AULA 3
GRADIENTE DESCENDENTE – RNA RASA

1. Objetivos

- Apresentar o algoritmo básico de treinamento supervisionado das RNAs ⇒ **Método do Gradiente Descendente**.
- Apresentar a forma de implementação eficiente do método do Gradiente Descendente ⇒ **Retro-Propagação**.
- Aplicação do método do Gradiente Descendente em uma RNA de um único neurônio.
- Aplicação do método do Gradiente Descendente em RNAs rasas.

2. Descrição do problema

- Para uma RNA ser capaz de resolver algum problema, como por exemplo, classificação ou ajuste de uma função, ela precisa ser treinada.
- O **aprendizado supervisionado** tem as seguintes características:
  - É realizado por meio da apresentação de um conjunto de dados (entradas e saídas) à RNA;
  - Exige um conjunto de dados rotulados e classificados;
  - Durante o aprendizado a saída gerada é comparada com a saída desejada ⇒ a diferença entre a saída desejada e a calculada é usada para o treinamento.
- Os algoritmos de treinamento das RNAs consistem de métodos de otimização no qual definida uma **função de custo**, o algoritmo calcula os parâmetros da rede (matrizes de ganhos $\mathbf{W}^{[l]}$ e vetores de viés $\mathbf{b}^{[l]}$, para $l = 1,...,L$) que minimiza essa função.
- O treinamento exige definir uma **função de custo** baseada em uma **função de erro** ⇒ existem diversas funções de erro que veremos ao longo da disciplina.
- Uma função de erro fornece uma medida da diferença entre a saída desejada e a saída gerada pela RNA.
- O objetivo do treinamento no aprendizado supervisionado é minimizar a função de custo.

3. Conjunto de dados de treinamento

- No treinamento de uma RNA são usados muito exemplos.

- **Exemplos de treinamento são divididos em pelo menos dois conjuntos:**
  - **Um conjunto usado para treinar a RNA;**
  - **Um conjunto usado para testar a RNA após o treinamento.**

- Conjunto de exemplos de treinamento:
  - Número de exemplos de treinamento $\Rightarrow m$;
  - Cada exemplo $\Rightarrow (\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, $\mathbf{x}^{(i)} \in R^{nx}$ e $\mathbf{y}^{(i)} \in R^{ny}$;
  - Os m exemplos $\Rightarrow \{(\mathbf{x}^{(1)}, \boldsymbol{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$.

4. Função de custo

- Objetivo do treinamento de uma RNA $\Rightarrow$ calcular os pesos da RNA de forma a minimizar uma função de custo.
- A função de custo também é chamada de critério de desempenho.
- A função de custo é baseada em uma função de erro.
- Existem inúmeras funções de erro $\Rightarrow$ as mais usadas são:
  - Erro quadrático;
  - Função logística;
  - Função softmax.
- A escola da função de erro utilizada $\Rightarrow$ depende do tipo de problema que se deseja resolver.
- É importante ressaltar que a função de erro tem que ter derivada contínua para poder ser usada como função de custo.

Função de erro quadrático

- Problemas de ajuste de funções $\Rightarrow$ função de erro mais utilizada é o erro quadrático médio.
- Nos problemas de ajuste de funções $\Rightarrow$ um elemento do conjunto de treinamento consiste de um vetor $\mathbf{x}^{(i)}$, de dimensão $(n_x, 1)$, e a sua saída correspondente $\mathbf{y}^{(i)}$, que normalmente é um vetor de dimensão $(n_y, 1)$.
- Conjunto de exemplos de treinamento:
  - Cada exemplo $\Rightarrow (\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, $\mathbf{x}^{(i)} \in R^{nx}$ e $\boldsymbol{y}^{(i)} \in R^{ny}$;

- Os *m* exemplos ⇒ $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$.

➢ **Função de erro quadrático para cada exemplo de treinamento:**

$$E(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}) = \sum_{j=1}^{n_y} (\hat{y}_j^{(i)} - y_j^{(i)})^2 = \left\| \hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)} \right\|_2^2 \tag{4}$$

onde $\|\mathbf{v}\|_2$ representa a norma 2 do vetor $\mathbf{v}$, ou seja:

$$\|\mathbf{v}\|_2 = \sqrt{v_1^2 + v_2^2 + \ldots + v_{n_y}^2} \tag{5}$$

➢ **Função de custo para os *m* exemplos de treinamento:**

$$J(\mathbf{W}, \mathbf{B}) = \frac{1}{m} \sum_{i=1}^{m} E(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{j=1}^{n_y} (\hat{y}_j^{(i)} - y_j^{(i)})^2 \right) = \frac{1}{m} \sum_{i=1}^{m} \left(\left\| \hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)} \right\|_2^2 \right) \tag{6}$$

onde $\mathbf{W}$ e $\mathbf{B}$ representam compactamente todos os parâmetros da RNA.

- **Observe novamente que a função de custo é função dos parâmetros da RNA e não das saídas calculadas pela RNA.**

Função de erro logística

➢ Problemas de classificação binária ⇒ a função de erro mais utilizada é a função logística.

➢ Nos problemas de classificação binária ⇒ um elemento do conjunto de treinamento consiste de um vetor $\mathbf{x}^{(i)}$, de dimensão $(n_x, 1)$, e a sua saída escalar correspondente $y^{(i)}$, que classifica positivamente ou negativamente o vetor de entrada, ou seja:

- $y^{(i)} = 0$ ⇒ entrada $\mathbf{x}^{(i)}$ classificada negativamente;
- $y^{(i)} = 1$ ⇒ entrada $\mathbf{x}^{(i)}$ classificada positivamente.

➢ Conjunto de exemplos de treinamento:

- Cada exemplo ⇒ $(\mathbf{x}^{(i)}, y^{(i)})$, $\mathbf{x}^{(i)} \in \mathrm{R}^{nx}$ e $y^{(i)} \in \{0, 1\}$;
- Os *m* exemplos ⇒ $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$.

➢ **Função logística para cada exemplo de treinamento:**

$$L(\hat{y}^{(i)}, y^{(i)}) = -y^{(i)} \log \hat{y}^{(i)} - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \tag{7}$$

Observa-se que a saída real é sempre, $y^{(i)} = 0$ ou 1 ⇒ portanto, existem somente duas possibilidades para a função de erro logística:

- Se $y^{(i)} = 1 \Rightarrow L(\hat{y}^{(i)}, y^{(i)}) = -\log \hat{y}^{(i)} \Rightarrow$ para minimização queremos $\log \hat{y}^{(i)}$ grande, ou seja, $\hat{y}^{(i)}$ grande;

- Se $y^{(i)} = 0 \Rightarrow L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)}) \Rightarrow$ para minimização queremos $\log(1 - \hat{y}^{(i)})$ grande, ou seja, $\hat{y}^{(i)}$ pequeno.

Na Figura 1 é apresentado o gráfico da função de erro logística com as duas possibilidades para a saída real.

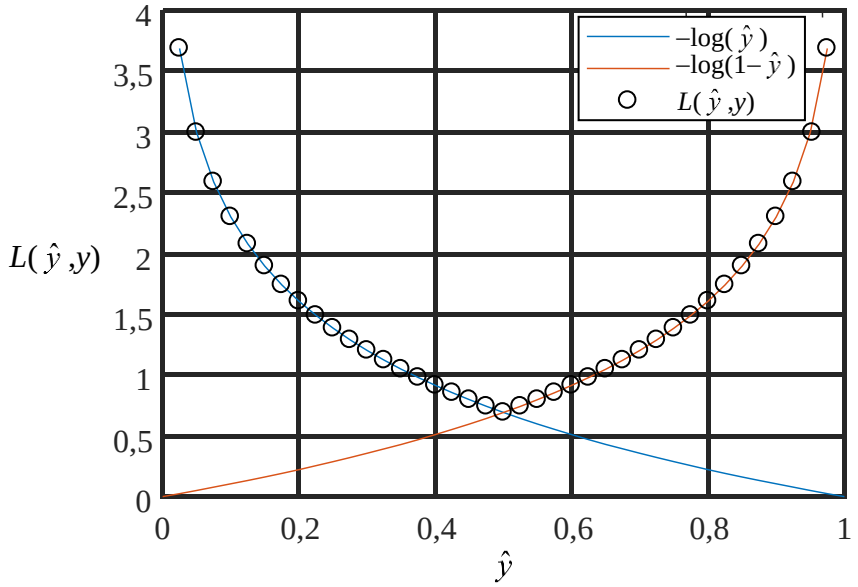


Figura 1. Função de erro logística em função da saída prevista $\hat{y}$.

➢ **Função de custo para os *m* exemplos de treinamento:**

$$J(\mathbf{W},\mathbf{B}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})\right] \quad (8)$$

onde **W** e **B** representam compactamente todos os parâmetros da RNA.

- **Observe novamente que a função de custo é função dos parâmetros da RNA e não das saídas calculadas pela RNA.**

➢ Veremos outras funções de erro e de custo ao longo da disciplina.

4. Método do Gradiente Descendente

➢ Objetivo de treinar uma RNA ⇒ calcular todos os parâmetros da RNA (**W** e **B**) de forma a minimizar a função de custo J(**W**, **B**).

- Dada uma função diferenciável, teoricamente é possível achar o seu valor mínimo determinando o ponto onde a derivada da função é igual a zero.

- Na maior parte dos casos podem existir vários pontos onde a derivada da função é igual a zero $\Rightarrow$ nesses casos temos que achar entre todos esses pontos para qual a função tem o menor valor.

- A aplicação desse processo de obtenção do valor mínimo da função de custo para uma RNA significa achar a combinação de todos os parâmetros que geram o menor valor para a função de custo.

- Teoricamente isso pode ser feito resolvendo um sistema de equações, que é definido igualando o gradiente da função de custo em relação a todos os parâmetros da RNA, contudo, isso somente é possível se a RNA possuir poucos parâmetros $\Rightarrow$ uma RNA profunda pode ter alguns milhões de parâmetros fazendo com que esse processo de solução seja impraticável.

- No caso das RNAs o processo de otimização da função de custo, para obtenção dos seus parâmetros é realizado em etapas:

  1. Execução da RNA para todos os exemplos do conjunto de dados de treinamento, de forma que dadas as entradas calculam-se as saídas previstas pela RNA;
  2. Cálculo da função de custo para todos os exemplos dos dados de treinamento, que representa o erro entre a saída prevista pela RNA e a saída desejada;
  3. Cálculo do gradiente da função de custo em relação a todos os parâmetros da RNA;
  4. Atualização dos parâmetros da RNA na direção oposta ao gradiente de forma a reduzir o valor da função de custo;
  5. Repetição das etapas 1 a 4 até a obtenção de um valor desejado para a função de custo, ou até os parâmetros da RNA convergirem.

- Esse método é chamado de **Gradiente Descendente** $\Rightarrow$ que é o "motor" básico das redes neurais artificiais.

- O **Gradiente Descendente** é, então, um método iterativo onde a cada iteração, para um conjunto de dados de treinamento, são atualizados os parâmetros da RNA.

- A atualização dos parâmetros da RNA (etapa 5) é realizada de acordo com as seguintes equações:

$$w_{k,j}^{[l]} = w_{k,j}^{[l]} - \alpha \frac{\partial J(\mathbf{W},\mathbf{B})}{\partial w_{k,j}^{[l]}}, \text{ para todos os ganhos da RNA } (l = 1...L,\ k = 1...n^{[l]},\ j = 1...n^{[l-1]}) \quad (9)$$

$$b_{k}^{[l]} = b_{k}^{[l]} - \alpha \frac{\partial J(\mathbf{W},\mathbf{B})}{\partial b_{k}^{[l]}}, \text{ para todos os vieses da RNA } (l = 1...L,\ k = 1...n^{[l]}) \quad (10)$$

onde:

α = taxa de aprendizado (número pequeno <<< 1);

$\frac{\partial J(\mathbf{W},\mathbf{B})}{\partial w_{i,j}^{[l]}}$ = derivada parcial da função de custo em relação ao ganho $w_{i,j}^{[l]}$;

$\frac{\partial J(\mathbf{W},\mathbf{B})}{\partial b_{i}^{[l]}}$ = derivada parcial da função de custo em relação ao viés $b_{i}^{[l]}$.

- A Figura 2 apresenta o processo iterativo do método do Gradiente Descendente para o caso de uma RNA com somente um parâmetro (*w*).

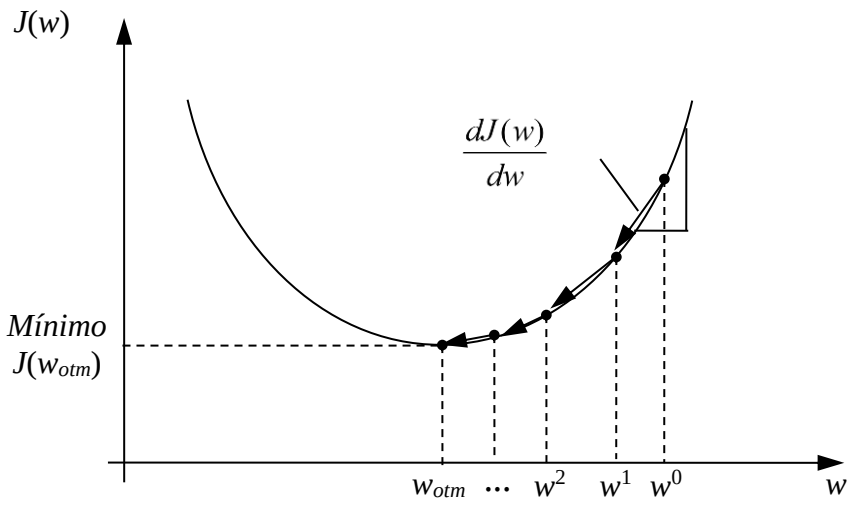


Figura 2. Esquema de funcionamento do método Gradiente Descendente.

- Na Figura 2, ao iniciarmos o processo à direita do ponto de mínimo da função de custo, então, a derivada da função de custo é positiva, assim, para aproximarmos do ponto mínimo temos que diminuir o parâmetro *w*. Por outro lado, se iniciarmos o processo à esquerda do ponto de mínimo da função de custo, então, a derivada da função de custo é negativa, assim, para aproximarmos do ponto mínimo temos que aumentar o parâmetro *w*.

- Observa-se que é importante escolher uma taxa de aprendizagem adequada:
  - Se for muito pequena o processo de convergência será muito lento e pode ficar preso em algum mínimo local;
  - Se for muito grande, as atualizações dos parâmetros podem levar a posições completamente aleatórias da curva e não convergir para o valor mínimo.

- Ressalta-se que a Figura 2 ilustra um caso não realístico para uma RNA com somente um único parâmetro $\Rightarrow$ na prática a função de custo depende de inúmeros parâmetros e, assim, temos

uma curva num espaço multidimensional, onde cada parâmetro da RNA é uma dimensão desse espaço e podem existir milhões desses parâmetros.

- A Figura 3 apresenta uma função de custo em um espaço bidimensional, somente para termos alguma idéia de como seria para um espaço multidimensional.

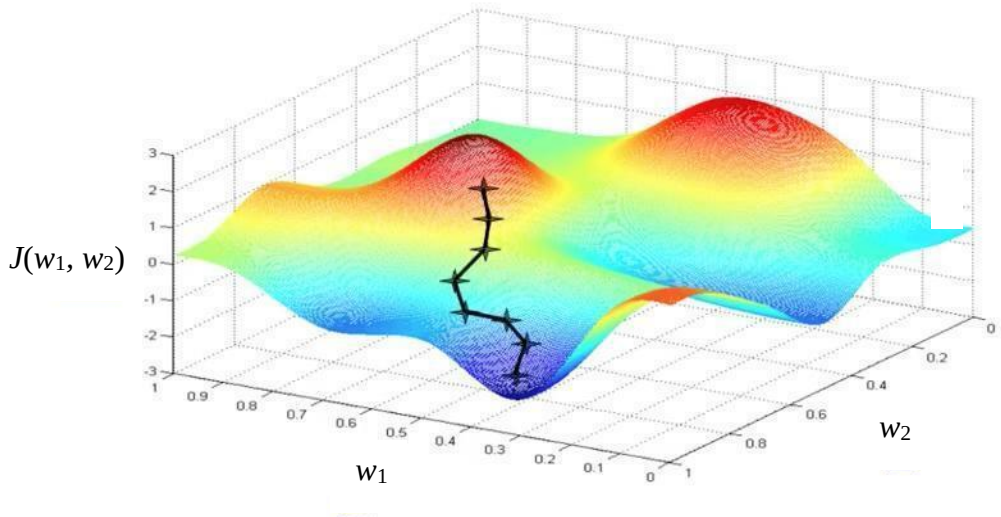


Figura 3. Exemplo de uma função de custo para uma RNA de dois parâmetros (https://github.com/RNogales94/ISI-Tensorflow).

- Existem diversas variantes do método gradiente descendente:
  - Algumas dessas variantes depende somente da forma como escolhemos os dados de treinamento em cada iteração do processo;
  - Outras variantes são modificações introduzidas para aumentar a eficiência do método para lidar com os mínimos locais da função de custo ⇒ veremos essas variações posteriormente.
- Ressalta-se que como o método do Gradiente descendente é um método iterativo é necessário inicializar os parâmetros da RNA de alguma forma.

5. Gradiente Descendente para uma RNA de um único neurônio

- O Gradiente Descendente é muito fácil de ser entendido para o caso de um único neurônio, o que auxilia estender o conceito para uma RNA mais complexa.
- Na Figura 4 é apresenta uma RNA de um único neurônio com duas entradas e uma única saída, utilizada para introduzir o método do Gradiente Descendente.

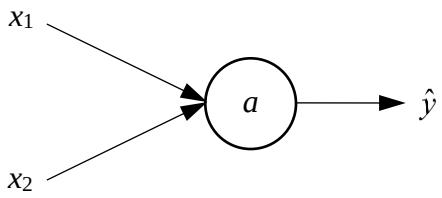


Figura 4. RNA de um único neurônio com duas entradas e uma saída.

- **Configuração da RNA:**
  - Número de entradas: $n_x = 2$;
  - Número de saídas: $n_y = 1$;
  - Número de exemplos de treinamento: *m;*
  - Número de camadas: $L = 1$;
  - Número de neurônios: $n = 1$;

- **Equações da RNA para cada exemplo do conjunto de treinamento (sobrescrito *i* que denota exemplo é eliminado para simplificar notação):**
  - Estado do neurônio:

$$z = w_1 x_1 + w_2 x_2 + b \tag{11}$$

  - Ativação do neurônio (saída da RNA):

$$\hat{y} = a = g(z) \tag{12}$$

- **Parâmetros da RNA:**
  - $\mathbf{W} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \Rightarrow$ dimensão $(1, n_x)$;
  - $b \Rightarrow$ escalar.

- **Função de custo:**

$$\boxed{J(w_1, w_2, b) = \frac{1}{m}\sum_{i=1}^{m} E(\hat{y}^{(i)}, y^{(i)})} \tag{13}$$

onde $E(\hat{y}^{(i)}, y^{(i)})$ é uma função de erro genérica.

- As **derivadas parciais da função de erro** em relação aos parâmetros da RNA para cada exemplo de treinamento são calculadas usando a **regra da cadeia da derivada**. Nas equações abaixo o sobrescrito (*i*) é eliminado para simplificar a notação.

$$\frac{\partial E(a,y)}{\partial w_1} = \left[\frac{\partial E(a,y)}{\partial a}\right]\left[\frac{\partial a}{\partial z}\right]\left[\frac{\partial z}{\partial w_1}\right] \tag{14}$$

$$\frac{\partial E(a,y)}{\partial w_2} = \left[\frac{\partial E(a,y)}{\partial a}\right]\left[\frac{\partial a}{\partial z}\right]\left[\frac{\partial z}{\partial w_2}\right] \tag{15}$$

$$\frac{\partial E(a,y)}{\partial b} = \left[\frac{\partial E(a,y)}{\partial a}\right]\left[\frac{\partial a}{\partial z}\right]\left[\frac{\partial z}{\partial b}\right] \tag{16}$$

Note que:

$\left[\frac{\partial E(a,y)}{\partial a}\right]$ = derivada da função de erro em relação à saída calculada ($\hat{y} = a$);

$\frac{\partial a}{\partial z} = \frac{d[g(z)]}{dz} \Rightarrow$ derivada da função de ativação *g* (ver equação 12);

$\frac{\partial z}{\partial w_1} = x_1$ (ver equação 11)

$\frac{\partial z}{\partial w_2} = x_2$ (ver equação 11)

$\frac{\partial z}{\partial b} = 1$ (ver equação 11)

Portanto, as derivadas parciais da função de erro em relação aos parâmetros da rede ficam:

$$\boxed{\frac{\partial E(a,y)}{\partial w_1} = \left[\frac{\partial E(a,y)}{\partial a}\right]\left[\frac{d[g(z)]}{\partial z}\right]x_1} \tag{17}$$

$$\boxed{\frac{\partial E(a,y)}{\partial w_2} = \left[\frac{\partial E(a,y)}{\partial a}\right]\left[\frac{d[g(z)]}{\partial z}\right]x_2} \tag{18}$$

$$\boxed{\frac{\partial E(a,y)}{\partial b} = \left[\frac{\partial E(a,y)}{\partial a}\right]\left[\frac{d[g(z)]}{\partial z}\right]} \tag{19}$$

- Função de custo considera todos os exemplos $\Rightarrow$ para obter as derivadas parciais da função de custo em relação aos parâmetros da RNA deve-se somar as derivadas parciais calculadas para cada um dos exemplos, ou seja:

$$\frac{\partial J(w_1,w_2,b)}{\partial w_1}=\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(\hat{y}^{(i)},y^{(i)})}{\partial w_1}\right]=\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a^{(i)},y^{(i)})}{\partial w_1}\right]$$
$$=\frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a^{(i)},y^{(i)})}{\partial a^{(i)}}\right]\left[\frac{d[g(z^{(i)})]}{dz^{(i)}}\right]x_1^{(i)}\right\} \tag{20}$$

$$\frac{\partial J(w_1,w_2,b)}{\partial w_2}=\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(\hat{y}^{(i)},y^{(i)})}{\partial w_2}\right]=\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a^{(i)},y^{(i)})}{\partial w_2}\right]$$
$$=\frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a^{(i)},y^{(i)})}{\partial a^{(i)}}\right]\left[\frac{d[g(z^{(i)})]}{dz^{(i)}}\right]x_2^{(i)}\right\} \tag{21}$$

$$\frac{\partial J(w_1,w_2,b)}{\partial b}=\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(\hat{y}^{(i)},y^{(i)})}{\partial b}\right]=\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a^{(i)},y^{(i)})}{\partial b}\right]$$
$$=\frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a^{(i)},y^{(i)})}{\partial a^{(i)}}\right]\left[\frac{d[g(z^{(i)})]}{dz^{(i)}}\right]\right\} \tag{22}$$

- Observe que o termo $\left[\frac{\partial E(a^{(i)},y^{(i)})}{\partial a^{(i)}}\right]\left[\frac{d[g(z^{(i)})]}{dz^{(i)}}\right]$ aparece em todas as derivadas parciais da função de custo em relação aos parâmetros da rede (equações 20, 21 e 22), assim, esse termo só precisa ser calculado uma única vez fazendo com que seja possível uma implementação eficiente desse cálculo.

- Tendo as derivadas parciais da função de custo para todos os exemplos de treinamento os pesos são atualizados iterativamente de forma a obter o conjunto de parâmetros da RNA que minimiza a função de custo, de acordo com as seguintes equações:

$$w_1=w_1-\alpha\frac{\partial J(w_1,w_2,b)}{\partial w_1}, \tag{23}$$

$$w_2=w_2-\alpha\frac{\partial J(w_1,w_2,b)}{\partial w_2}, \tag{24}$$

$$b=b-\alpha\frac{\partial J(w_1,w_2,b)}{\partial b}. \tag{25}$$

6. Gradiente Descendente para uma RNA rasa simples

- A aplicação do algoritmo do Gradiente Descendente para uma RNA rasa de uma única camada não é tão simples quanto para o caso de um único neurônio ⇒ dessa forma, vamos inicialmente apresentá-lo para uma RNA de uma única camada intermediária com dois neurônios e uma única saída, conforme mostrado na Figura 5.

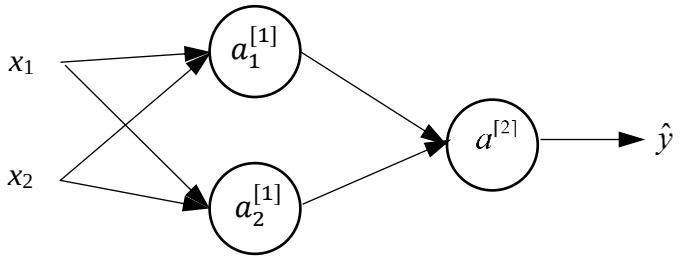


Figura 5. RNA rasa com duas entradas e uma saída para exemplificar o método Gradiente Descendente.

- **Configuração da RNA:**
  - Número de entradas: $n_x = 2$;
  - Número de saídas: $n_y = 1$;
  - Número de exemplos de treinamento: *m;*
  - Número de camadas: $L = 2$;
  - Número de neurônios da camada intermediária: $n^{[1]} = 2$.

- **Equações da RNA para cada exemplo do conjunto de treinamento (sobrescrito *i* que denota exemplo é eliminado para simplificar notação):**
  - Estados da camada intermediária:

$$\begin{cases} z_1^{[1]} = w_{11}^{[1]} x_1 + w_{12}^{[1]} x_2 + b_1^{[1]} \\ z_2^{[1]} = w_{21}^{[1]} x_1 + w_{22}^{[1]} x_2 + b_2^{[1]} \end{cases} \tag{26}$$

  - Ativações da camada intermediária:

$$\begin{cases} a_1^{[1]} = g^{[1]}(z_1^{[1]}) \\ a_2^{[1]} = g^{[1]}(z_2^{[1]}) \end{cases} \tag{27}$$

  - Estado da camada de saída:

$$z^{[2]} = w_1^{[2]} a_1^{[1]} + w_2^{[2]} a_2^{[1]} + b^{[2]} \tag{28}$$

- Ativação da camada saída (saída da RNA):

$$\hat{y} = a^{[2]} = g^{[2]}(z^{[2]}) \tag{29}$$

➢ **Parâmetros da RNA:**

- $\mathbf{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} \Rightarrow$ dimensão ($n^{[1]} = 2$, $n_x = 2$);
- $\mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_1^{[2]} \end{bmatrix} \Rightarrow$ dimensão ($n^{[1]} = 2$, 1);
- $\mathbf{W}^{[2]} = \begin{bmatrix} w_1^{[2]} & w_2^{[2]} \end{bmatrix} \Rightarrow$ dimensão ($n_y = 1$, $n^{[1]} = 2$);
- $b^{[2]} \Rightarrow$ escalar.

➢ **Função de custo:**

$$\boxed{J(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{W}^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} E(\hat{y}^{(i)}, y^{(i)})} \tag{30}$$

onde $E(\hat{y}^{(i)}, y^{(i)})$ é uma função de erro genérica.

➢ **As derivadas parciais da função de erro em relação aos parâmetros da camada de saída** da RNA para cada exemplo de treinamento são calculadas usando a regra da cadeia da derivada. Note que o sobrescrito (i) é eliminado para simplificar a notação.

$$\frac{\partial E(a^{[2]}, y)}{\partial w_1^{[2]}} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{\partial a^{[2]}}{\partial z^{[2]}}\right]\left[\frac{\partial z^{[2]}}{\partial w_1^{[2]}}\right] \tag{31}$$

$$\frac{\partial E(a^{[2]}, y)}{\partial w_2^{[2]}} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{\partial a^{[2]}}{\partial z^{[2]}}\right]\left[\frac{\partial z^{[2]}}{\partial w_2^{[2]}}\right] \tag{32}$$

$$\frac{\partial E(a^{[2]}, y)}{\partial b^{[2]}} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{\partial a^{[2]}}{\partial z^{[2]}}\right]\left[\frac{\partial z^{[2]}}{\partial b^{[2]}}\right] \tag{33}$$

Note que:

$\left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]$ = derivada da função de erro em relação à saída calculada ($\hat{y} = a^{[2]}$);

$\frac{\partial a^{[2]}}{\partial z^{[2]}} = \frac{d\left[g^{[2]}(z^{[2]})\right]}{dz^{[2]}} \Rightarrow$ derivada da função de ativação $g^{[2]}$ (ver equação 29);

$$\frac{\partial z^{[2]}}{\partial w_1^{[2]}} = a_1^{[1]} \text{ (ver equação 28);}$$

$$\frac{\partial z^{[2]}}{\partial w_2^{[2]}} = a_2^{[1]} \text{ (ver equação 28);}$$

$$\frac{\partial z^{[2]}}{\partial b^{[2]}} = 1 \text{ (ver equação 28).}$$

Portanto, as derivadas parciais da função de erro em relação aos parâmetros da camada de saída ficam:

$$\boxed{\frac{\partial E(a^{[2]}, y)}{\partial w_1^{[2]}} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] a_1^{[1]}} \quad (34)$$

$$\boxed{\frac{\partial E(a^{[2]}, y)}{\partial w_2^{[2]}} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] a_2^{[1]}} \quad (35)$$

$$\boxed{\frac{\partial E(a^{[2]}, y)}{\partial b^{[2]}} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right]} \quad (36)$$

- Para a **camada intermediária** vamos derivar a expressão para a **derivada parcial da função de erro** para o peso $w_{11}^{[1]}$ e depois estendemos para os demais pesos das ligações. Assim, usando a regra da cadeia da derivada, temos:

$$\frac{\partial E(a^{[2]}, y)}{\partial w_{11}^{[1]}} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{\partial a^{[2]}}{\partial z^{[2]}}\right]\left[\frac{\partial z^{[2]}}{\partial a_1^{[1]}}\right]\left[\frac{\partial a_1^{[1]}}{\partial z_1^{[1]}}\right]\left[\frac{\partial z_1^{[1]}}{\partial w_{11}^{[1]}}\right] \quad (37)$$

Note que:

$$\frac{\partial z^{[2]}}{\partial a_1^{[1]}} = w_1^{[2]} \text{ (ver equação 28);}$$

$$\frac{\partial a_1^{[1]}}{\partial z_1^{[1]}} = \frac{d\left[g^{[1]}(z_1^{[1]})\right]}{dz_1^{[1]}} \Rightarrow \text{derivada da função de ativação } g^{[1]} \text{ (ver equação 27);}$$

$$\frac{\partial z_1^{[1]}}{\partial w_{11}^{[1]}} = x_1 \text{ (ver equação 26).}$$

Portanto, a derivada parcial fica:

$$\frac{\partial E(a^{[2]},y)}{\partial w_{11}^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] w_1^{[2]} \left[\frac{dg^{[1]}(z_1^{[1]})}{dz_1^{[1]}}\right] x_1 \tag{38}$$

Analogamente para os outros pesos das ligações, temos:

$$\frac{\partial E(a^{[2]},y)}{\partial w_{12}^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] w_1^{[2]} \left[\frac{dg^{[1]}(z_1^{[1]})}{dz_1^{[1]}}\right] x_2 \tag{39}$$

$$\frac{\partial E(a^{[2]},y)}{\partial w_{21}^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] w_2^{[2]} \left[\frac{dg^{[1]}(z_2^{[1]})}{dz_2^{[1]}}\right] x_1 \tag{40}$$

$$\frac{\partial E(a^{[2]},y)}{\partial w_{22}^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] w_2^{[2]} \left[\frac{dg^{[1]}(z_2^{[1]})}{dz_2^{[1]}}\right] x_2 \tag{41}$$

No caso dos vieses da **camada intermediária** a **derivada parcial da função de erro** para o viés $b_1^{[1]}$ é dada por:

$$\frac{\partial E(a^{[2]},y)}{\partial b_1^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{\partial a^{[2]}}{\partial z^{[2]}}\right]\left[\frac{\partial z^{[2]}}{\partial a_1^{[1]}}\right]\left[\frac{\partial a_1^{[1]}}{\partial z_1^{[1]}}\right]\left[\frac{\partial z_1^{[1]}}{\partial b_1^{[1]}}\right] \tag{42}$$

Note novamente que:

$\frac{\partial z^{[2]}}{\partial a_1^{[1]}} = w_1^{[2]}$ (ver equação 28);

$\frac{\partial a_1^{[1]}}{\partial z_1^{[1]}} = \frac{d\left[g^{[1]}(z_1^{[1]})\right]}{dz_1^{[1]}} \Rightarrow$ derivada da função de ativação $g^{[1]}$ (ver equação 27);

e que:

$\frac{\partial z_1^{[1]}}{\partial b_1^{[1]}} = 1$ (ver equação 26).

Então, a derivada parcial fica:

$$\frac{\partial E(a^{[2]},y)}{\partial b_1^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] w_1^{[2]} \left[\frac{dg^{[1]}(z_1^{[1]})}{dz_1^{[1]}}\right] \tag{43}$$

Analogamente para o viés $b_2^{[1]}$ temos:

$$\frac{\partial E(a^{[2]},y)}{\partial b_2^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{\partial a^{[2]}}{\partial z^{[2]}}\right]\left[\frac{\partial z^{[2]}}{\partial a_2^{[1]}}\right]\left[\frac{\partial a_2^{[1]}}{\partial z_2^{[1]}}\right]\left[\frac{\partial z_2^{[1]}}{\partial b_2^{[1]}}\right] \tag{44}$$

resultando em:

$$\boxed{\frac{\partial E(a^{[2]},y)}{\partial b_2^{[1]}} = \left[\frac{\partial E(a^{[2]},y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] w_2^{[2]} \left[\frac{dg^{[1]}(z_2^{[1]})}{dz_2^{[1]}}\right]} \tag{45}$$

- A função de custo considera todos os exemplos ⇒ para obter as derivadas parciais da função de custo em relação aos parâmetros da RNA basta somar as derivadas parciais de cada um dos exemplos. Assim, temos:

$$\boxed{\begin{aligned}\frac{\partial J(\mathbf{W}^{[1]},\mathbf{b}^{[1]},\mathbf{W}^{[2]},b^{[2]})}{\partial w_k^{[2]}} &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E^{(i)}(\hat{y}^{(i)},y^{(i)})}{\partial w_k^{[2]}}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial w_k^{[2]}}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2](i)})}{dz^{[2]}}\right] a_k^{[1](i)}\right\}, \text{ para } k=1,2\end{aligned}} \tag{46}$$

$$\boxed{\begin{aligned}\frac{\partial J(\mathbf{W}^{[1]},\mathbf{b}^{[1]},\mathbf{W}^{[2]},b^{[2]})}{\partial b^{[2]}} &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E^{(i)}(y^{(i)},\hat{y}^{(i)})}{\partial b_k^{[2]}}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a^{[2](i)},y(i))}{\partial b^{[2]}}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2](i)})}{dz^{[2]}}\right]\right\}\end{aligned}} \tag{47}$$

$$\boxed{\begin{aligned}\frac{\partial J(\mathbf{W}^{[1]},\mathbf{b}^{[1]},\mathbf{W}^{[2]},b^{[2]})}{\partial w_{k,j}^{[1]}} &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E^{(i)}(\hat{y}^{(i)},y^{(i)})}{\partial w_{k,j}^{[1]}}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial w_{k,j}^{[1]}}\right] = \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2](i)})}{dz^{[2]}}\right] w_k^{[2]}\left[\frac{dg^{[1]}(z_k^{[1](i)})}{dz^{[1]}}\right] x_j\right\}, \\ &\text{para } k = 1,2,\ j = 1,2\end{aligned}} \tag{48}$$

$$\boxed{\begin{aligned}\frac{\partial J(\mathbf{W}^{[1]},\mathbf{b}^{[1]},\mathbf{W}^{[2]},b^{[2]})}{\partial b_k^{[1]}} &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E^{(i)}(\hat{y}^{(i)},y^{(i)})}{\partial b_k^{[1]}}\right] = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial b_k^{[1]}}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2](i)})}{dz^{[2]}}\right] w_k^{[2]}\left[\frac{dg^{[1]}(z_k^{[1](i)})}{dz^{[1]}}\right]\right\}, \\ &\text{para } k = 1,2\end{aligned}} \tag{49}$$

- Ressalta-se que o termo $\left[\frac{\partial E(a^{[2](i)},y^{(i)})}{\partial a^{[2]}}\right]\left[\frac{d[g(z^{[2](i)})]}{dz^{[2]}}\right]$ aparece em todas as derivadas parciais da função de custo em relação aos parâmetros da rede (equações 46 a 49), assim, esse termo só precisa ser calculado uma única vez fazendo com que seja possível uma implementação eficiente desse cálculo.

- Tendo as derivadas parciais da função de custo para todos os exemplos de treinamento os pesos são atualizados iterativamente de forma a obter o conjunto de parâmetros da RNA que minimiza a função de custo, de acordo com as equações (9) e (10).

7. Gradiente Descendente para uma RNA rasa

- Podemos generalizar o algoritmo do Gradiente Descendente para uma RNA rasa com múltiplas entradas e múltiplas saídas.

- Na Figura 6 é apresenta uma RNA rasa com múltiplas entradas e múltiplas saídas.

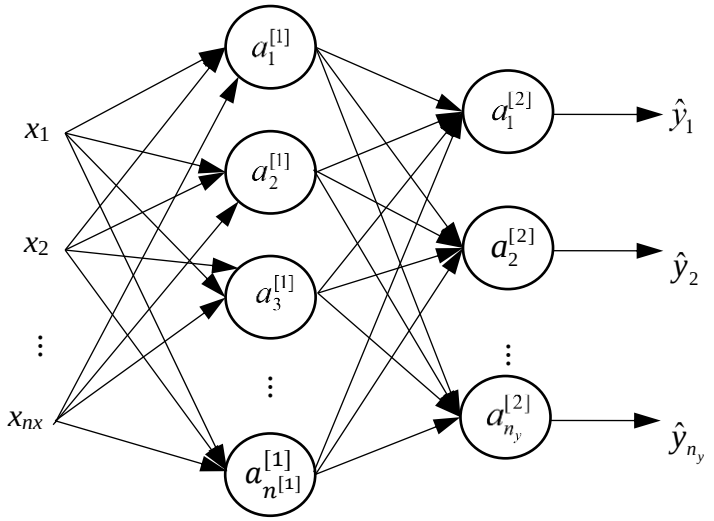


Figura 6. RNA rasa com múltiplas entradas e múltiplas saídas para exemplificar o método do Gradiente Descendente.

- **Configuração da RNA:**
  - Número de entradas: n_x;
  - Número de saídas: n_y;
  - Número de exemplos de treinamento: *m;*
  - Número de camadas: $L = 2$;
  - Número de neurônios da camada intermediária: $n^{[1]}$.

- **Equações da RNA para cada exemplo do conjunto de treinamento (sobrescrito *i* que denota exemplo é eliminado para simplificar notação):**
  - Estados da camada intermediária:

$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]} \tag{49}$$

  - Ativações da camada intermediária:

$$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]}) \tag{50}$$

- Estados da camada de saída:

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \tag{51}$$

- Ativação da camada saída (saída da RNA):

$$\hat{\mathbf{y}} = \mathbf{a}^{[2]} = g^{[2]}(\mathbf{z}^{[2]}) \tag{52}$$

➢ **Parâmetros da RNA:**

- $\mathbf{W}^{[1]} \Rightarrow$ dimensão ($n^{[1]}$, n_x);
- $\mathbf{W}^{[2]} \Rightarrow$ dimensão (n_y, $n^{[1]}$);
- $\mathbf{b}^{[1]} \Rightarrow$ dimensão ($n^{[1]}$, 1);
- $\mathbf{b}^{[2]} \Rightarrow$ dimensão (n_y, 1).

➢ **Função de custo:**

$$\boxed{J(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{W}^{[2]}, \mathbf{b}^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} E(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})} \tag{53}$$

onde $E(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$ é uma função de erro genérica. Note que no caso a saída da rede é um vetor de dimensão (n_y, 1).

➢ Para a **camada de saída** as **derivadas parciais da função de erro** em relação aos parâmetros da rede para cada exemplo de treinamento são calculadas usando a regra da cadeia da derivada. Note que sobrescrito *i* é eliminado das equações abaixo para simplificar a notação.

$$\frac{\partial E(a_k^{[2]}, y_k)}{\partial w_{k,j}^{[2]}} = \left[\frac{\partial E(a_k^{[2]}, y_k)}{\partial a_k^{[2]}}\right]\left[\frac{\partial a_k^{[2]}}{\partial z_k^{[2]}}\right]\left[\frac{\partial z_k^{[2]}}{\partial w_{k,j}^{[2]}}\right], \text{ para } k = 1,...,n_y \text{ e } j = 1,...,n^{[1]} \tag{54}$$

$$\frac{\partial E(a_k^{[2]}, y_k)}{\partial b_k^{[2]}} = \left[\frac{\partial E(a_k^{[2]}, y_k)}{\partial a_k^{[2]}}\right]\left[\frac{\partial a_k^{[2]}}{\partial z^{[2]}}\right]\left[\frac{\partial z_k^{[2]}}{\partial b_k^{[2]}}\right], \text{ para } k = 1,...,n_y \tag{55}$$

Note que os pesos relacionados ao *k*-ésimo neurônio da camada de saída somente dependem da saída desse neurônio ($\hat{y}_k = a_k^{[2]}$).

$\left[\frac{\partial E(a_k^{[2]}, y_k)}{\partial a_k^{[2]}}\right]$ = derivada da função de erro em relação à *k*-ésima saída ($\hat{y}_k = a_k^{[2]}$);

$\frac{\partial a_k^{[2]}}{\partial z_k^{[2]}} = \frac{d\left[g^{[2]}(z_k^{[2]})\right]}{dz_k^{[2]}} \Rightarrow$ derivada da função de ativação $g^{[2]}$ (ver equação 52);

$\frac{\partial z_k^{[2]}}{\partial w_{k,j}^{[2]}} = a_j^{[1]}$ (ver equação 51);

$\frac{\partial z_k^{[2]}}{\partial b_k^{[2]}} = 1$ (ver equação 51).

Substituindo essas expressões nas equações (54) e (55), as derivadas parciais da função de erro em relação aos parâmetros da camada de saída ficam:

$$\frac{\partial E(a_k^{[2]}, y_k)}{\partial w_{k,j}^{[2]}} = \left[\frac{\partial E(a_k^{[2]}, y_k)}{\partial a_k^{[2]}}\right]\left[\frac{dg^{[2]}(z_k^{[2]})}{dz_k^{[2]}}\right] a_j^{[1]}, \text{ para } k = 1,...,n_y \text{ e } j = 1,...,n^{[1]} \quad (56)$$

$$\frac{\partial E(a_k^{[2]}, y_k)}{\partial b_k^{[2]}} = \left[\frac{\partial E(a_k^{[2]}, y_k)}{\partial a_k^{[2]}}\right]\left[\frac{dg^{[2]}(z_k^{[2]})}{dz_k^{[2]}}\right], \text{ para } k = 1,...,n_y \quad (57)$$

- No caso da **camada intermediária**, em razão das saídas de todos os neurônios dessa camada contribuírem para todas as saídas da RNA, então, **as derivadas parciais** dos parâmetros da camada intermediária dependem de todas as saídas da rede. Isso pode ser melhor visualizado na Figura 7, onde as setas em vermelho representam que a derivada parcial da função de custo em relação ao peso $w_{11}^{[1]}$ dependem de todos os valores ligados à essa "cadeia" de cálculo.

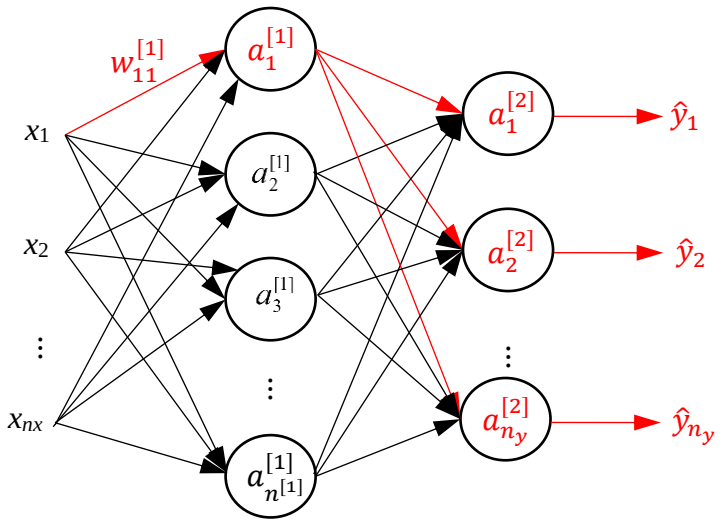


Figura 7. Representação da cadeia de cálculo da derivada parcial da função de custo em relação ao peso da camada intermediária $w_{11}^{[1]}$.

- Dessa forma, para calcular as derivadas parciais da função de erro em função dos parâmetros da camada intermediária temos que somar a contribuição de todas as saídas. Para os pesos das ligações temos:

$$\frac{\partial E(\hat{\mathbf{y}},\mathbf{y})}{\partial w_{k,j}^{[1]}} = \sum_{n=1}^{n_y}\left[\frac{\partial E(a_n^{[2]},y_n)}{\partial w_{k,j}^{[1]}}\right], \text{ para } k = 1,...,n^{[1]} \text{ e } j = 1,...,n_x \tag{58}$$

Aplicando a regra da cadeia da deriva, cada termo da somatória é dado por:

$$\frac{\partial E(a_n^{[2]},y_n)}{\partial w_{k,j}^{[1]}} = \left[\frac{\partial E(a_n^{[2]},y_n)}{\partial a_n^{[2]}}\right]\left[\frac{\partial a_n^{[2]}}{\partial z_n^{[2]}}\right]\left[\frac{\partial z_n^{[2]}}{\partial a_k^{[1]}}\right]\left[\frac{\partial a_k^{[1]}}{\partial z_k^{[1]}}\right]\left[\frac{\partial z_k^{[1]}}{\partial w_{k,j}^{[1]}}\right] \tag{59}$$

Note que:

$\left[\frac{\partial E(a_n^{[2]},y_n)}{\partial a_n^{[2]}}\right]$ = derivada da função de erro em relação à *n*-ésima saída ($\hat{y}_n = a_n^{[2]}$);

$\frac{\partial a_n^{[2]}}{\partial z_n^{[2]}} = \frac{d\left[g^{[2]}(z_n^{[2]})\right]}{dz_n^{[2]}} \Rightarrow$ derivada da função de ativação $g^{[2]}$ (ver equação 52);

$\frac{\partial z_n^{[2]}}{\partial a_k^{[1]}} = w_{n,k}^{[2]}$ (ver equação 51);

$\frac{\partial a_k^{[1]}}{\partial z_k^{[1]}} = \frac{d\left[g^{[1]}(z_k^{[1]})\right]}{dz_k^{[1]}} \Rightarrow$ derivada da função de ativação $g^{[1]}$ (ver equação 50);

$\frac{\partial z_k^{[1]}}{\partial w_{k,j}^{[1]}} = x_j$ (ver equação 49).

Substituindo essas expressões na equação (59) resulta em:

$$\frac{\partial E(a_n^{[2]},y_n)}{\partial w_{k,j}^{[1]}} = \left[\frac{\partial E(a_n^{[2]},y_n)}{\partial a_n^{[2]}}\right]\left[\frac{d\left[g^{[2]}(z_n^{[2]})\right]}{dz_n^{[2]}}\right] w_{n,k}^{[2]} \left[\frac{d\left[g^{[1]}(z_k^{[1]})\right]}{dz_k^{[1]}}\right] x_j \tag{60}$$

Substituindo a equação (60) na equação (58) obtemos as derivadas parciais da função de erro em relação aos pesos da camada intermediária, ou seja:

$$\boxed{\frac{\partial E(\hat{\mathbf{y}},\mathbf{y})}{\partial w_{k,j}^{[1]}} = \sum_{n=1}^{n_y}\left\{\left[\frac{\partial E(a_n^{[2]},y_n)}{\partial a_n^{[2]}}\right]\left[\frac{d\left[g^{[2]}(z_n^{[2]})\right]}{dz_n^{[2]}}\right] w_{n,k}^{[2]} \left[\frac{d\left[g^{[1]}(z_k^{[1]})\right]}{dz_k^{[1]}}\right] x_j\right\}, \quad \text{para } k = 1,...,n^{[1]} \text{ e } j = 1,...,n_x} \tag{61}$$

- Analogamente para os vieses dos neurônios da camada intermediária temos:

$$\frac{\partial E(\hat{\mathbf{y}},\mathbf{y})}{\partial b_k^{[1]}} = \sum_{n=1}^{n_y}\left[\frac{\partial E(a_n^{[2]},y_n)}{\partial b_k^{[1]}}\right], \text{ para } k = 1,...,n^{[1]} \tag{62}$$

onde cada termo da somatória é calculado usando a regra da cadeia, resultando em:

$$\frac{\partial E(a_n^{[2]}, y_n)}{\partial b_k^{[1]}} = \left[\frac{\partial E(a_n^{[2]}, y_n)}{\partial a_n^{[2]}}\right]\left[\frac{\partial a_n^{[2]}}{\partial z_n^{[2]}}\right]\left[\frac{\partial z_n^{[2]}}{\partial a_k^{[1]}}\right]\left[\frac{\partial a_k^{[1]}}{\partial z_k^{[1]}}\right]\left[\frac{\partial z_k^{[1]}}{\partial b_k^{[1]}}\right] \tag{63}$$

A única parcela que altera na equação (63) em relação à equação (59) é o último termo, ou seja:

$\frac{\partial z_k^{[1]}}{\partial b_k^{[1]}} = 1$ (ver equação 49).

Dessa forma, a equação (62) resulta em:

$$\frac{\partial E(\hat{\mathbf{y}}, \mathbf{y})}{\partial b_k^{[1]}} = \sum_{n=1}^{n_y}\left\{\left[\frac{\partial E(a_n^{[2]}, y_n)}{\partial a_n^{[2]}}\right]\left[\frac{d\left[g^{[2]}(z_n^{[2]})\right]}{dz_n^{[2]}}\right] w_{n,k}^{[2]}\left[\frac{d\left[g^{[1]}(z_k^{[1]})\right]}{dz_k^{[1]}}\right]\right\}, \text{ para } k = 1,...,n^{[1]} \tag{64}$$

- Função de custo considera todos os exemplos $\Rightarrow$ para obter as derivadas parciais da função de custo em relação aos parâmetros da camada intermediária basta somar as derivadas parciais de cada um dos exemplos, ou seja:

$$\begin{aligned}\frac{\partial J(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{W}^{[2]}, \mathbf{b}^{[2]})}{\partial w_{k,j}^{[2]}} &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a_k^{[2](i)}, y_k^{(i)})}{\partial w_{k,j}^{[2]}}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a_k^{[2](i)}, y_k^{(i)})}{\partial a_k^{[2](i)}}\right]\left[\frac{dg^{[2]}(z_k^{[2](i)})}{dz_k^{[2](i)}}\right] a_j^{[1](i)}\right\}, \\ &\qquad \text{para } k = 1,...,n_y \text{ e } j = 1,...,n^{[1]}\end{aligned} \tag{65}$$

$$\begin{aligned}\frac{\partial J(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{W}^{[2]}, \mathbf{b}^{[2]})}{\partial b_k^{[2]}} &= \frac{1}{m}\sum_{i=1}^{m}\left[\frac{\partial E(a_k^{[2](i)}, y_k^{(i)})}{\partial b_k^{[2]}}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\{\left[\frac{\partial E(a_k^{[2](i)}, y_k^{(i)})}{\partial a_k^{[2](i)}}\right]\left[\frac{dg^{[2]}(z_k^{[2](i)})}{dz_k^{[2](i)}}\right]\right\}, \text{ para } k = 1,..., n_y\end{aligned} \tag{66}$$

$$\begin{aligned}\frac{\partial J(\mathbf{W}^{[1]}, \mathbf{b}^{[1]}, \mathbf{W}^{[2]}, \mathbf{b}^{[2]})}{\partial w_{k,j}^{[1]}} &= \frac{1}{m}\sum_{i=1}^{m}\sum_{n=1}^{n_y}\left[\frac{\partial E(a_n^{[2](i)}, y_n^{(i)})}{\partial w_{k,j}^{[1]}}\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\{\sum_{n=1}^{n_y}\left\{\left[\frac{\partial E(a_n^{[2](i)}, y_n^{(i)})}{\partial a_n^{[2](i)}}\right]\left[\frac{d\left[g^{[2]}(z_n^{[2](i)})\right]}{dz_n^{[2](i)}}\right] w_{n,k}^{[2]}\left[\frac{d\left[g^{[1]}(z_k^{[1](i)})\right]}{dz_k^{[1](i)}}\right] x_j^{(i)}\right\}\right\}, \\ &\qquad \text{para } k = 1,...,n^{[1]} \text{ e } j = 1,...,n_x\end{aligned} \tag{67}$$

$$\frac{\partial J(\mathbf{W}^{[1]},\mathbf{b}^{[1]},\mathbf{W}^{[2]},\mathbf{b}^{[2]})}{\partial b_k^{[1]}} = \frac{1}{m}\sum_{i=1}^{m}\sum_{n=1}^{n_y}\left[\frac{\partial E(a_n^{[2](i)},y_n^{(i)})}{\partial b_k^{[1]}}\right]$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left\{\sum_{n=1}^{n_y}\left\{\left[\frac{\partial E(a_n^{[2](i)},y_n^{(i)})}{\partial a_n^{[2](i)}}\right]\left[\frac{d\left[g^{[2]}(z_n^{[2](i)})\right]}{dz_n^{[2](i)}}\right]w_{n,k}^{[2]}\left[\frac{d\left[g^{[1]}(z_k^{[1](i)})\right]}{dz_k^{[1](i)}}\right]\right\}\right\}, \qquad (68)$$

para $k = 1,...,n^{[1]}$

- Ressalta-se que os termos $\left[\frac{\partial E(a_k^{[2](i)},y_k^{(i)})}{\partial a_k^{[2](i)}}\right]\left[\frac{d[g(z_k^{[2](i)})]}{dz_k^{[2](i)}}\right]$, que são referentes à derivada parcial da função de custo em relação aos parâmetros da última camada se propagam para a camada intermediária e aparecem em todas as derivadas parciais da função de custo em relação aos parâmetros da rede (equações 65 a 68), assim, esse termo só precisa ser calculado uma única vez fazendo com que seja possível uma implementação eficiente desse cálculo.

- Tendo as derivadas parciais da função de custo para todos os exemplos de treinamento os parâmetros são atualizados iterativamente de acordo com as equações (9) e (10), para obter o conjunto de parâmetros da RNA que minimiza a função de custo,

7. Algoritmo de Retro-Propagação

- Na medida em que partes das derivadas parciais da função de custo em relação aos parâmetros da camada de saída são usados também nos cálculos das derivadas parciais da função de custo em relação aos parâmetros da camada intermediária, então, o cálculo do Gradiente Descendente pode ser implementado de forma eficiente usando um algoritmo denominado de **Retro-Propagação** ("**Back Propagation**").

- **Algoritmo de Retro-Propagação:**
  - Consiste de uma forma eficiente e conveniente para calcular as derivadas parciais da função de custo em relação aos parâmetros da RNA;
  - O algoritmo de Retro-Propagação é realizado em sequência ao cálculo das saídas da RNA para cada exemplo de treinamento.

- Cálculo das saídas da RNA $\Rightarrow$ **Propagação para Frente** ("**Foward Propagation**").

- O Quadro 1 apresenta o algoritmo para a implementação do cálculo da Propagação para Frente e da Retro Propagação para o treinamento de uma RNA rasa com uma única saída.

- **Vetorização de cálculo:**
  - Todos os cálculos, tanto da propagação para frente como da propagação para trás podem ser realizados matricialmente de forma bastante eficiente $\Rightarrow$ **cálculo vetorizado**;
  - É uma forma de evitar ao máximo o uso de comandos de repetição e, assim, tornar o cálculo muito mais rápido e eficiente computacionalmente;
  - Vetorização de cálculo é usado em muitas outras áreas da computação.

- No algoritmo do Quadro 1 a vetorização é implementada somente para cada exemplo ⇒ mas é possível vetorizar completamente esse cálculo eliminando o "loop" que percorre os exemplos de treinamento. Nesse caso, todos os cálculos são realizados por produtos de matrizes, para todos os exemplos usando uma única operação matricial.

Quadro 1. Algoritmo de cálculo da propagação para frente e da retro-propagação para uma RNA rasa de múltiplas saídas.

Inicialização dos parâmetros

$\mathbf{W}^{[1]}$ = 0,01*random($n^{[1]}$,n_x)

$\mathbf{b}^{[1]}$ = zeros($n^{[1]}$,1)

$\mathbf{W}^{[2]}$ = 0,01*random(n_y,$n^{[1]}$)

$\mathbf{b}^{[2]}$ = zeros(n_y,1)

Iteração das épocas

for época =1 to N_epocas_max

Inicialização da função de custo e das derivadas parciais

$J = 0$

$\partial J/\partial \mathbf{W}^{[1]}$ = zeros($n^{[1]}$,n_x)

$\partial J/\partial \mathbf{b}^{[1]}$ = zeros($n^{[1]}$,1)

$\partial J/\partial \mathbf{W}^{[2]}$ = zeros(n_y,$n^{[1]}$)

$\partial J/\partial \mathbf{b}^{[2]}$ = zeros(n_y,1)

Propagação para frente - Iteração em todos os exemplos de treinamento

for i = 1 to m

$$\mathbf{z}^{[1](i)}\big|_{(n^{[1]},1)} = \mathbf{W}^{[1]}\big|_{(n^{[1]},n_x)}\ \mathbf{x}^{(i)}\big|_{(n_x,1)} + \mathbf{b}^{[1]}\big|_{(n^{[1]},1)}$$

$$\mathbf{a}^{[1](i)}\big|_{(n^{[1]},1)} = g^{[1]}(\mathbf{z}^{[1](i)})\big|_{(n^{[1]},1)}$$

$$\mathbf{z}^{[2](i)}\big|_{(n_y,1)} = \mathbf{W}^{[2]}\big|_{(n_y,n^{[1]})}\ \mathbf{a}^{[1](i)}\big|_{(n^{[1]},1)} + \mathbf{b}^{[2]}\big|_{(n_y,1)}$$

$$\mathbf{a}^{[2](i)}\big|_{(n_y,1)} = g^{[2]}(\mathbf{z}^{[2](i)})\big|_{(n_y,1)}$$

Atualização da função de custo

$$J \mathrel{+}= E(\mathbf{a}^{[2](i)}, \mathbf{y}^{(i)})\big|_{(n_y,1)}$$

(continuação)

Multiplicação elemento por elemento

Retro-propagação

$$\partial E/\partial \mathbf{a}^{[2](i)}\big|_{(n_y,1)} = \frac{\partial E(\mathbf{a}^{[2](i)}, \mathbf{y}^{(i)})}{\partial a^{[2]}}\Big|_{(n_y,1)}$$

$$\partial E/\partial \mathbf{z}^{[2](i)}\big|_{(n_y,1)} = \partial E/\partial \mathbf{a}^{[2](i)}\big|_{(n_y,1)} * \frac{dg^{[2]}(\mathbf{z}^{[2](i)})}{dz^{[2]}}\Big|_{(n_y,1)}$$

$$\partial J/\partial \mathbf{W}^{[2]}\big|_{(n_y,n^{[1]})} \mathrel{+}= \partial E/\partial \mathbf{z}^{[2](i)}\big|_{(n_y,1)}\ \mathbf{a}^{[1](i)T}\big|_{(1,n^{[1]})}$$

$$\partial J/\partial \mathbf{b}^{[2]}\big|_{(n_y,1)} \mathrel{+}= \partial E/\partial \mathbf{z}^{[2](i)}\big|_{(n_y,1)}$$

$$\partial E/\partial \mathbf{a}^{[1](i)}\big|_{(n^{[1]},1)} = \mathbf{W}^{[2]T}\big|_{(n^{[1]},n_y)}\ \partial E/\partial \mathbf{z}^{[2](i)}\big|_{(n_y,1)}$$

$$\partial E/\partial \mathbf{z}^{[1](i)}\big|_{(n^{[1]},1)} = \partial E/\partial \mathbf{a}^{[1](i)}\big|_{(n^{[1]},1)} * \frac{dg^{[1]}(\mathbf{z}^{[1](i)})}{d\mathbf{z}^{[1]}}\Big|_{(n^{[1]},1)}$$

$$\partial J/\partial \mathbf{W}^{[1]}\big|_{(n^{[1]},n_x)} \mathrel{+}= \partial E/\partial \mathbf{z}^{[1](i)}\big|_{(n^{[1]},1)}\ \mathbf{x}^{(i)T}\big|_{(1,n_x)}$$

$$\partial J/\partial \mathbf{b}^{[1]}\big|_{(n^{[1]},1)} \mathrel{+}= \partial E/\partial \mathbf{z}^{[1](i)}\big|_{(n^{[1]},1)}$$

Final do loop de exemplos

Atualização dos parâmetros

$$\mathbf{W}^{[1]} = \mathbf{W}^{[1]} - \alpha \partial J/\partial \mathbf{W}^{[1]}\big|_{(n^{[1]},n_x)}$$

$$\mathbf{b}^{[1]} = \mathbf{b}^{[1]} - \alpha \partial J/\partial \mathbf{b}^{[1]}\big|_{(n^{[1]},1)}$$

$$\mathbf{W}^{[2]} = \mathbf{W}^{[2]} - \alpha \partial J/\partial \mathbf{W}^{[2]}\big|_{(n_y,n^{[1]})}$$

$$\mathbf{b}^{[2]} = \mathbf{b}^{[2]} - \alpha \partial J/\partial \mathbf{b}^{[2]}\big|_{(n_y,1)}$$

Final do loop de épocas

- Observe que no algoritmo do Quadro 1 existem dois "loops" de repetição:
  - Loop externo ⇒ percorre as épocas de treinamento;
  - Loop interno ⇒ percorre todos os exemplos em cada época de treinamento;
  - Em cada iteração ou época todos os exemplos de treinamento são apresentados à RNA.

- **ÉPOCA ⇒ representa cada iteração do algoritmo de atualização dos parâmetros da RNA para todos os exemplos de treinamento.**

- São necessárias várias épocas de treinamento para a função de custo atingir um valor muito pequeno e não variar mais, ou para os parâmetros da RNA convergirem para um valor constante.

- O algoritmo do Quadro 1 pode ser representado por um diagrama de blocos como mostrado na Figura 8.

- O diagrama da Figura 8 possui dois sentidos:

  - Da esquerda para a direita ⇒ Propagação para Frente;
  - Da direita para a esquerda ⇒ Retro-Propagação.

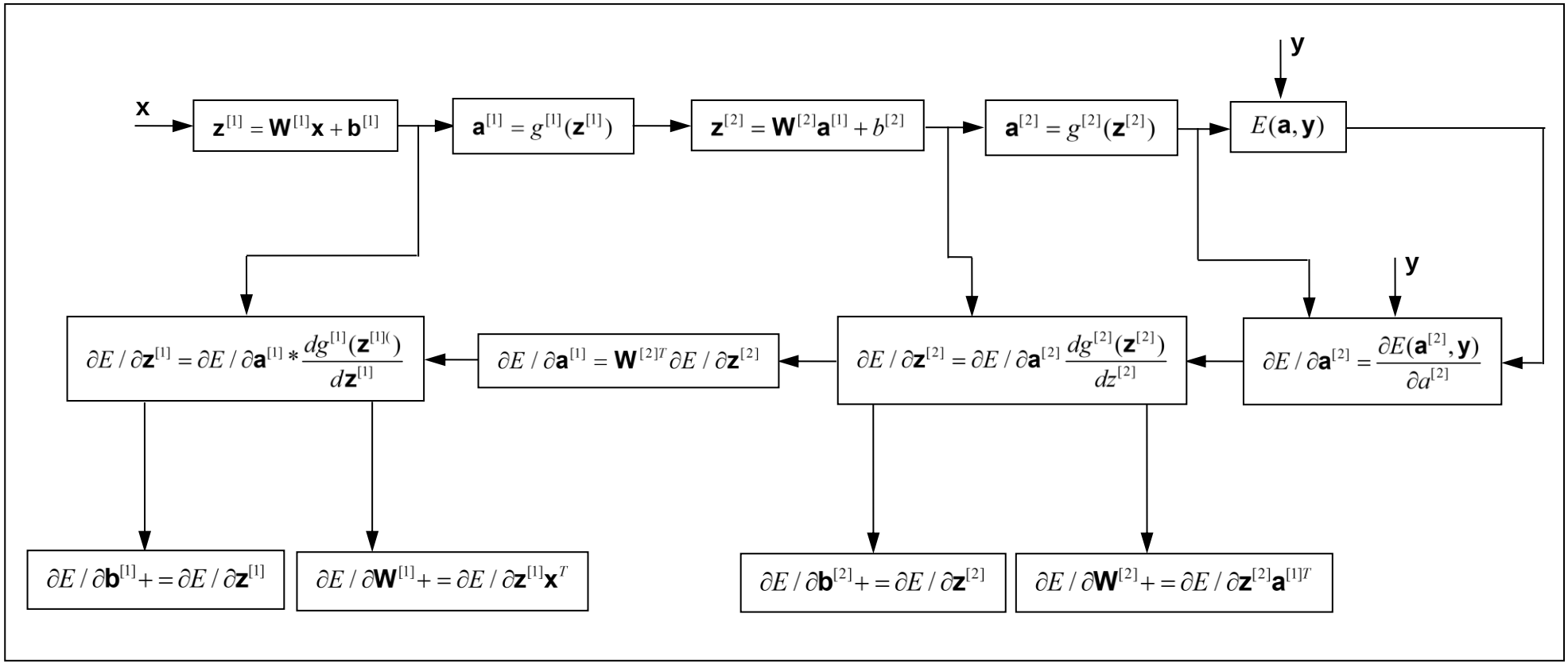


Figura 8. Fluxo de cálculo da propagação para frente e da retro propagação em um RNA rasa.

8. Inicialização dos parâmetros

- Como o método do Gradiente descendente é um método iterativo é necessário inicializar os parâmetros da RNA de alguma forma. Método utilizado:
  - Inicialização dos pesos das ligações com números aleatórios com distribuição uniforme e valores pequenos;
  - Inicialização dos vieses com zeros.

- No algoritmo do Quadro 1 temos:

$$\begin{cases} \mathbf{W}^{[1]} = 0{,}01*\text{random}(n^{[1]}, n_x) \\ \mathbf{b}^{[1]} = \text{zeros}(n^{[1]}, 1) \\ \mathbf{W}^{[2]} = 0{,}01*\text{random}(n_y, n^{[1]}) \\ \mathbf{b}^{[2]} = \text{zeros}(n_y, 1) \end{cases}$$

- Nesse caso os parâmetros são inicializados com números aleatórios com distribuição uniforme e média igual a 0,005.

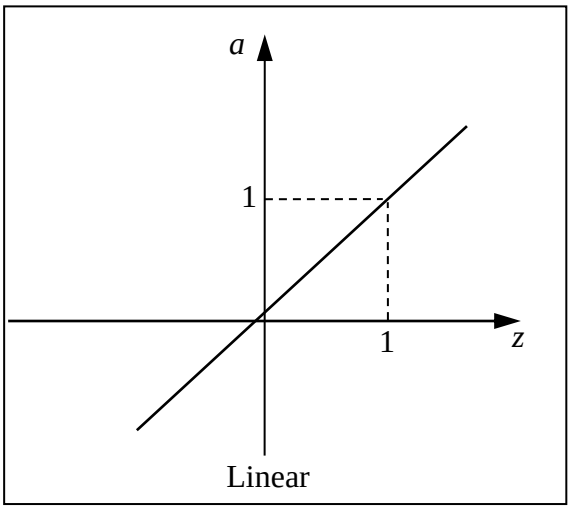
- Os valores iniciais dos parâmetros devem ser bem pequenos para garantir que os estados dos neurônios sejam próximos de zero de forma a evitar problemas de saturação das funções de ativação e de gradientes muito pequenos ou muito grandes.

- Existem métodos para inicializar os parâmetros de forma eficiente que veremos com mais detalhes posteriormente.

9. Derivadas das funções de ativação

- Como visto, o método do Gradiente Descendente utiliza as derivadas das funções de ativação.

Função linear

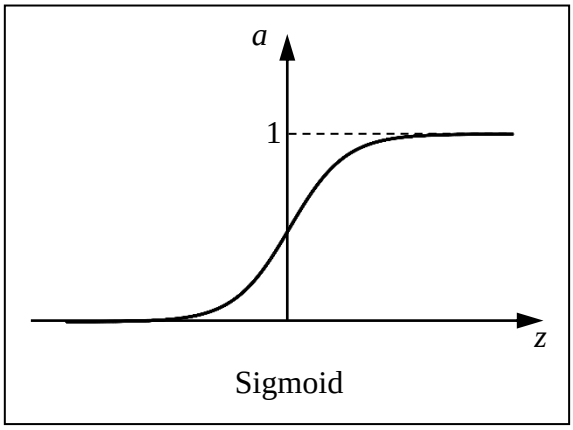


$$a = g(z) = z \tag{28}$$

Derivada em relação à z:

$$\boxed{\frac{dg(z)}{dz} = 1} \tag{29}$$

Função sigmóide



Sigmoid

$$\boxed{a = \sigma(z) = \frac{1}{1+e^{-z}}} \tag{30}$$

Derivada em relação à z:

$$\frac{d\sigma(z)}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} \tag{31}$$

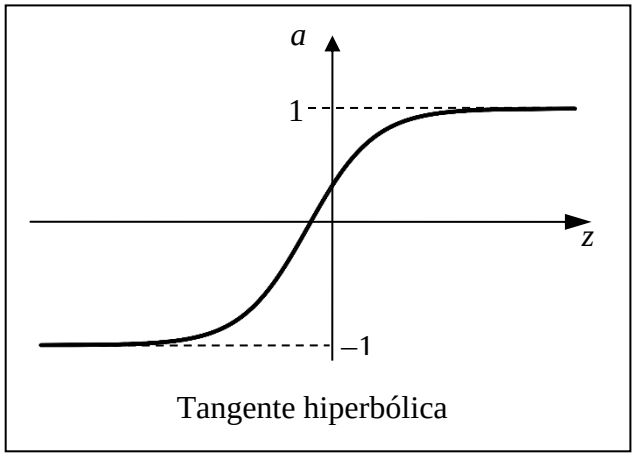
Rearranjando:

$$\frac{d\sigma(z)}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = \underbrace{\frac{1}{(1+e^{-z})}}_{a}\underbrace{\left[1-\frac{1}{(1+e^{-z})}\right]}_{(1-a)}$$

$$\boxed{\frac{d\sigma(z)}{dz} = a(1-a)} \tag{32}$$

Função tangente hiperbólica

$$\boxed{a = g(z) = \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}} \tag{33}$$



Tangente hiperbólica

Derivada em relação à *z*:

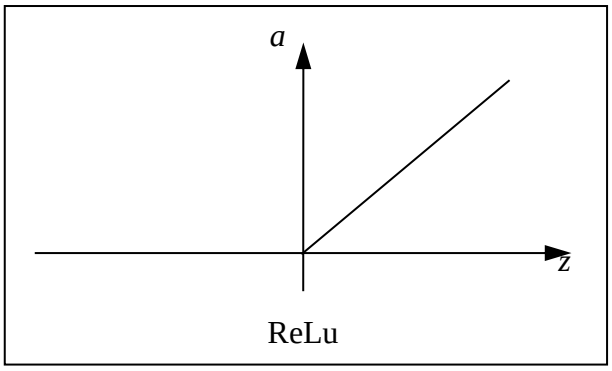
$$\frac{dg(z)}{dz} = \frac{(e^{z} + e^{-z})^2 - (e^{z} - e^{-z})^2}{(e^{z} + e^{-z})^2} \tag{34}$$

Rearranjando:

$$\frac{dg(z)}{dz} = \underbrace{\frac{(e^{z} + e^{-z})^2}{(e^{z} + e^{-z})^2}}_{1} - \underbrace{\left[\frac{(e^{z} - e^{-z})}{(e^{z} + e^{-z})}\right]^2}_{\tanh(z)}$$

$$\boxed{\frac{dg(z)}{dz} = 1 - a^2} \tag{35}$$

Função ReLu



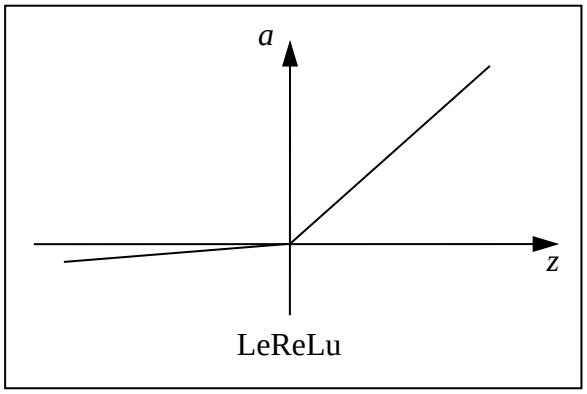
ReLu

$$\boxed{a = g(z) = \max(0, z) = \begin{cases} 0, \text{se } z < 0 \\ z, \text{se } z \geq 0 \end{cases}} \tag{36}$$

Derivada em relação à *z*:

$$\boxed{\frac{dg(z)}{dz} = \begin{cases} 0, \text{ se } z < 0 \\ 1, \text{ se } z \geq 0 \end{cases}} \tag{37}$$

Função Leaky ReLu



$$a = g(z) = \max(0{,}01z, z) = \begin{cases} 0{,}01z, \text{se } z < 0 \\ z, \text{se } z \geq 0 \end{cases} \tag{38}$$

Derivada em relação à *z*:

$$\frac{dg(z)}{dz} = \begin{cases} 0{,}01, \text{ se } z < 0 \\ 1, \quad \text{se } z \geq 0 \end{cases} \tag{39}$$

10. Processo de treinamento de uma RNA

- Como visto no algoritmo mostrado no Quadro 1, o treinamento de um RNA é um processo iterativo no qual se deseja calcular os parâmetros da RNA para que ela aprenda os padrões contidos nos exemplos fornecidos.

- Dado um conjunto de exemplos de treinamento e uma estrutura de RNA, o processo de treinamento consiste nas seguintes etapas:

  1. Inicializar os parâmetros da RNA;

  2. Executar a RNA com todos os exemplos de treinamento e obter todas as saídas previstas;

  3. Calcular os erros entre as saídas desejadas e as previstas pela RNA computando a função de custo;

  4. Calcular o gradiente da função de custo em relação a todos os parâmetros da RNA;

  5. Atualizar os parâmetros da RNA;

  6. Repetir os passos 2 a 5 quantas vezes for necessário para a função de custo atingir um valor mínimo desejado, ou até o número máximo de repetições for alcançado, ou até os parâmetros convergirem para valores constantes.