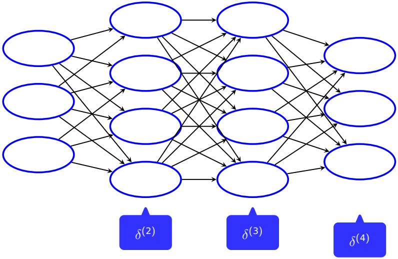
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$a^{(3)} = g\left(z^{(3)}\right)$$

$$z^{(4)} = \Theta^{(3)} a^{(3)} \quad \left(\text{adicionar } a_0^{(3)}\right)$$

$$a^{(4)} = h_\Theta(x) = g\left(z^{(4)}\right)$$

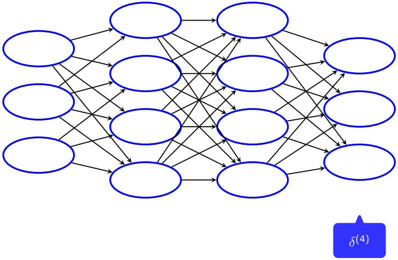
Gradiente descendente



Ideia:

Calcular $\delta_j^{(l)} =$ "erro" produzido por cada nó *j* da camada *l*.

Gradiente descendente



Backpropagation:

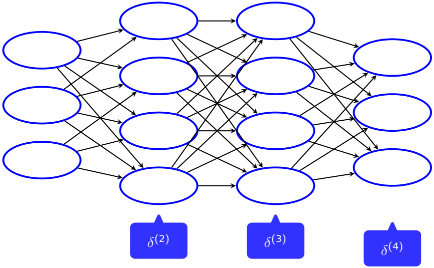
Para cada unidade de saída (L = 4)

$$\delta^{(4)} = a_j^{(4)} - y_j$$

Ideia:

Calcular $\delta_j^{(l)}$ = "erro" produzido por cada nó *j* da camada *l*.

Gradiente descendente



Backpropagation:

Para cada unidade de saída (L = 4)

$$\delta^{(4)} = a^{(4)} - y_j$$

$$\delta^{(3)} = \left(\Theta^{(3)}\right)^T \delta^{(4)} .* g' \left(z^{(3)}\right)$$

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* g' \left(z^{(2)}\right)$$

Ideia:

Calcular $\delta_j^{(l)} =$ "erro" produzido por cada nó *j* da camada *l*.

Derivada da função sigmoidal:

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

Portanto:

$$g'(z^{(l)}) = a^l .* (1 - a^l)$$

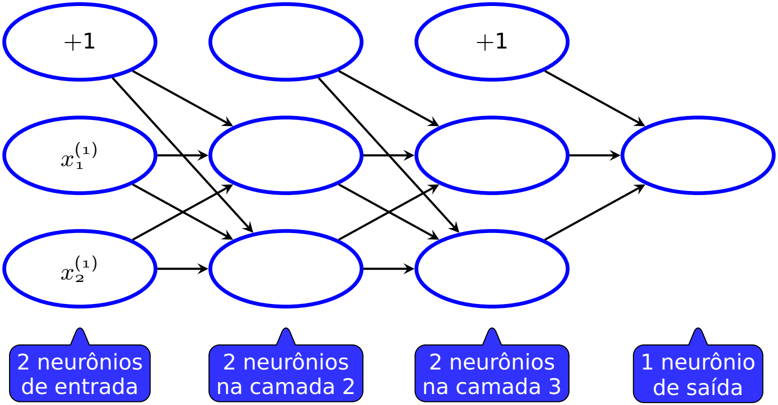
Backpropagation

- Base de treinamento: $\left\{ \left(x^{(1)}, y^{(1)}\right), ..., \left(x^{(m)}, y^{(m)}\right) \right\}$
- Inicializar $\Delta_{ij}^{(l)} = 0$ (para todo l, i, j)
- Para $i = 1 : m$
  - $a^{(1)} = x^{(i)}$
  - Aplicar **forward propagation** para calcular $a^{(l)}$ para $l = 2, 3, ..., L$
  - Calcular $\delta^{(L)} = a^{(L)} - y^{(i)}$
  - Calcular $\delta^{(L-1)}, \delta^{(L-2)}, ..., \delta^{(2)}$
  - Acumular as derivadas parciais de cada exemplo: $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
- Calcular a derivada da função custo:

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} \quad \Longrightarrow \quad D_{ij}^{(l)} := \begin{cases} \frac{1}{m} \Delta_{ij}^{(l)} & \text{se } j = 0 \\ \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{se } j \neq 0 \end{cases}$$

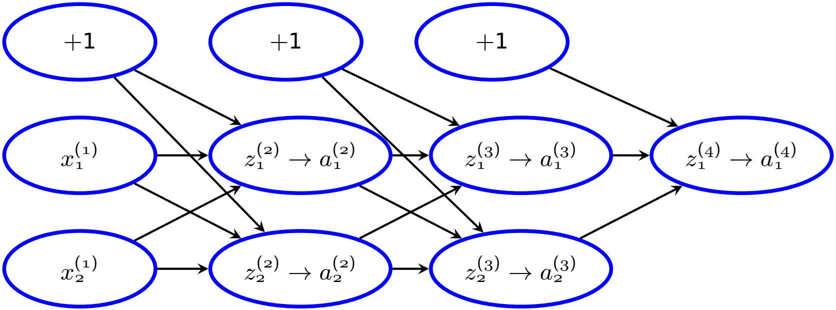
Exemplo

- Forward propagation



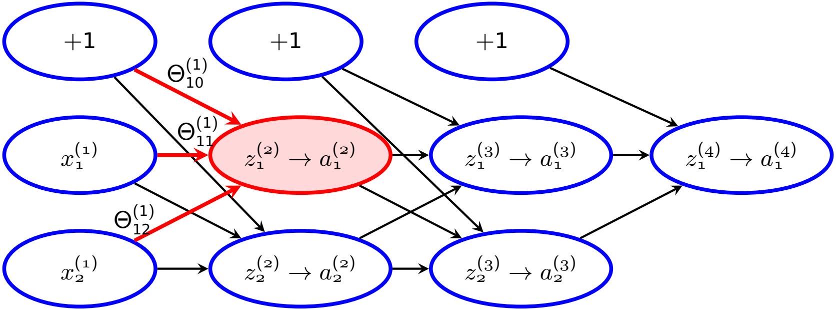
Exemplo

- Forward propagation



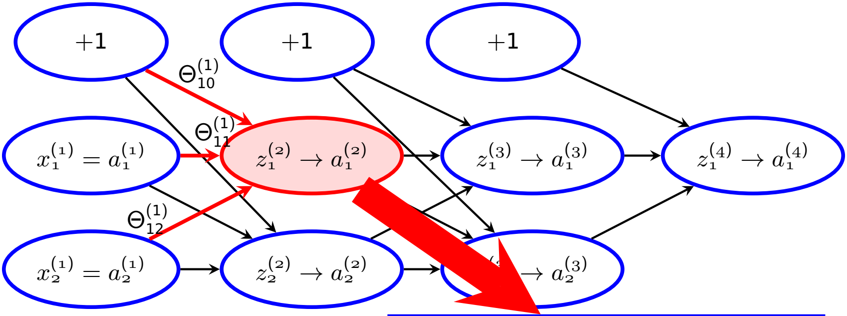
Exemplo

- Forward propagation



Exemplo

- Forward propagation

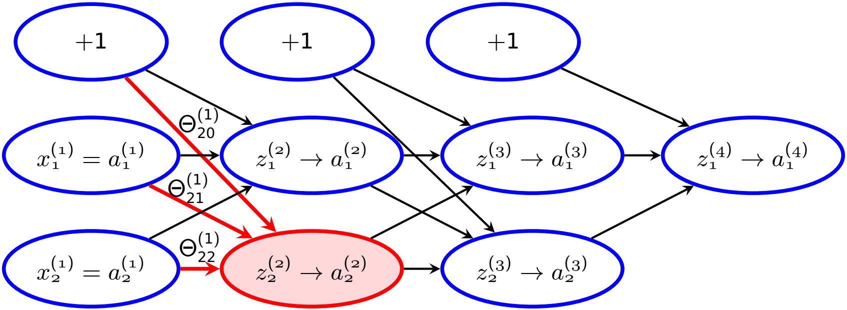


$$z_1^{(2)} = \Theta_{10}^{(1)} * 1 + \Theta_{11}^{(1)} * a_1^{(1)} + \Theta_{12}^{(1)} * a_2^{(1)}$$

$$a_1^{(2)} = g(z_1^{(2)})$$

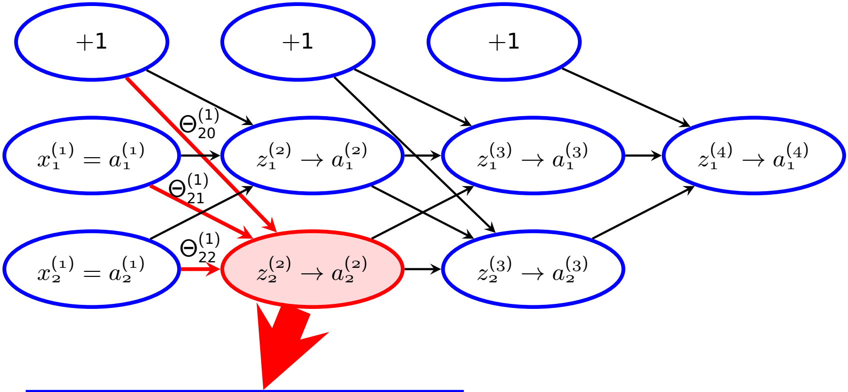
Exemplo

- Forward propagation



Exemplo

- Forward propagation

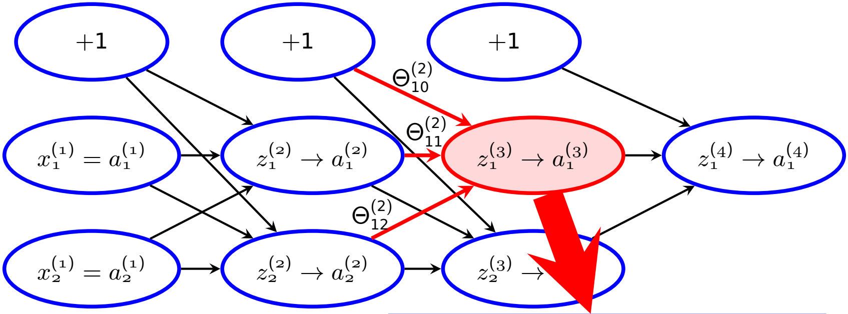


$$z_1^{(2)} = \Theta_{20}^{(1)} * 1 + \Theta_{21}^{(1)} * a_1^{(1)} + \Theta_{22}^{(1)} * a_2^{(1)}$$

$$a_2^{(2)} = g(z_2^{(2)})$$

Exemplo

- Forward propagation

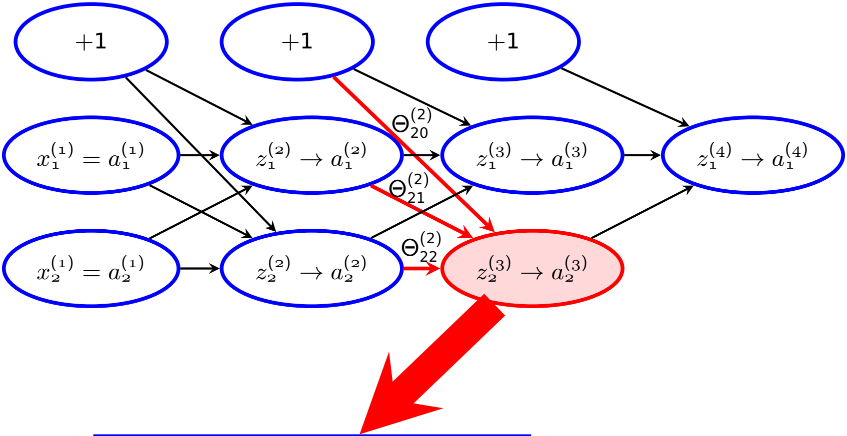


$$z_1^{(3)} = \Theta_{10}^{(2)} * 1 + \Theta_{11}^{(2)} * a_1^{(2)} + \Theta_{12}^{(2)} * a_2^{(2)}$$

$$a_1^{(3)} = g(z_1^{(3)})$$

Exemplo

- Forward propagation

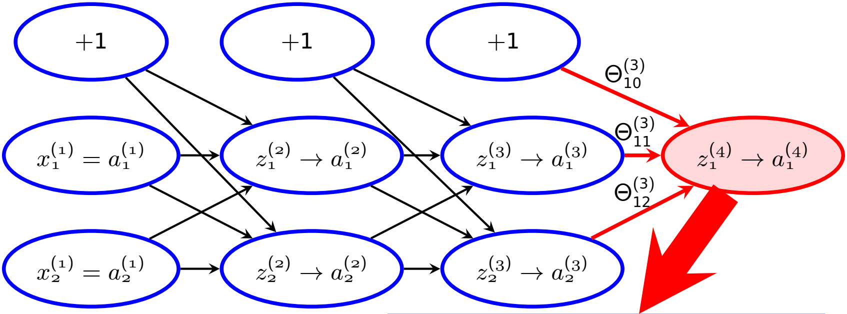


$$z_2^{(3)} = \Theta_{20}^{(2)} * 1 + \Theta_{21}^{(2)} * a_1^{(2)} + \Theta_{22}^{(2)} * a_2^{(3)}$$

$$a_2^{(3)} = g(z_2^{(3)})$$

Exemplo

- Forward propagation

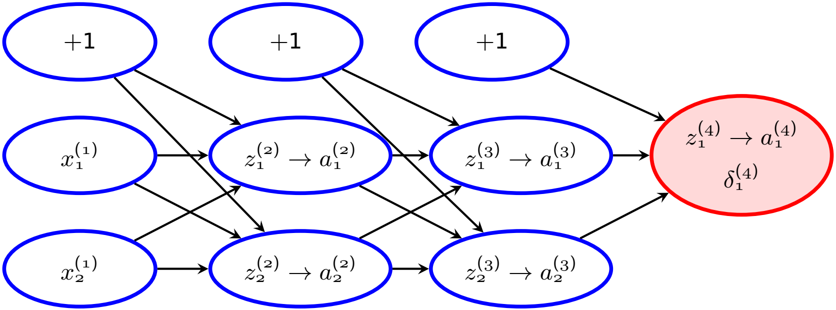


$$z_1^{(4)} = \Theta_{10}^{(3)} * 1 + \Theta_{11}^{(3)} * a_1^{(3)} + \Theta_{12}^{(3)} * a_2^{(2)}$$

$$a_1^{(4)} = h_\Theta(x) = g(z_1^{(4)})$$

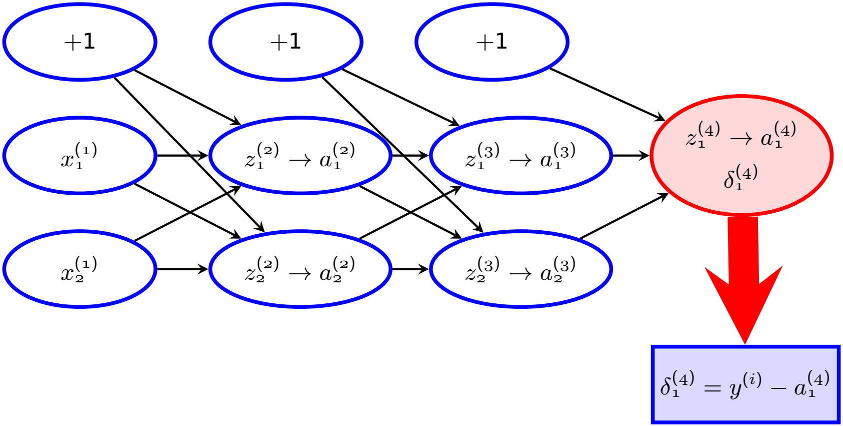
Exemplo

- Backpropagation



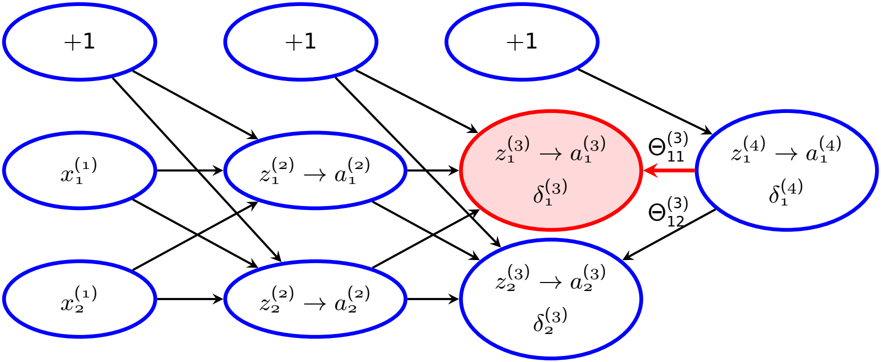
Exemplo

- Backpropagation



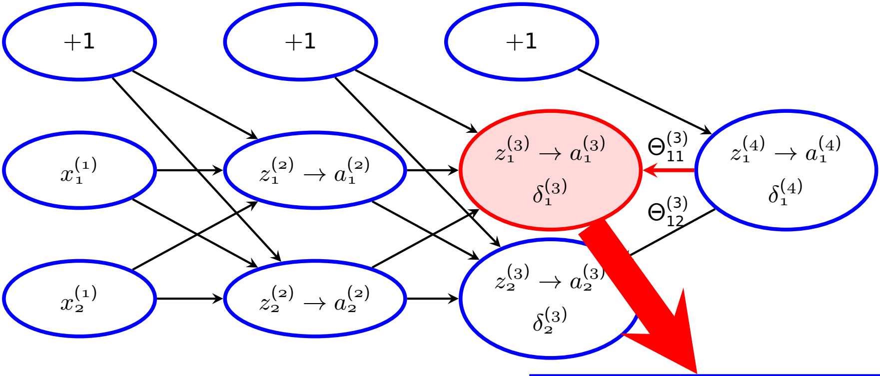
Exemplo

- Backpropagation



Exemplo

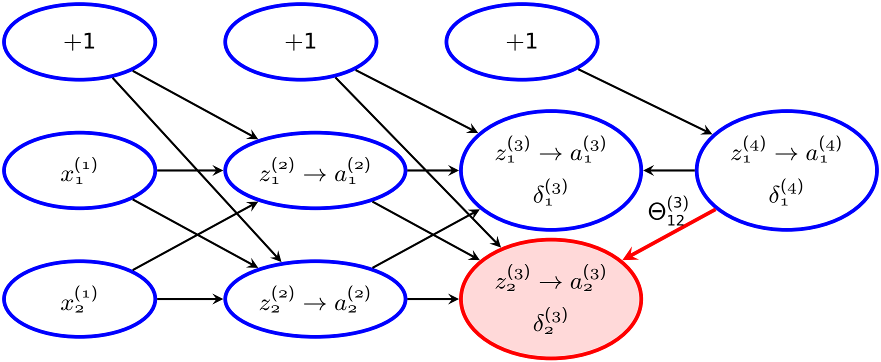
- Backpropagation



$$\begin{aligned} \delta_1^{(3)} &= \Theta_{11}^{(3)} * \delta_1^{(4)} * g'(z_1^{(3)}) \\ &= \Theta_{11}^{(3)} * \delta_1^{(4)} * [a_1^{(3)} * (1 - a_1^{(3)})]) \end{aligned}$$

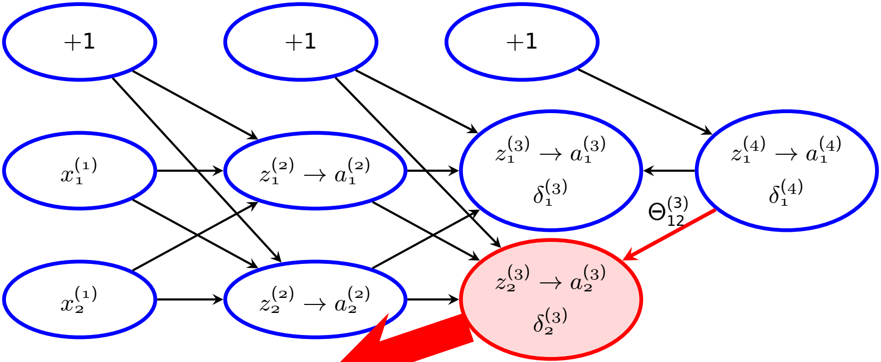
Exemplo

- Backpropagation



Exemplo

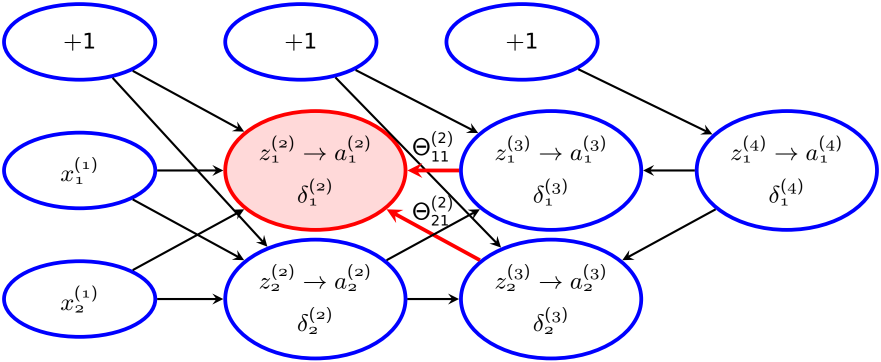
- Backpropagation



$$
\begin{aligned}
\delta_2^{(3)} &= \Theta_{12}^{(3)} * \delta_1^{(4)} * g'(z_2^{(3)}) \\
&= \Theta_{12}^{(3)} * \delta_1^{(4)} * [a_2^{(3)} * (1 - a_2^{(3)})])
\end{aligned}
$$

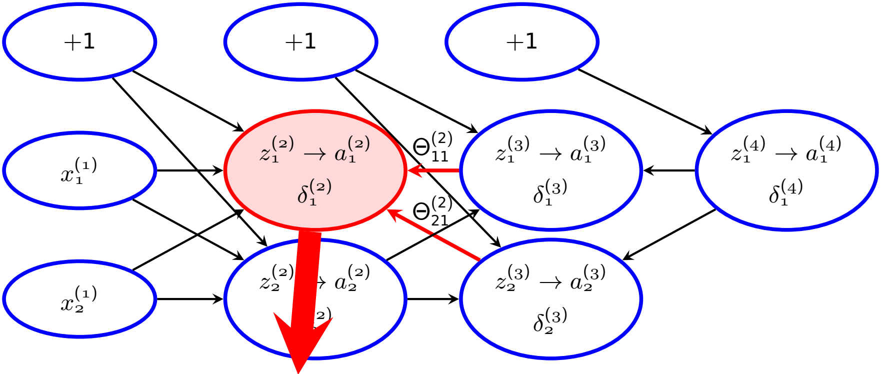
Exemplo

- Backpropagation



Exemplo

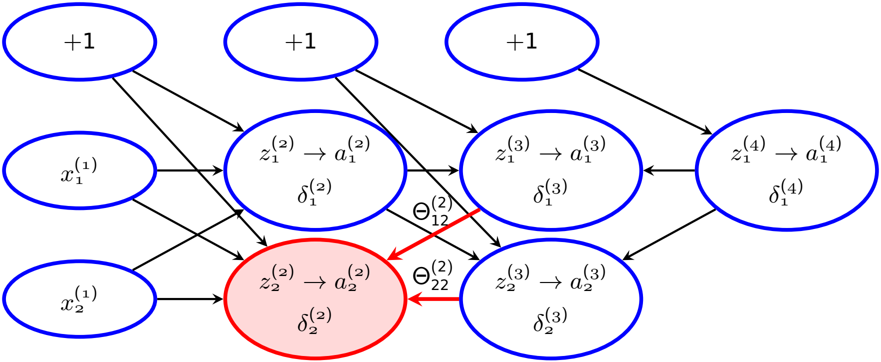
- Backpropagation



$$\begin{aligned}\delta_1^{(2)} &= \left[\Theta_{11}^{(2)} * \delta_1^{(3)} + \Theta_{21}^{(2)} * \delta_2^{(3)}\right] * g'(z_1^{(2)}) \\ &= \left[\Theta_{11}^{(2)} * \delta_1^{(3)} + \Theta_{21}^{(2)} * \delta_2^{(3)}\right] * \left[a_1^{(2)} * (1 - a_1^{(2)})\right]\end{aligned}$$

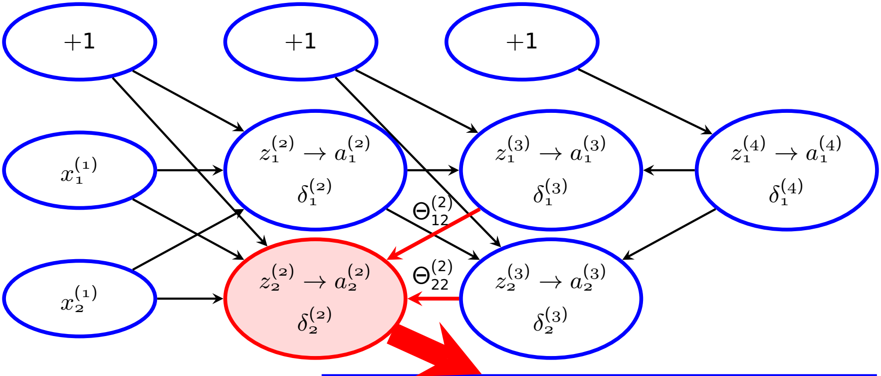
Exemplo

- Backpropagation



Exemplo

- Backpropagation



$$\begin{aligned}\delta_2^{(2)} &= \left[\Theta_{12}^{(2)} * \delta_1^{(3)} + \Theta_{22}^{(2)} * \delta_2^{(3)}\right] * g'\left(z_2^{(2)}\right) \\ &= \left[\Theta_{12}^{(2)} * \delta_1^{(3)} + \Theta_{22}^{(2)} * \delta_2^{(3)}\right] * \left[a_2^{(2)} * \left(1 - a_2^{(2)}\right)\right]\end{aligned}$$

Resumo

- **1. Definir a arquitetura da rede**
  - **No. de neurônios na entrada**: quantidade de atributos por amostra
  - **No. de neurônios na saída**: quantidade de classes
  - **No. de camadas intermediárias**: normalmente 1 camada intermediária
  - **No. de neurônios nas camadas intermediárias**: quantidade igual em todas as camadas intermediárias (no. de neurônios maior que camadas de E/S).
    - Obs: quanto mais neurônios e/ou camadas, maior será o esforço computacional.

Resumo

- **2. Treinamento da rede**
  - Inicializar pesos com **valores aleatórios próximos de zero**
  - Implementar **forward propagation** para obter $h_\Theta(x^{(i)})$ para cada $x^{(i)}$
  - Implementar **função custo** $J(\Theta)$
  - Implementar **backpropagation** para obter as derivadas parciais $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$

Resumo

2. Treinamento da rede

- Inicializar pesos com **valores aleatórios próximos de zero**
- Implementar **forward propagation** para obter $h_\Theta(x^{(i)})$ para cada $x^{(i)}$
- Implementar **função custo** $J(\Theta)$
- Implementar **backpropagation** para obter as derivadas parciais $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$
  - Para cada amostra da base i = 1 : m
  - Executar **forward propagation** e **backpropagation** usando $(x^{(i)}, y^{(i)})$ para obter os valores de ativação $a^{(l)}$ e os erros $\delta^{(l)}$ para $l = 2, ..., L$
  - Acumular os erros: $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
- Computar $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$

Resumo

- **2. Treinamento da rede**
  - Inicializar pesos com **valores aleatórios próximos de zero**
  - Implementar **forward propagation** para obter $h_\Theta(x^{(i)})$ para cada $x^{(i)}$
  - Implementar **função custo** $J(\Theta)$
  - Implementar **backpropagation** para obter as derivadas parciais $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$
    - Para cada amostra da base i = 1 : m
    - Executar **forward propagation** e **backpropagation** usando $(x^{(i)}, y^{(i)})$ para obter os valores de ativação $a^{(l)}$ e os erros $\delta^{(l)}$ para $l = 2, ..., L$
    - Acumular os erros: $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
  - Computar $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$
  - Empregar **método de otimização** para setar parâmetros Θ que minimizem $J(\Theta)$

Resumo

- Empregar **método de otimização** para setar parâmetros Θ que minimizem $J(\Theta)$
  - Deseja-se $\min_\theta J(\theta)$
  - **Método do gradiente**:

$$\text{Repita } \{ \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad \}$$

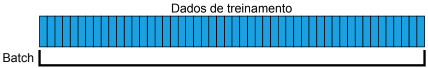
- Atualização simultânea para todo θ_j

Técnicas avançadas de otimização

- Existem funções de otimização que desenvolvem o papel do GD de forma muito mais eficiente
  - **SGD** (gradiente descendente estocástico)
    - Requer pouca memória
    - Converge rápido, já que atualiza os pesos com grande frequência
    - Boa capacidade de generalização
  - **Adam** (*Adaptive Moment Estimation*)
    - É rápido. Por isso, é bastante usado em *deep learning*
    - Adaptação da taxa de aprendizado
    - Menor capacidade de generalização que o SGD
  - **Adagrad** (*Adaptive Gradient*)
  - **RMSprop** (*Root Mean Square Propagation*)
  - **Adadelta** (*Adaptive Delta*)
- Comparação de técnicas de otimização: `https://github.com/Jaewan-Yun/optimizer-visualization`

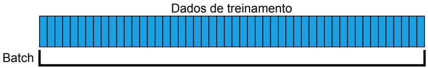
Gradiente descendente: *batch*, *mini-batch* e estocástico

- **Batch**: a cada época, todos os dados de treino são avaliados antes de calcular o gradiente.

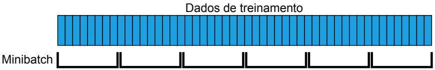


Gradiente descendente: *batch*, *mini-batch* e estocástico

- **Batch**: a cada época, todos os dados de treino são avaliados antes de calcular o gradiente.

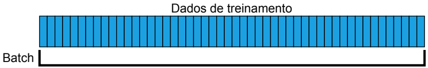


- **Mini Batch**: a cada época, os dados são divididos em subgrupos para atualização do gradiente

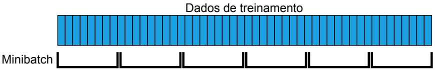


Gradiente descendente: *batch*, *mini-batch* e estocástico

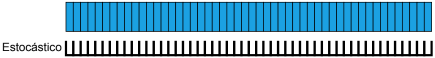
- **Batch**: a cada época, todos os dados de treino são avaliados antes de calcular o gradiente.



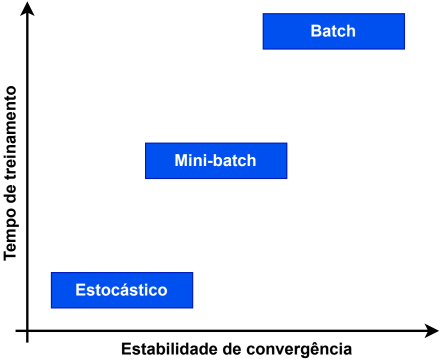
- **Mini Batch**: a cada época, os dados são divididos em subgrupos para atualização do gradiente



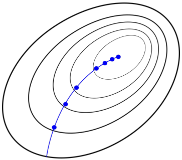
- **Estocástico**: a cada época, o gradiente é atualizado uma vez por registro



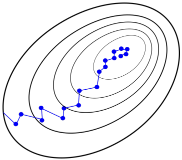
Comparativo de abordagens



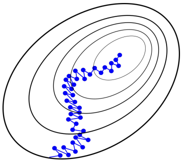
Comparativo de abordagens



(a) Batch



(b) Mini-batch



(c) Estocástico

Links e referências

- Aulas do Prof. Tiago Agostinho de Almeida (UFSCar, campus de Sorocaba)
- CARVALHO, André Carlos Ponce de Leon Ferreira et al. Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina. Disponível em: Minha Biblioteca, (2nd edição). Grupo GEN, 2021.
- Playgrounds
  - `https://playground.tensorflow.org`
  - `https://akarzazi.github.io/neural-network-playground/`
  - `https://playground.geosci.ai/`
  - `https://deeperplayground.org/`