REGRESSÃO LOGÍSTICA

Prof. Valmir Macário Filho

DC - UFRPE

CLASSIFICAÇÃO DE OBJETOS

Pessoa

vs

Fundo

- Fronteira de Classificação → Hiperplano em $\mathbb{R}^n$:
  - $w_o \; = \; \sum_{i=1}^{n} w_i \, x_i \; = 0$, por exemplo: uma linha em $\mathbb{R}^2$

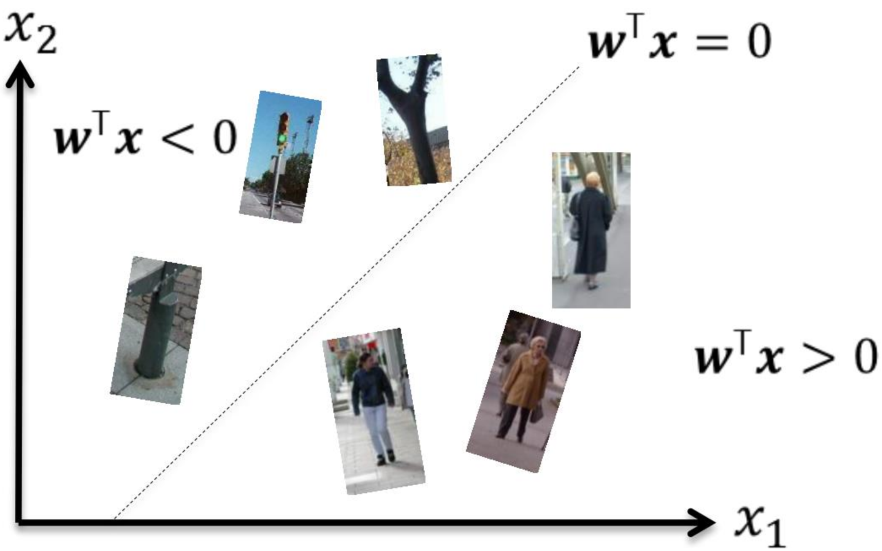
    um plano em $\mathbb{R}^3$

Vetor que define o hiperplano (modelo): $w = (w_0, w_1, \ldots, w_n)^T$

Ponto em $\mathbb{R}^{n+1}$ (descritor) $x = (1, \; x_1, \ldots, w_n)^T$

$w^T x = 0$ forma compacta

CLASSIFICAÇÃO BINÁRIA



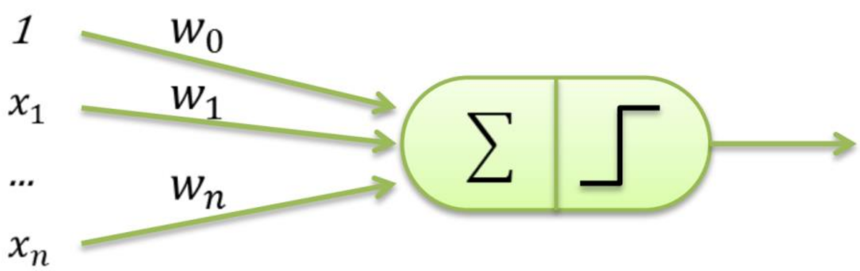
- $w^T x > 0$ 1 = Pedestre
- $w^T x < 0$ 0 = Fundo
- Para $w^T x = 0$, teríamos que decidir o valor

- Para 2 características ($n = 2$)
  - w= $(w_o, w_2, w_3)^T$
  - $x = (1,\ x_1, x_2, x_3)^T$

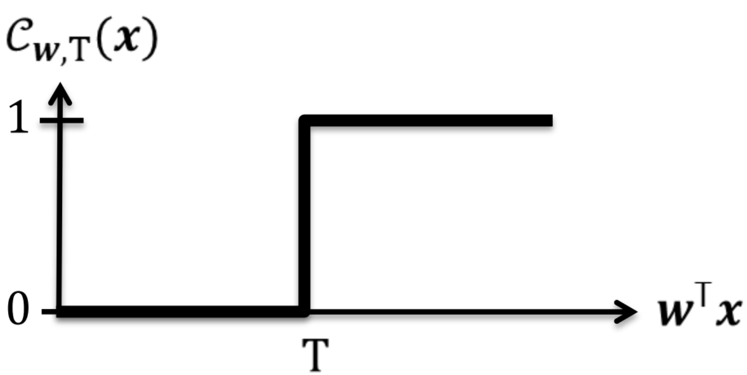
- Alternativamente
  - Classificar $(x; w\ , T)$ = Limiar $(w^T x, T)$

- Onde
  - Limiar $(y, T) = \begin{cases} 0\ se\ y < T \\ 1\ se\ y \geq T \end{cases}$

COM REDES NEURAIS



Classificar $(x; w\ , T) \equiv \mathcal{C}_{w,T}(x)$

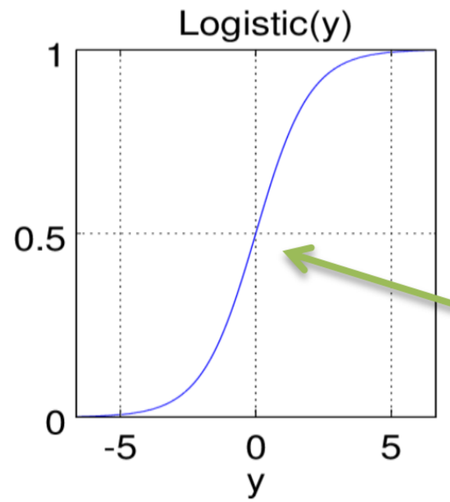


Pra cada valor $\boldsymbol{w}^T\boldsymbol{x}$, é dado como resposta um valor 0 ou 1, dado T

FUNÇÃO LOGÍSTICA

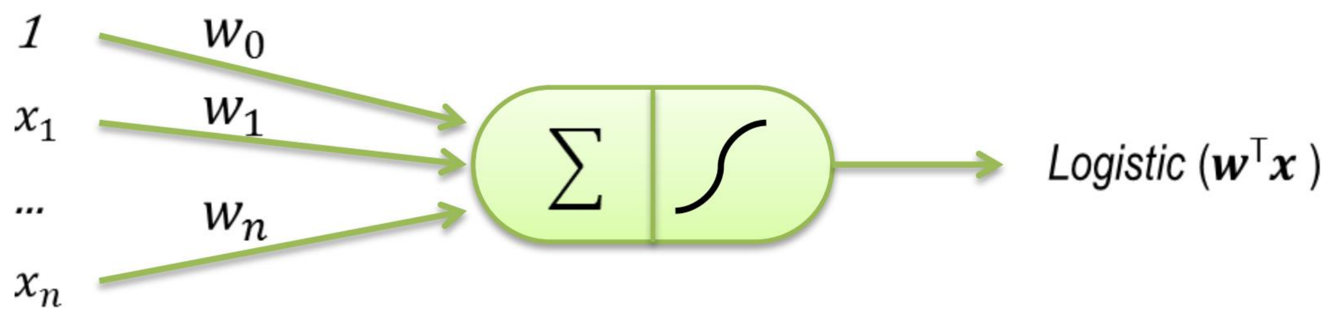
- Alternativa: converter os valores $w^T x$ para uma transição mais suave: por exemplo, utilizar a função sigmóid (também chamada de função logística): *Logistic* $(w^T x)$

$$Logistic\,(y) = \frac{1}{1+e^{-y}}$$



Transição mais suave

COM REDES NEURAIS



PROBABILIDADE CONDICIONAL

- Seja $Y \in \{0,1\}\ e\ X\ \in \mathbb{R}^{n+1}$ das variáveis aleatórias, y

  $P(Y|X)$ a probabilidade de Y condicionada a X.

- Se Y é usado para classificar, $Y = 1 \rightarrow$ Pedestre, $Y = 0 \rightarrow$ Fundo e X é o vetor de características então ao utilizar $P(Y|X)$:

- **Se** $P(Y = 1|X = x) < P(Y = 0|X = x)$, então $Y = 0$

  Senão $Y = 1$

FUNÇÃO LOGÍSTICA

- Utilizando uma forma paramétrica de $P(Y|X)$ baseada na função logística:

$$P(Y = 1|X = x; w) = Logistic\ (w^T x)$$

$$P(Y = 0|X = x; w) = 1\ - Logistic\ (w^T x)$$

FUNÇÃO LOGÍSTICA

$P(Y = 1|X = x) < P(Y = 0|X = x)$,
Então $Y = 0$,
Senão $Y = 1$

Se $1 < \frac{P(Y = 1|X = x, w)}{P(Y = 0|X = x, w)}$,
Então $Y = 0$,
Senão $Y = 1$

Vemos que $\frac{P(Y = 1|X = x, w)}{P(Y = 0|X = x, w)} = \exp(-w^T x)$, é possível simplificar

Ln() $1 < \exp(-w^T x)$

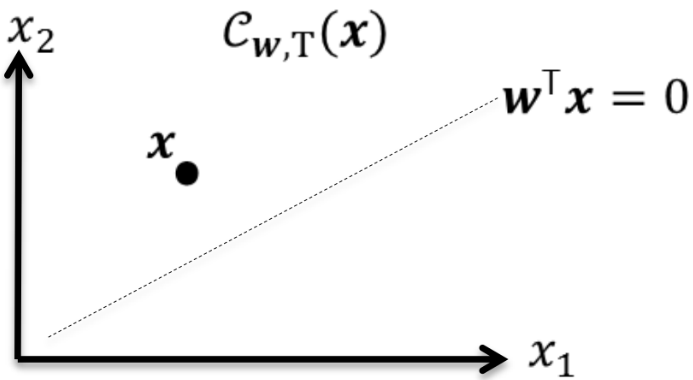
$0 < \exp(-w^T x)$

Se $w^T x < 0$
Então $Y = 0$,
Senão $Y = 1$

Isto equivale a usar T$= 0.5$ em relação ao valor $P(Y = 1|X = x)$

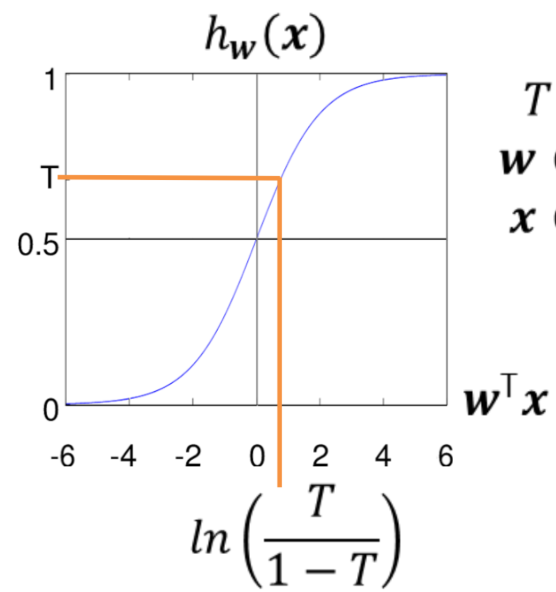
Usando T com um valor qualquer teríamos $w^T x < ln\left(\frac{T}{1-T}\right)$

APRENDIZAGEM



$$\mathcal{C}_{w,T}(x) = \begin{cases} 1 \; se \; h_w(x) < T \sim w^T x < \ln\left(\dfrac{T}{T-1}\right) \\ 0, senão \end{cases}$$

Onde $h_w(x) = Logistic(\, w^T x) = \frac{1}{1+e^{-w^T x}}$



$T \in [0,1]$

$\boldsymbol{w} \in \mathbb{R}^{n+1}$

$\boldsymbol{x} \in \mathbb{R}^{n+1}$

$x_0 = 1$

Que w usar?

APRENDIZAGEM

- w se aprende a partir de um conjunto de exemplos
- Conjunto de treinamento: $\mathcal{S} = \{(x'^1, y^1)\}, \ldots, (x'^M, y^M)\}$
  - (x'^j, y^j) j-ésimo exemplo de treinamento
  - $x'^j \in \mathbb{R}^n$ transforma $x^j \in \mathbb{R}^{n+1}$, onde $x^j = (1, x'^j)$
  - $y^j \in \{0,1\}$ classificação binária e supervisionada

$\mathcal{S} \rightarrow$ **Regressão Logística** $\rightarrow w$

Minimiza o "custo do erro" de classificar $(x^1, \ldots, x^M)$

OTIMIZAÇÃO

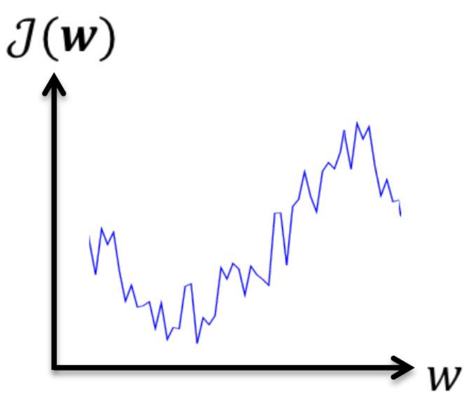
- O custo do erro pode ser definido como $\left(y^j - h_w, h(x^j)\right)^2$ utilizando o erro quadrático (L_2).

- O vetor w^* que minimiza o erro de classificação de $\mathcal{S}$ que pode-se obter resolvendo:

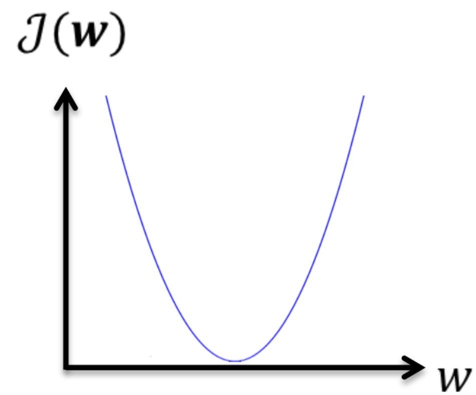
$$\boldsymbol{w}^* \longleftarrow \underset{\boldsymbol{w} \in \mathbb{R}^{n+1}}{argmin} \underbrace{\boxed{\frac{1}{M}\sum_{j=1}^{M}\left(y^j - h_{\boldsymbol{w}}(\boldsymbol{x}^j)\right)^2}}_{\mathcal{J}(\boldsymbol{w})}$$

OTIMIZAÇÃO

- Não existe uma solução fácil pra obter w^*
- Para tanto, se utiliza um método de otimização que a partir de valores iniciais de w e algumas equações de atualização, nos leve a solução que buscamos
- Porém, essa a função $\mathcal{J}(w)$ não é convexa a respeito de w. Portanto, muito difícil de otimizar (neste caso, minimizar)



não convexa

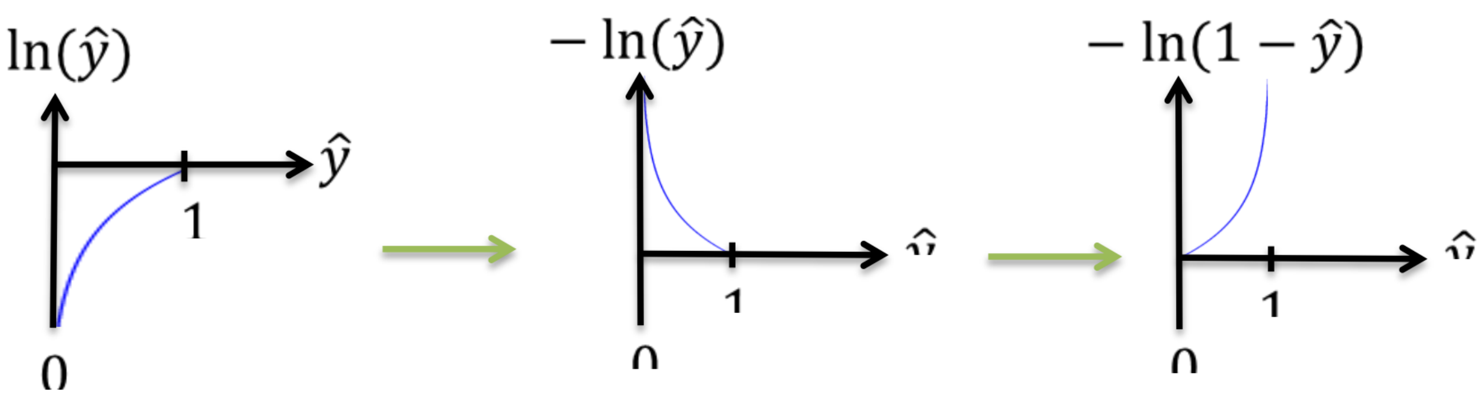


convexa

OTIMIZAÇÃO

- Por isso, se utiliza a seguinte alternativa:
- $\left(y^j - h_w, h(x^j)\right)^2 \; custo\left(h_w(x^j), y^j\right)$
- $custo(\hat{y}, y) = \begin{cases} -\ln(\hat{y}) & se\ y = 1 \\ -\ln(1-\hat{y}) & se\ y = 0 \end{cases}$

$\hat{y} \in [0,1]$



$\hat{y} \to 1 \Rightarrow custo \to 0$
$\hat{y} \to 0 \Rightarrow custo \to \infty$

$\hat{y} \to 0 \Rightarrow custo \to 0$
$\hat{y} \to 1 \Rightarrow custo \to \infty$

OTIMIZAÇÃO

- $custo(\hat{y}, y) = \begin{cases} -\ln(\hat{y}) & se\ y = 1 \\ -\ln(1-\hat{y}) & se\ y = 0 \end{cases}$
- $-(y\ ln(\hat{y}) + (1-y)\ln(1-\hat{y}))$

$\mathcal{J}(w)$ é convexa com respeito a w

- $\mathcal{J}(w) = -\frac{1}{M}\sum_{j=1}^{M} y^j\ ln\left(h_w\left(x^j\right)\right) + (1-y^j)\ln\left(1-h_w\left(x^j\right)\right)$

$$\boldsymbol{w}^* \longleftarrow \underset{\boldsymbol{w}\,\in\,\mathbb{R}^{n+1}}{argmin}\, \mathcal{J}(\boldsymbol{w})$$

REGRESSÃO LOGÍSTICA

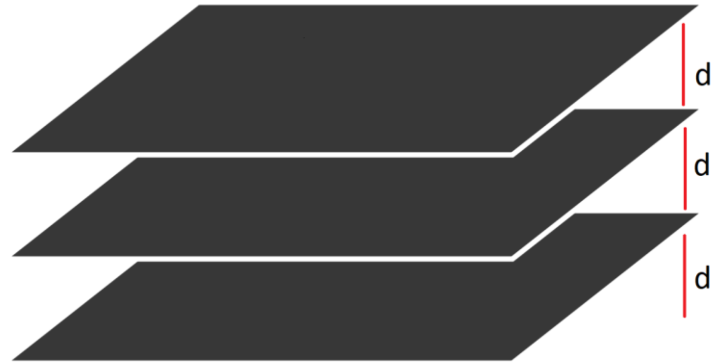
- (x^j, y^j) j-ésimo exemplo de treinamento
- $x^j \in \mathbb{R}^{n+1}$ descritor do exemplo j
- $y^j \in \{0,1\}$ rótulo do exemplo j
- $h_w(x^j)$ = Logistic $(w^T x^j)$

- $\mathcal{J}(w) = -\frac{1}{M}\sum_{j=1}^{M} y^j \, ln\left(h_w(x^j)\right) + (1 - y^j)\ln\left(1 - h_w(x^j)\right)$

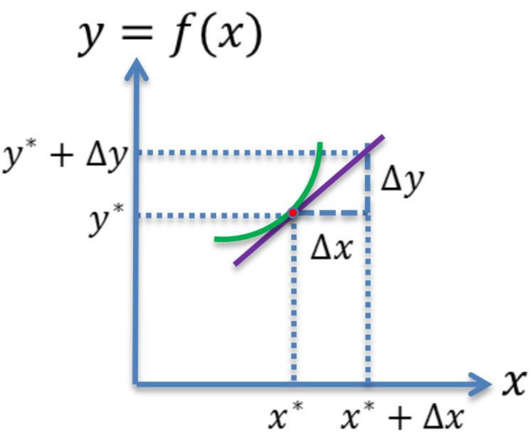
$$\boldsymbol{w}^* \longleftarrow \underset{\boldsymbol{w} \in \mathbb{R}^{n+1}}{argmin}\, \mathcal{J}(\boldsymbol{w})$$

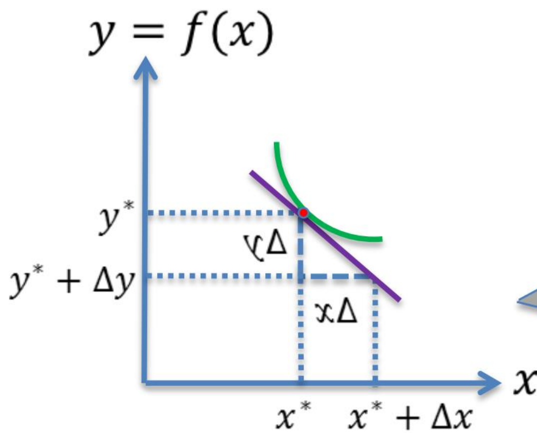
$\mathcal{J}(w)$ é convexa
com respeito a w

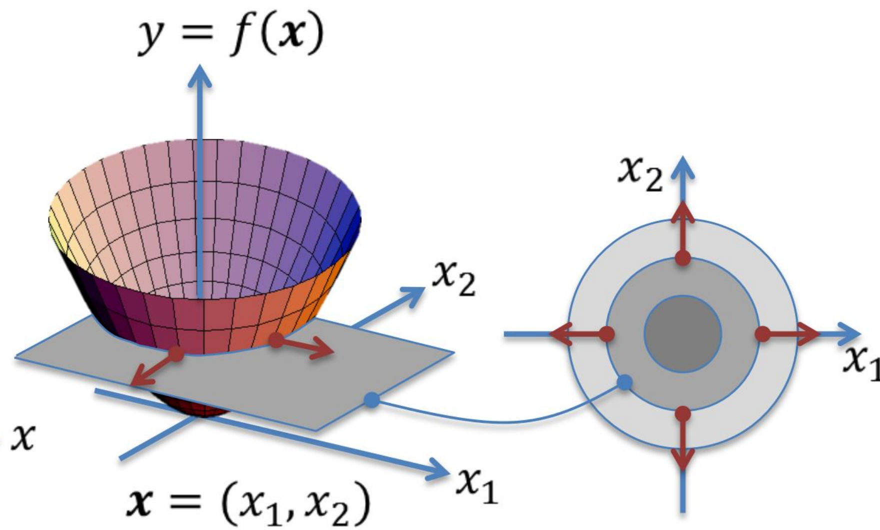
GRADIENTE DESCENDENTE



- Por ser convexa, pode-se utilizar algum algoritmo do tipo "descida de gradiente".



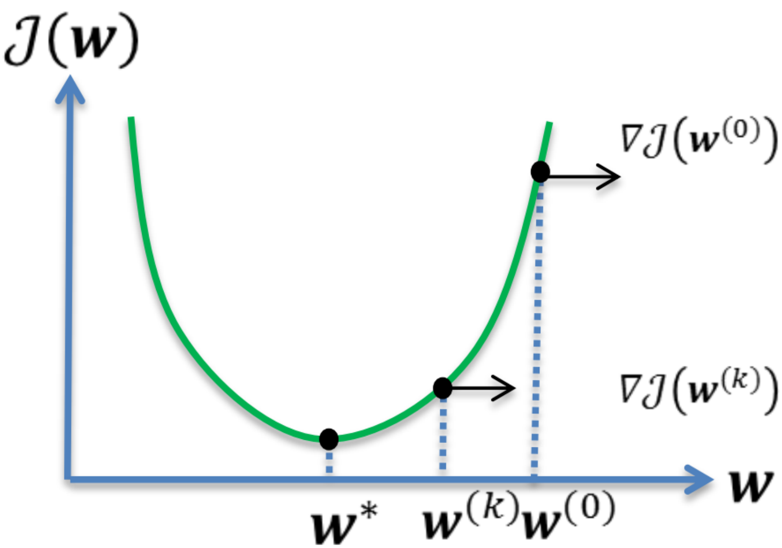




$$\frac{df(x)}{dx} \sim \frac{\Delta y}{\Delta x}$$

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2} \right)^{\mathsf{T}}$$

GRADIENTE DESCENDENTE



$$\nabla\mathcal{J}(\boldsymbol{w}) = \left(\frac{\partial\mathcal{J}(\boldsymbol{w})}{\partial w_0}, \dots, \frac{\partial\mathcal{J}(\boldsymbol{w})}{\partial w_n}\right)^{\mathsf{T}}$$

Repetir (em paralelo)

$$\begin{array}{ccc} {w_0}^{(k)} & \longleftarrow & {w_0}^{(k-1)} - \dfrac{\partial}{\partial w_0}\mathcal{J}(\boldsymbol{w}^{(k-1)}) \\ \vdots & & \vdots \qquad\qquad \vdots \\ {w_n}^{(k)} & \longleftarrow & {w_n}^{(k-1)} - \dfrac{\partial}{\partial w_n}\mathcal{J}(\boldsymbol{w}^{(k-1)}) \end{array}$$

Até convergir / ver $\mathcal{J}(w)$

GRADIENTE DESCENDENTE

Repetir (em paralelo)

$$w_0{}^{(k)} \longleftarrow w_0{}^{(k-1)} - \alpha \frac{\partial}{\partial w_0} \mathcal{J}(\boldsymbol{w}^{(k-1)})$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$w_n{}^{(k)} \longleftarrow w_n{}^{(k-1)} - \alpha \frac{\partial}{\partial w_n} \mathcal{J}(\boldsymbol{w}^{(k-1)})$$

Até convergir / ver $\mathcal{J}(w)$

- $\alpha \in \mathbb{R}^+$: velocidade de aprendizagem

GRADIENTE DESCENDENTE

- $\frac{\partial \mathcal{J}(w)}{\partial w_i} = \frac{\partial}{\partial w_i}\left[-\frac{1}{M}\sum_{j=1}^{M} y^j \; ln\left(h_w\left(x^j\right)\right) + \left(1-y^j\right)\ln\left(1-h_w\left(x^j\right)\right)\right]$
- $\frac{1}{M}\sum_{j=1}^{M}\left(h_w\left(x^j\right)-y^j\right)x_i^j$

- Repetir (em paralelo)

$$
\begin{aligned}
w_0{}^{(k)} &\leftarrow w_0{}^{(k-1)} - \alpha\frac{1}{M}\sum_{j=1}^{M}\left(h_{\boldsymbol{w}}\left(\boldsymbol{x}^j\right)-y^j\right)x_0{}^j \\
\vdots \qquad & \qquad \vdots \qquad\qquad\qquad\qquad \vdots \\
w_n{}^{(k)} &\leftarrow w_n{}^{(k-1)} - \alpha\frac{1}{M}\sum_{j=1}^{M}\left(h_{\boldsymbol{w}}\left(\boldsymbol{x}^j\right)-y^j\right)x_n{}^j
\end{aligned}
$$

Até convergir / ver $\mathcal{J}(w)$

REVISÃO DE CÁLCULO

- Regra da cadeia: Se h(x) = g(f(x)) então, pra qualquer mudança em x, f(x) muda por $\frac{\partial f}{\partial x}$, e pra cada mudança em f, g muda por $\frac{\partial g}{\partial f}$, então a mudança total em h(x) = g(f(x) é:

  - $\frac{\partial h}{\partial x} = \frac{\partial g}{\partial f}\frac{\partial f}{\partial x}$

ALGUMAS DERIVADAS

Função

Derivadas:

$$y = \frac{1}{2}\sum_{k \in K}(c_k - x_k)^2$$

$$\frac{\partial y}{\partial x_i} = -(c_i - x_i)$$

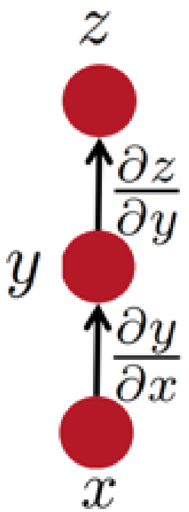
$$y = \sum w_i . x_i$$

$$\frac{\partial y}{\partial w_i} = x_i$$

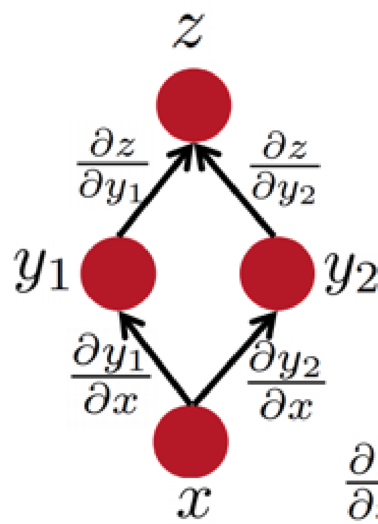
$$y = \frac{1}{1+\exp\{-(x-T)\}}$$

$$\frac{\partial y}{\partial x} = \frac{\exp\{-(x-T)\}}{(1+\exp\{-(x-T)\})^2} = y(1-y)$$

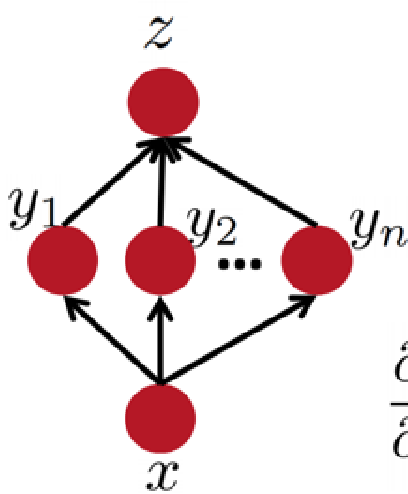
EXEMPLOS EM REDES NEURAIS



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$



$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x}$$

BACKPROPAGATION

- O primeiro passo é a definição do custo do erro na última camada:
- $L(w) = L_{MSE} = \frac{1}{M}\sum_{j=1}^{M}\left(h_w\left(x^j\right) - y^j\right)^2$
- Erro: $y^j - h_w\left(x^j\right) = y - z$

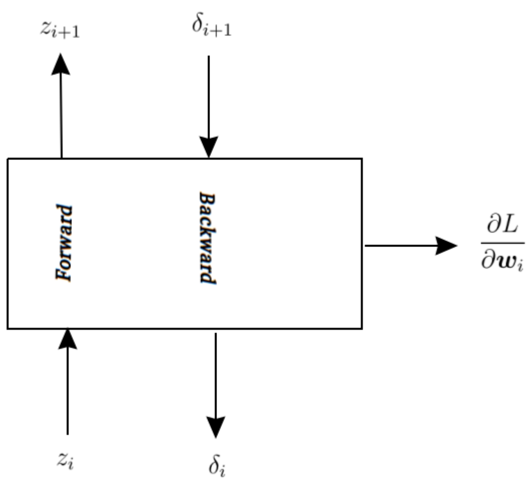
- Então definimos a derivada da camada em função de suas entradas
- $\frac{\partial f(z)}{\partial z} = w$
- $\frac{\partial f(z)}{\partial z} = -\frac{2}{M}\sum(y - z)$

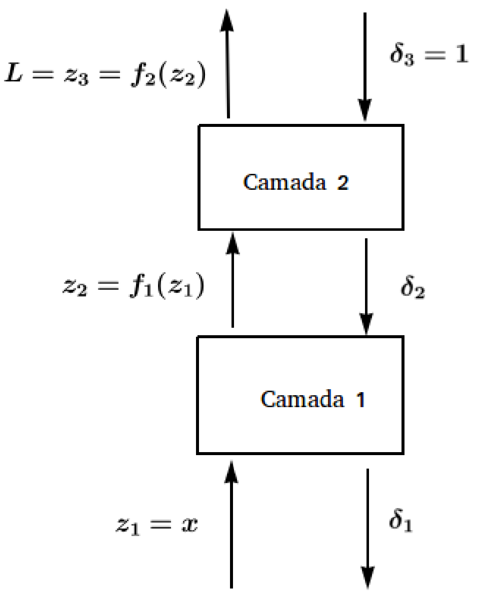
BACKPROPAGATION

- Depois, definimos a derivada da camada em função com respeito aos parâmetros da camada
  - $\frac{\partial f(z)}{\partial w} = z$
- Intuitivamente, queremos saber como mudanças em w afetam a função de custo L(w)
- **Se definirmos camadas com esses dois ou três mecanismos será possível saber a derivada da função custo com respeito a qualquer parâmetro**

MODULARIDADE

- No passo *forward,* uma camada i recebe como entrada z_i e produz como saída z_{i+1}
- Também, a derivada da função de custo com relação a entrada i como δ_i
- Assim, podemos derivar o algoritmo de backpropagation em termos de z e δ.





BACKWARD

- $\delta_i = \frac{\partial L}{\partial z_i} = \frac{\partial L}{\partial z_{i+1}} * \frac{\partial z_{i+1}}{\partial z_i} = \delta_{i+1} \frac{\partial z_{i+1}}{\partial z_i}$

- Vemos que δ_i é definido pelo próximo δ_{i+1}, então pode-se definir todos δ_s, recursivamente.

- Como na última camada a derivada da entrada é o próprio custo, o último δ_i pode ser 1.

BACKWARD

- Agora, derivamos em relação aos parâmetros de entrada:

- $\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z_{i+1}} * \frac{\partial z_{i+1}}{\partial w_i} = \delta_{i+1} \frac{\partial z_{i+1}}{\partial w_i}$

- Então, a derivada da função de custo pode ser definida em termos de δ_{i+1} e da derivada da saída da camada em relação aos pesos

- O poder desse algoritmo é ele não nos diz como deve ser essa arquitetura, podemos organizar da maneira que quisermos, desde que a função de ativação seja diferenciável.

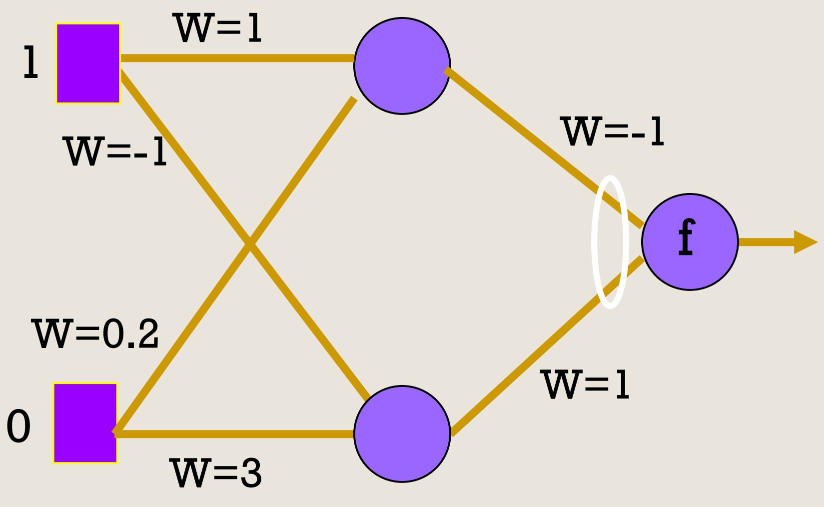
ALGORITMO DE APRENDIZADO

- Passo foward: $f(z_i, w) = \ wz_i = z_{i+1}$
- Passo backward última camada: $\delta_i = \delta_{i+1} \frac{\partial f(z_i)}{\partial z_i} = \delta_{i+1} w$
- Passo backward outras camadas: $\frac{\partial L}{\partial w_i} = \delta_{i+1} \frac{\partial z_{i+1}}{\partial w_i} = \delta_{i+1} z_i$

- Se utilizarmos o erro quadrático médio:
- $f(z_i, w) = \ \frac{1}{M} \sum (y - z_i)^2 = z_{i+1}$
- Passo backward: $\delta_i = \delta_{i+1} \frac{\partial f(z_i)}{\partial z_i} = -\delta_{i+1} \frac{1}{M} \sum (y - z_i)$
  - Como vamos multiplicar o último δ pela taxa de aprendizado, podemos eliminar $\frac{1}{M}$ da equação.
  - $a\delta_{i+1} \sum (z_i - y)$
- $\frac{\partial L}{\partial w_i} = \delta_{i+1} \frac{\partial z_{i+1}}{\partial w_i} = \delta_{i+1} z_i = \delta_{i+1} \sum (\delta_i w)$

REDE NEURAL MULTICAMADA

- Observem o exemplo da rede neural abaixo aprendendo com o valor (x_1=1, x_2=0;y=1), que significa que a entrada é 1,0 e a saída esperada é y=1



$$net = \sum w_i \, x_i$$

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

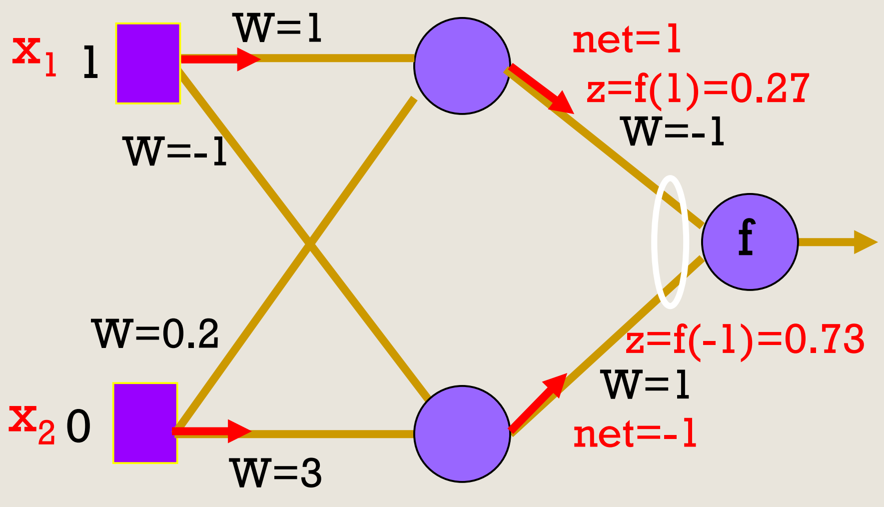
FORWARD

Fase Forward: Calcula a saída de cada neurônio da camada da camada intermediária.

net $= \Sigma\, w_i\, x_i$

net=1*1+0*0,2=1

net=1*(-1)+0*3=-1



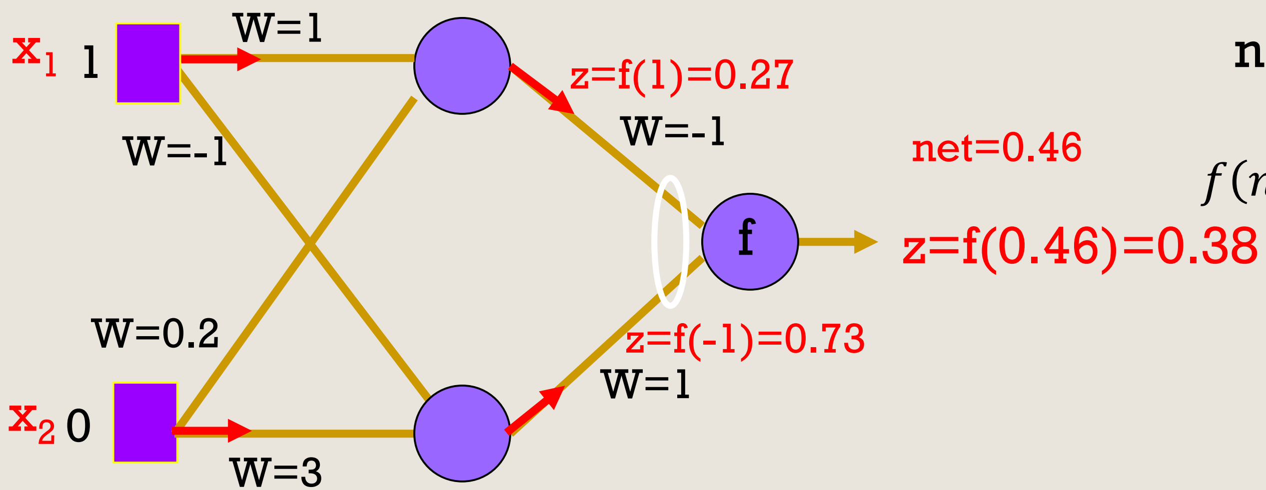
net $= \Sigma\, w_i\, x_i$

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

FORWARD

Fase Forward: Calcula a saída de cada neurônio da camada de saída.

net=0.27*(-1)+0.73*1=0.46



$$\text{net} = \sum w_i x_i$$

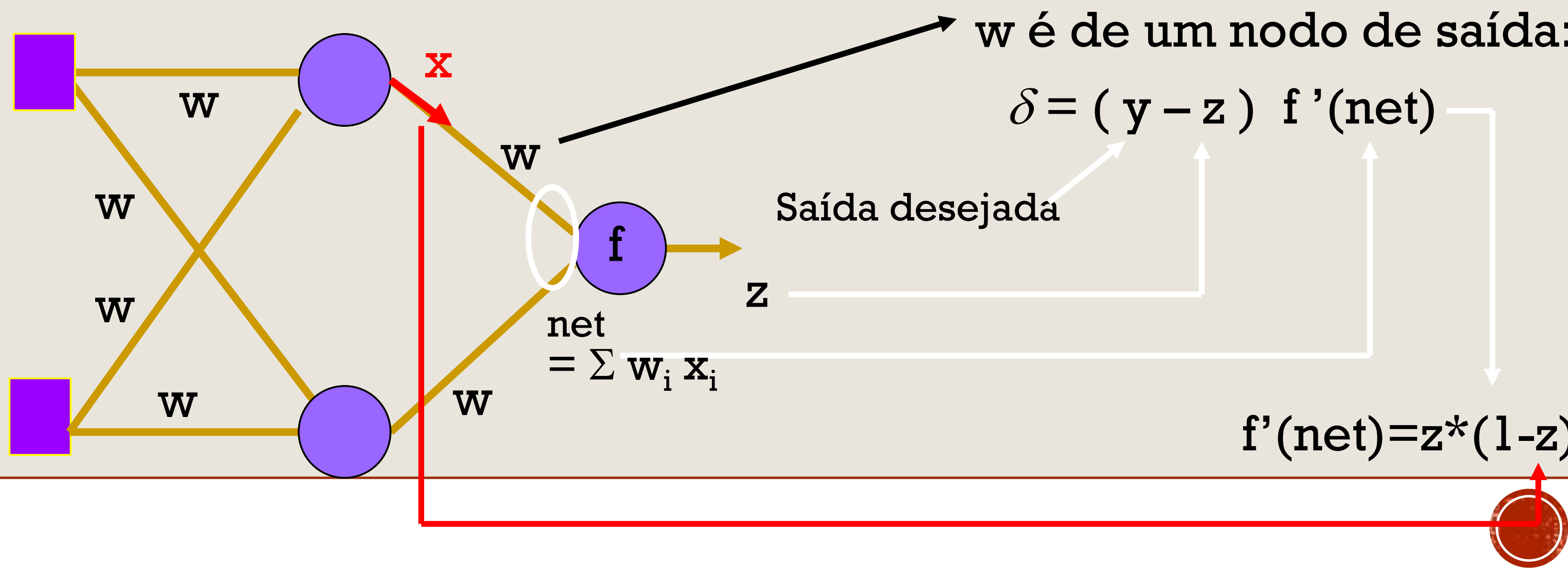
$$f(net) = \frac{1}{1 + e^{-net}}$$

BACKPROPAGATION

Fase Backward: Ajusta os pesos da rede a partir da camada de saída.

$$\Delta w_{ji} = \eta\, \delta_j(t)\, x_i(t)$$

$$onde \quad \delta_j(t) = \begin{cases} (d_j - y_j)\ f'(net_j), & se\ \ for\ nodo\ de\ saída \\ (\sum_l \delta_l\ w_{l\,j})\ f'(net_j), & caso\ contrário \end{cases}$$



BACKPROPAGATION

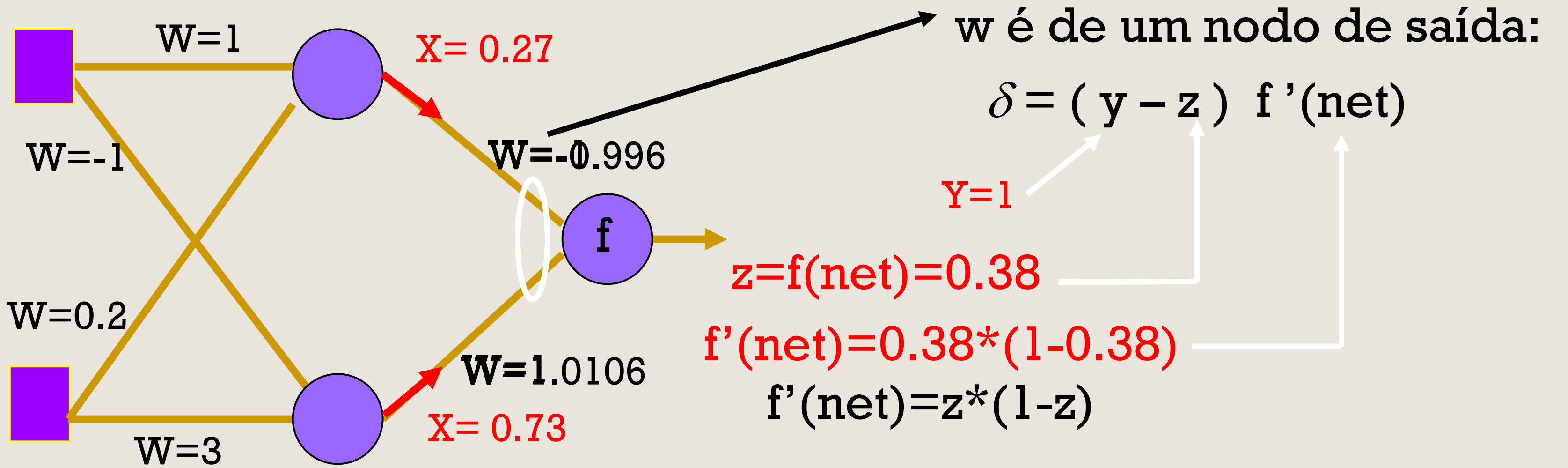
Fase Backward: Ajusta os pesos da rede a partir da camada de saída. Use (η=0.1)

$$\Delta w_{21} = 0.1 * 0.146 * 0.27 = 0.004$$
$$\Delta w_{21} = -1 + 0.004 = -0.996$$

$$\delta_j = (1 - 0.38) * 0.38(1 - 0.38) = 0.146$$

$$\Delta w_{22} = 0.1 * 0.146 * 0.73 = 0.0106$$
$$\Delta w_{22} = 1 + 0.017 = 1.0106$$

$$\Delta w_{ji} = \eta \ \delta_j(t) \ x_i(t)$$
$$onde \quad \delta_j(t) = (y_j - z_j) \ f'(net_j), \quad se \ for \ nodo \ de \ saída$$



BACKPROPAGATION

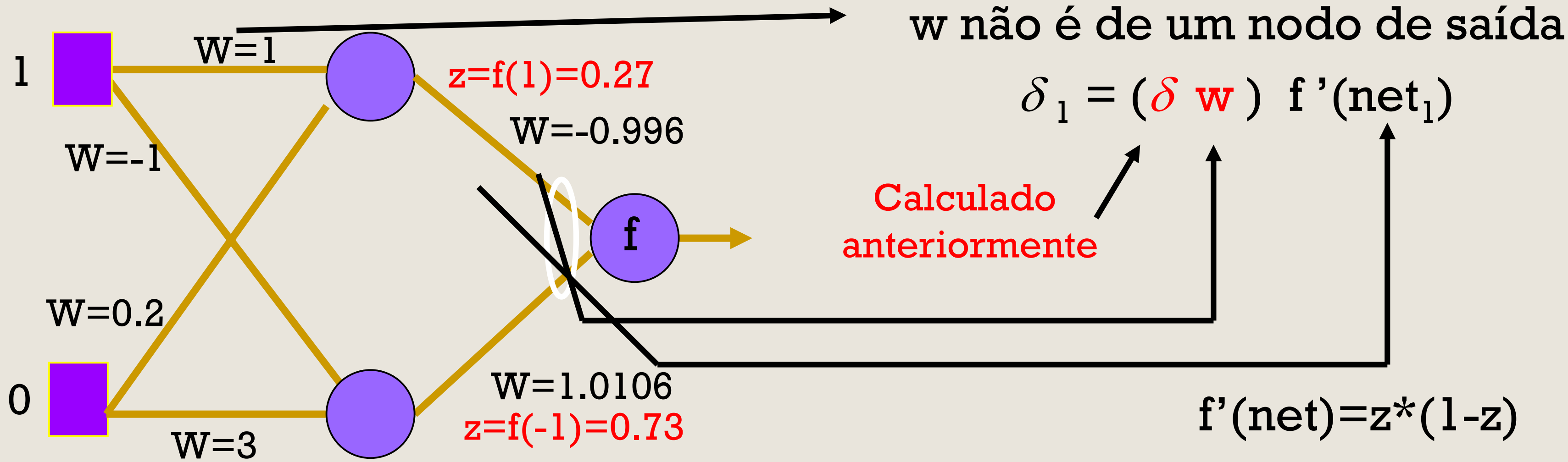
Fase Backward: Ajusta os pesos da camada intermediária.

$$\Delta w_{11} = 0.1 * (0.03) * 1 = 0.003$$
$$\Delta w_{11} = 1 + (0.003) = 1.003$$

$$\delta_1 = (0.146 * (-0.996)) * 0.27(1 - 0.27)$$
$$= -0{,}03$$

$$\Delta w_{12} = 0.1 * 0.03 * 0 = 0$$
$$\Delta w_{12} = 0.2 + 0 = 0.2$$

$$\Delta w_{ji} = \eta\ \delta_j(t)\ x_i(t)$$
$onde \quad \delta_j(t) = \left(\sum_l \delta_l\ w_{lj}\right)\ f'\left(net_j\right)$ para nós intermediários



BACKPROPAGATION

Fase Backward: Ajusta os pesos da camada intermediária.

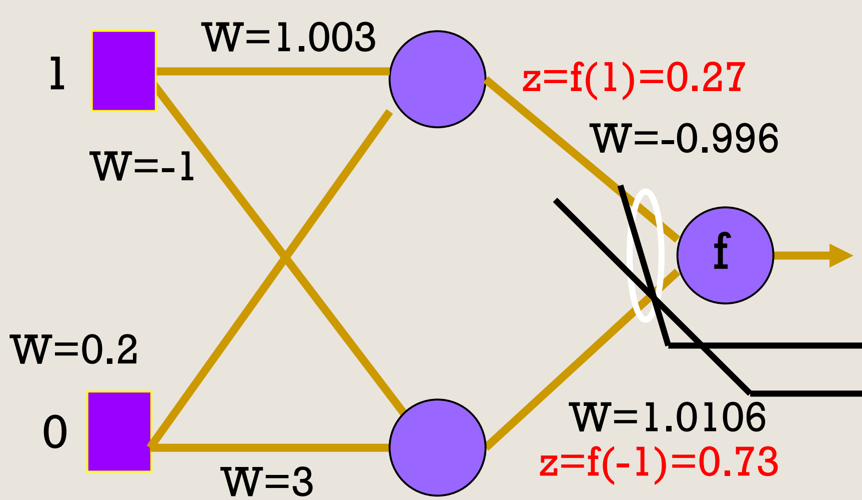
$$\Delta w_{11} = 0.1 * (0.03) * 1 = 0.003$$
$$\Delta w_{11} = \ 1 + (0.003) = 1.003$$

$$\delta_1 = (0.146 * (-0.996)) * 0.27(1 - 0.27)$$
$$= -0{,}03$$

$$\Delta w_{12} = 0.1 * 0.03 * 1 = 0$$
$$\Delta w_{12} = 0.2 + 0 = 0.2$$

$\Delta w_{ji} = \eta \ \delta_j(t) \ x_i(t)$

onde $\delta_j(t) = \left(\sum_l \delta_l \ w_{lj}\right) \ f'\left(net_j\right)$ para nós intermediários



w não é de um nodo de saída:

$\delta_1 = (\delta \ \mathbf{w}) \ \ f'(net_1)$

Calculado anteriormente

f'(net)=z*(1-z)

BACKPROPAGATION

Fase Backward: Ajusta os pesos da camada intermediária.

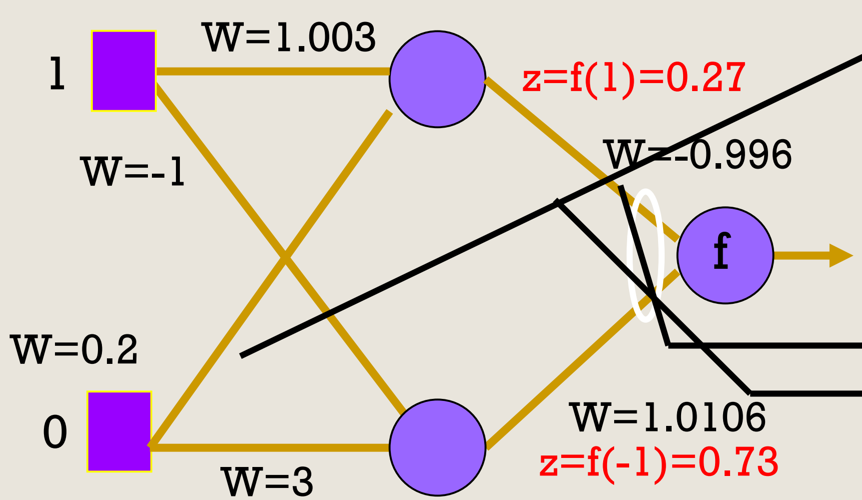
$$\delta_2 = (0.146 * (1.0106)) * 0.73(1 - 0.73) = 0.029$$

$$\Delta w_{13} = 0.1 * 0.029 * 1 = 0.0029$$
$$\Delta w_{13} = -1 + 0.0029 \ = -0.997$$

$$\Delta w_{14} = 0.1 * -0.442 * 0 = 0$$
$$\Delta w_{14} = 3 + 0 = 3$$

$\Delta w_{ji} = \eta \ \delta_j(t) \ x_i(t)$

$onde \quad \delta_j(t) = \left(\sum_l \delta_l \ w_{lj}\right) \ f'\left(net_j\right)$ para nós intermediários



w não é de um nodo de saída:

$\delta_1 = (\delta \ w) \ f'(net_1)$

Calculado anteriormente

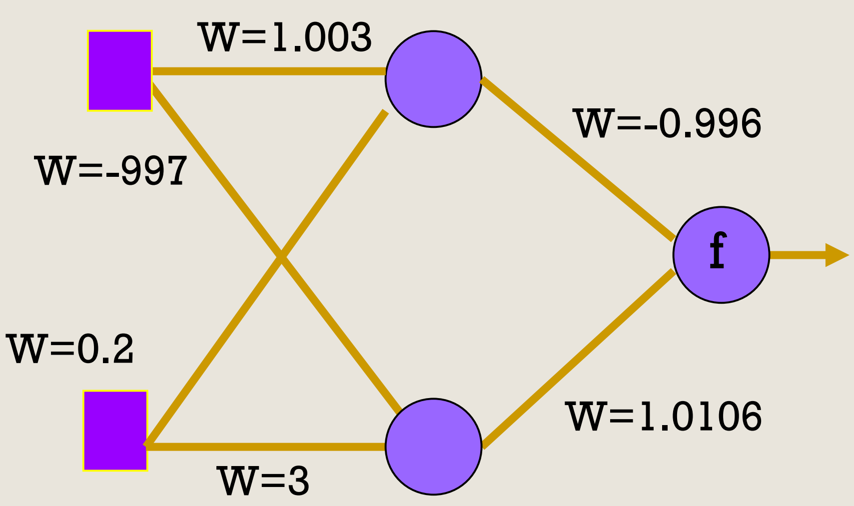
f'(net)=z*(1-z)

BACKPROPAGATION

Rede após fase backward

$\Delta w_{ji} = \eta\ \delta_j(t)\ x_i(t)$

onde $\delta_j(t) = \left(\sum_l \delta_l\ w_{lj}\right)\ f'\left(net_j\right)$ para nós intermediários



w não é de um nodo de saída:

$\delta_1 = (\delta\ \mathbf{w})\ \ f'(net_1)$