Um exemplo numérico do funcionamento de uma rede MLP usando *backpropagation*

André Paulino de Lima[1], **Sarajane Marques Peres**[1]

[1]Escola de Artes, Ciências e Humanidades – Universidade de São Paulo
São Paulo – SP, Brasil

{andre.p.lima, sarajane}@usp.br

1. Um diálogo inspirado em fatos reais

Um dia desses, no Lab de IA ...

- Aluno: Falaê, monitor, belê? Então, eu recebi um feedback da primeira entrega dizendo que existe algum problema com o número de pesos na minha rede.

- Monitor: Falaê! E qual foi o feedback?

- Aluno: Foi esse: ”Dimensões da matriz de pesos da camada escondida incompatível: Informada w0 = (1 , 26), Prevista w0 = (9, 27)”.

- Monitor: E quantas características você extraiu de cada imagem?

- Aluno: 26.

- Monitor: E quantos neurônios você alocou para a camada escondida?

- Aluno: 9.

- Monitor: Então a matriz de pesos da camada escondida deveria ter dimensões (9, 27).

- Aluno: Por quê?

2. Boa pergunta. Por quê?

Vamos simplificar um pouco o cenário, para ficar mais fácil de criar um diagrama para guiar a discussão. Imagine que o Aluno tenha extraído apenas 2 características (ao invés de 26) e que tenha alocado 3 neurônios na camada escondida (ao invés de 9). Podemos representar graficamente essas duas decisões de desenho de rede com o diagrama apresentado na Figura 1. Vamos adotar a mesma nomenclatura (nomes de variáveis) usada no livro texto da Faussett, seção 6.1.2.

2.1. Número de Entradas da Rede

O **número de entradas da rede** deve ser o número de características extraídas (n) de uma imagem mais um, por conta do bias. No nosso exemplo, como o vetor **x** possui 2 características ($n = 2$), a rede possui 3 entradas ($n + 1$). Vale lembrar que, diferentemente das entradas x_1 e x_2, que são alimentadas com valores provenientes do vetor de características **x** (*input training vector*, no diagrama da Figura 1 e no diagrama na Figura 6.1 do livro da Faussett), o valor da entrada x_0 é sempre alimentado com o valor 1.

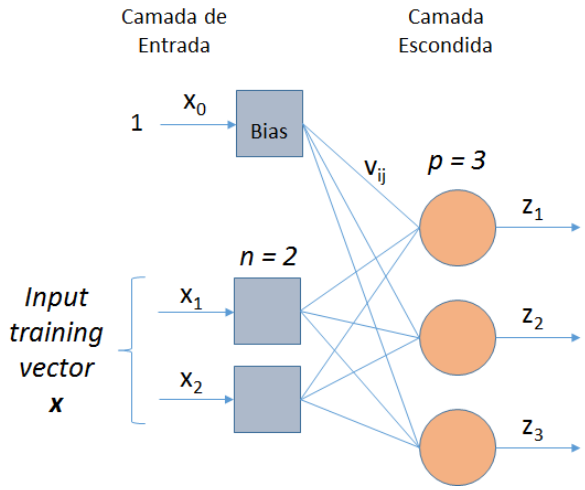

Figura 1. Uma representação da camada escondida.

2.2. Número de Pesos da Camada Escondida

Usando o resultado acima, podemos concluir que cada neurônio da camada escondida recebe uma cópia do valor nas $n + 1$ entradas da rede e computa uma saída, segundo a fórmula $z_j = f(v_{0j} + \sum_i x_i.v_{ij})$, em que f é a função de ativação (Faussett, página 292). Tomei a liberdade de representar a multiplicação entre escalares pelo operador ponto, inserido entre x_i e v_{ij}. Podemos reinterpretar essa equação da seguinte forma:

$$
\begin{aligned}
z_j = f(v_{0j} \quad & + \sum_{i} x_i.v_{ij}) \\
= f(v_{0j} \quad & + \sum_{i=1}^{n} x_i.v_{ij}) \\
= f(1.v_{0j} \quad & + \sum_{i=1}^{n} x_i.v_{ij}) \\
= f(x_0.v_{0j} \quad & + \sum_{i=1}^{n} x_i.v_{ij}) \\
z_j = f(\quad & \sum_{i=0}^{n} x_i.v_{ij})
\end{aligned}
\tag{1}
$$

Portanto, pelo desenvolvimento acima, cada neurônio da camada escondida precisa de $n + 1$ pesos $(v_{0j}, v_{1j}, \ldots, v_{nj})$ e, como temos p neurônios na camada escondida, o **número de pesos da camada escondida** é $p.(n+1)$. No diagrama da Figura 1, cada uma das conexões entre uma entrada e um neurônio representa um peso e, por inspeção visual, é possível contar 9 pesos. Essa contagem confere com o resultado esperado: $p.(n + 1) = 3.(2 + 1) = 9$.

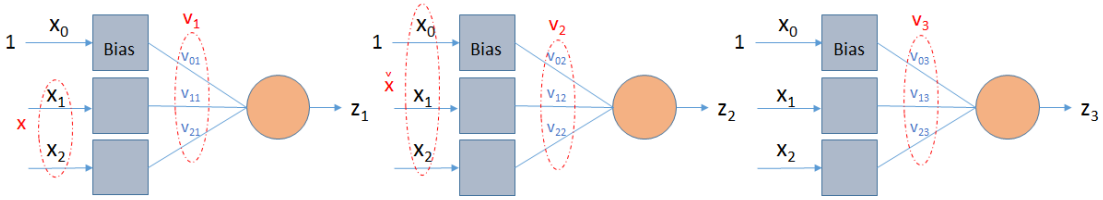

Figura 2. Representação dos pesos da camada escondida.

2.3. Matriz de Pesos da Camada Escondida

Para cada neurônio da camada escondida, executamos a computação expressa na Equação 1. Isso significa que usamos os mesmos valores de entrada (vetor **x**), mas cada neurônio usa seus próprios pesos. De certa forma, podemos reinterpretar o resultado da Equação 1 como:

$$
\begin{aligned}
z_j &= f(\sum_{i=0}^{n} x_i.v_{ij}) \\
&= f(\sum_{i=0}^{n} v_{ij}.x_i) \\
&= f(v_{0j}.x_0 + v_{1j}.x_1 + \ldots + v_{nj}.x_n) \\
z_j &= f(\mathbf{v}'_j \bullet \check{\mathbf{x}})
\end{aligned}
\tag{2}
$$

Em que $\check{\mathbf{x}}$ representa o vetor de características **x** acrescido do bias na posição 0 e $\mathbf{v}_j$ representa um vetor com os pesos do neurônio que produz a saída z_j, como ilustra a Figura 2. Tomei a liberdade de representar o produto entre vetores com o operador $\bullet$, e a operação de transposição com o operador $'$. Note que, como ambos possuem a mesma dimensão, o resultado do produto é um valor escalar.

Observando o diagrama da Figura 1, nota-se que a camada escondida produz p saídas. Se interpretarmos a saída da camada escondida como um vetor (**z**), poderíamos desenvolver uma expressão que reflita o comportamento da camada:

$$
\mathbf{z} = \begin{bmatrix} z_1 \\ \ldots \\ z_p \end{bmatrix} = \begin{bmatrix} f(\mathbf{v}'_1 \bullet \check{\mathbf{x}}) \\ \ldots \\ f(\mathbf{v}'_p \bullet \check{\mathbf{x}}) \end{bmatrix} = f(\begin{bmatrix} \mathbf{v}'_1 \bullet \check{\mathbf{x}} \\ \ldots \\ \mathbf{v}'_p \bullet \check{\mathbf{x}} \end{bmatrix}) = f(\begin{bmatrix} \mathbf{v}'_1 \\ \ldots \\ \mathbf{v}'_p \end{bmatrix} \bullet \check{\mathbf{x}}) = f(\mathbf{V} \bullet \check{\mathbf{x}})
\tag{3}
$$

Em que **V** é a matriz de pesos da camada escondida. Nesta matriz, a linha 1 corresponde aos pesos em $\mathbf{v}_1$, a linha 2 corresponde aos pesos em $\mathbf{v}_2$ e assim por diante. Por consequência, essa matriz possui $(n + 1)$ colunas. Essa matriz, portanto, apresenta dimensões $(p, n + 1)$ que, no nosso caso simplificado é (3, 2 + 1) = (3, 3) e, no caso do diálogo da seção 1, é (9, 26 + 1) = (9, 27).

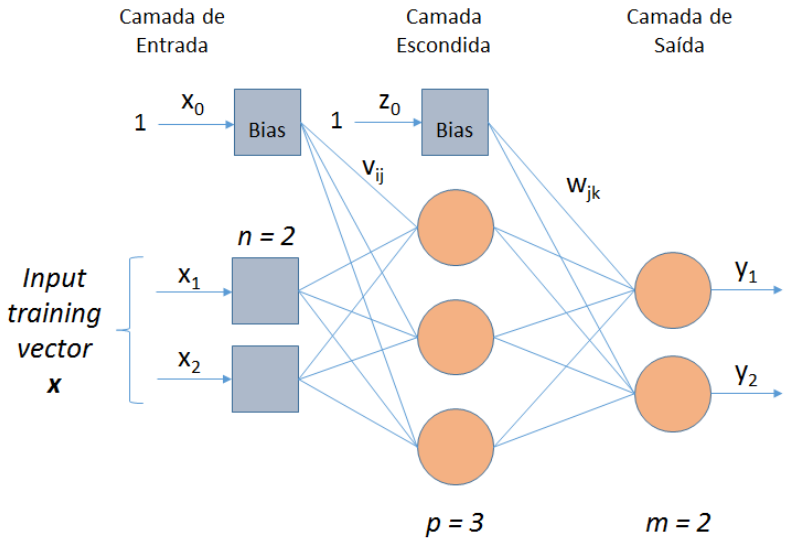

Figura 3. Uma rede consistente com os requisitos do EP.

2.4. Número de Pesos da Camada de Saída

Vamos estender nosso diagrama para incluir a camada de saída e, no espírito de manter o exemplo simples, vamos alocar apenas 2 neurônios. O diagrama da Figura 3 representa a rede MLP com duas camadas, consistente com os requisitos do EP.

Replicando a raciocínio que desenvolvemos para determinar o número de pesos na camada escondida, começamos por descrever a expressão que descreve o comportamento de um neurônio da camada de saída, a partir da equação usada no texto da Faussett:

$$
\begin{aligned}
y_k &= f(w_{0k} + \sum_{j} z_j.w_{jk}) \\
&= f(w_{0k} + \sum_{j=1}^{p} z_j.w_{jk}) \\
&= f(1.w_{0k} + \sum_{j=1}^{p} z_j.w_{jk}) \\
&= f(z_0.w_{0k} + \sum_{j=1}^{p} z_j.w_{jk}) \\
y_k &= f(\sum_{j=0}^{p} z_j.w_{jk})
\end{aligned}
\tag{4}
$$

Portanto, pelo desenvolvimento acima, cada neurônio da camada de saída precisa de $p+1$ pesos $(w_{0k}, w_{1k}, \ldots, w_{pk})$ e, como temos m neurônios na camada de saída, o **número de pesos da camada de saída** é $m.(p+1)$. No diagrama da Figura 3, cada uma das

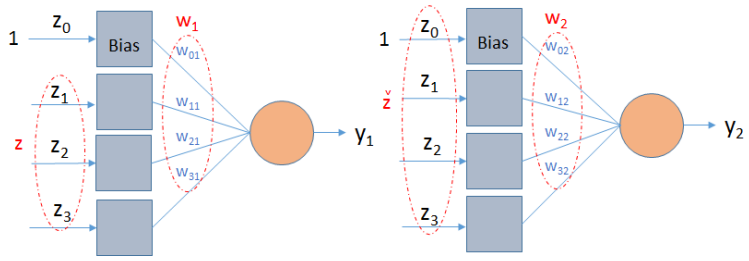

Figura 4. Representação dos pesos da camada de saída.

conexões entre as saídas dos neurônios da camada escondida e um neurônio da camada de saída representa um peso e, por inspeção visual, é possível contar 8 pesos. Essa contagem confere com o resultado esperado: $m.(p+1) = 2.(3+1) = 8$.

2.5. Matriz de Pesos da Camada de Saída

Novamente, replicando o raciocínio que desenvolvemos para determinar a estrutura da matriz de pesos da camada escondida, começamos por elaborar a versão matricial da computação executada por um neurônio:

$$\begin{aligned} y_k &= f(\sum_{j=0}^{p} z_j.w_{jk}) \\ &= f(\sum_{j=0}^{p} w_{jk}.z_j) \\ &= f(w_{0k}.z_0 + w_{1k}.z_1 + \ldots + w_{pk}.z_p) \\ &= f(\mathbf{w}'_k \bullet \check{\mathbf{z}}) \end{aligned} \tag{5}$$

Em que $\check{\mathbf{z}}$ representa o vetor $\mathbf{z}$ (saída da camada escondida) acrescido do bias na posição 0 e $\mathbf{w}_k$ representa um vetor com os pesos do neurônio que produz a saída y_k, como ilustrado na Figura 4. Como a camada de saída produz m saídas, podemos interpretar a saída da rede como um vetor ($\mathbf{y}$) e criar uma representação vetorial unificada para a camada:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \ldots \\ y_m \end{bmatrix} = \begin{bmatrix} f(\mathbf{w}'_1 \bullet \check{\mathbf{z}}) \\ \ldots \\ f(\mathbf{w}'_m \bullet \check{\mathbf{z}}) \end{bmatrix} = f(\begin{bmatrix} \mathbf{w}'_1 \bullet \check{\mathbf{z}} \\ \ldots \\ \mathbf{w}'_m \bullet \check{\mathbf{z}} \end{bmatrix}) = f(\begin{bmatrix} \mathbf{w}'_1 \\ \ldots \\ \mathbf{w}'_m \end{bmatrix} \bullet \check{\mathbf{z}}) = f(\mathbf{W} \bullet \check{\mathbf{z}}) \tag{6}$$

Em que $\mathbf{W}$ é a matriz de pesos da camada de saída. Nesta matriz, a linha 1 corresponde aos pesos em $\mathbf{w}_1$, a linha 2 corresponde aos pesos em $\mathbf{w}_2$ e assim por diante. Por consequência, essa matriz possui $(p+1)$ colunas. Essa matriz, portanto, apresenta dimensões $(m, p+1)$ que, no nosso caso simplificado é (2, 3 + 1) = (2, 4).

2.6. Uma MLP, uma equação

Nas seções anteriores, construímos equações que representam o comportamento de cada camada. Podemos usar as Equações 3 e 6 para construir uma única expressão que represente a rede neural descrita na Figura 3:

$$\mathbf{y} = f(\mathbf{W} \bullet \check{\mathbf{z}}) = f(\mathbf{W} \bullet \check{f}(\mathbf{V} \bullet \check{\mathbf{x}})) \tag{7}$$

Esta é uma representação mais compacta, concorda?

3. Um exemplo numérico

Usando a rede neural da Figura 3, vamos assumir que os pesos em **V** e **W** foram inicializados da seguinte forma:

$$\mathbf{V} = \begin{bmatrix} v_{01} & v_{11} & v_{21} \\ v_{02} & v_{12} & v_{22} \\ v_{03} & v_{13} & v_{23} \end{bmatrix} = \begin{bmatrix} -0.1 & 0.1 & -0.1 \\ -0.1 & 0.1 & 0.1 \\ 0.1 & -0.1 & -0.1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} w_{01} & w_{11} & w_{21} & w_{31} \\ w_{02} & w_{12} & w_{22} & w_{32} \end{bmatrix} = \begin{bmatrix} -0.1 & 0.1 & 0.0 & 0.1 \\ 0.1 & -0.1 & 0.1 & -0.1 \end{bmatrix}$$

Adicionalmente, vamos assumir que:

- a função sigmóide é implementada em todos os neurônios.
- a taxa de aprendizado é $\alpha = 0.5$.
- o vetor de características $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, cuja classificação esperada seja $\mathbf{t} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

3.1. Cálculo da saída da rede neural (*forward step*)

Esta seção cobre os passos 3, 4 e 5 do algoritmo da página 294 do livro da Faussett.

Vamos primeiro calcular a saída da camada escondida usando a formulação descrita pela Faussett. Use a Figura 1 e a Equação 1 como referências.

$$z_j = f(\sum_{i=0}^{n} x_i.v_{ij})$$

$$z_1 = f(\sum_{i=0}^{n} x_i.v_{i1}) = f(\sum_{i=0}^{2} x_i.v_{i1}) = f(x_0.v_{01} + x_1.v_{11} + x_2.v_{21})$$
$$= f(1 \times -0.1 + 1 \times 0.1 + 1 \times -0.1) = f(-0.1) = 0.4750$$

$$z_2 = f(\sum_{i=0}^{n} x_i.v_{i2}) = f(\sum_{i=0}^{2} x_i.v_{i2}) = f(x_0.v_{02} + x_1.v_{12} + x_2.v_{22})$$
$$= f(1 \times -0.1 + 1 \times 0.1 + 1 \times 0.1) = f(0.1) = 0.5250$$

$$z_3 = f(\sum_{i=0}^{n} x_i.v_{i3}) = f(\sum_{i=0}^{2} x_i.v_{i3}) = f(x_0.v_{03} + x_1.v_{13} + x_2.v_{23})$$
$$= f(1 \times 0.1 + 1 \times -0.1 + 1 \times -0.1) = f(-0.1) = 0.4750$$

Para finalizar, vamos calcular a saída da rede usando a formulação descrita pela Faussett. Use a Figura 3 e a Equação 4 como referências.

$$y_k = f(\sum_{j=0}^{p} z_j.w_{jk})$$

$$y_1 = f(\sum_{j=0}^{p} z_j.w_{j1}) = f(\sum_{j=0}^{3} z_j.w_{j1}) = f(z_0.w_{01} + z_1.w_{11} + z_2.w_{21} + z_3.w_{31})$$
$$= f(1 \times -0.1 + 0.4750 \times 0.1 + 0.5250 \times 0.0 + 0.4750 \times 0.1) = f(-0.1)$$
$$= 0.4988$$

$$y_2 = f(\sum_{j=0}^{p} z_j.w_{j2}) = f(\sum_{j=0}^{3} z_j.w_{j2}) = f(z_0.w_{02} + z_1.w_{12} + z_2.w_{22} + z_3.w_{32})$$
$$= f(1 \times 0.1 + 0.4750 \times -0.1 + 0.5250 \times 0.1 + 0.4750 \times -0.1) = f(-0.1)$$
$$= 0.5144$$

Observe que o mesmo resultado poderia ser obtido usando a equações 2 e 5:

$$\mathbf{z} = f(\mathbf{V} \bullet \check{\mathbf{x}})$$
$$= f(\begin{bmatrix} v_{01} & v_{11} & v_{21} \\ v_{02} & v_{12} & v_{22} \\ v_{03} & v_{13} & v_{23} \end{bmatrix} \bullet \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix})$$
$$= f(\begin{bmatrix} -0.1 & 0.1 & -0.1 \\ -0.1 & 0.1 & 0.1 \\ 0.1 & -0.1 & -0.1 \end{bmatrix} \bullet \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix})$$
$$= f(\begin{bmatrix} -0.1 \\ 0.1 \\ -0.1 \end{bmatrix}) = \begin{bmatrix} 0.4750 \\ 0.5250 \\ 0.4750 \end{bmatrix}$$

$$\mathbf{y} = f(\mathbf{W} \bullet \check{\mathbf{z}})$$
$$= f(\begin{bmatrix} w_{01} & w_{11} & w_{21} & w_{31} \\ w_{02} & w_{12} & w_{22} & w_{32} \end{bmatrix} \bullet \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix})$$
$$= f(\begin{bmatrix} -0.1 & 0.1 & 0.0 & 0.1 \\ 0.1 & -0.1 & 0.1 & -0.1 \end{bmatrix} \bullet \begin{bmatrix} 1 \\ 0.4750 \\ 0.5250 \\ 0.4750 \end{bmatrix})$$
$$= f(\begin{bmatrix} -0.0050 \\ 0.0575 \end{bmatrix}) = \begin{bmatrix} 0.4988 \\ 0.5144 \end{bmatrix}$$

3.2. Cálculo do ajuste dos Pesos da Camada de Saída (*backward step*)

Esta seção cobre o passo 6 do algoritmo descrito na página 294 do livro da Fausset, em que a **informação de erro** é definida como $\delta_k = (t_k - y_k).f'(y\_in_k)$, sendo f' a derivada da função f e $y\_in_k = \sum_j z_j.w_{jk}$. No caso da função sigmóide, podemos definir sua derivada em termos da própria função: $f'(a) = f(a).(1 - f(a))$. Assim, procedemos com o cálculo da informação de erro para os neurônios na saída da rede:

$$\delta_k = (t_k - y_k).f'(y\_in_k) = (t_k - y_k).f'(\sum_j z_j.w_{jk}) = (t_k - y_k).f'(\sum_{j=0}^{p} z_j.w_{jk}) \tag{8}$$

$$\begin{aligned}
\delta_1 &= (t_1 - y_1).f'(\sum_{j=0}^{3} z_j.w_{j1}) \\
&= (t_1 - y_1).f'(z_0.w_{01} + z_1.w_{11} + z_2.w_{21} + z_3.w_{31}) \\
&= (1 - 0.4988).f'(1 \times -0.1 + 0.4750 \times 0.1 + 0.5250 \times 0.0 + 0.4750 \times 0.1) \\
&= 0.5012 \times f'(-0.0050) = 0.5012 \times 0.2500 = 0.1253
\end{aligned}$$

$$\begin{aligned}
\delta_2 &= (t_2 - y_2).f'(\sum_{j=0}^{3} z_j.w_{j2}) \\
&= (t_2 - y_2).f'(z_0.w_{02} + z_1.w_{12} + z_2.w_{22} + z_3.w_{32}) \\
&= (0 - 0.5144).f'(1 \times 0.1 + 0.4750 \times -0.1 + 0.5250 \times 0.1 + 0.4750 \times -0.1) \\
&= -0.5144 \times f'(0.0575) = -0.5144 \times 0.2498 = -0.1285
\end{aligned}$$

Ainda no passo 6, o **termo de correção** é definido como $\Delta w_{jk} = \alpha.\delta_k.z_j$. Logo, os termos de ajuste dos pesos associados a cada neurônio da camada de saída são:

$$\begin{aligned}
\Delta w_{01} &= \alpha.\delta_1.z_0 = 0.5 \times 0.1253 \times 1 = & 0.0627 \\
\Delta w_{11} &= \alpha.\delta_1.z_1 = 0.5 \times 0.1253 \times 0.4750 = & 0.0298 \\
\Delta w_{21} &= \alpha.\delta_1.z_2 = 0.5 \times 0.1253 \times 0.5250 = & 0.0329 \\
\Delta w_{31} &= \alpha.\delta_1.z_3 = 0.5 \times 0.1253 \times 0.4750 = & 0.0298
\end{aligned}$$

$$\begin{aligned}
\Delta w_{02} &= \alpha.\delta_2.z_0 = 0.5 \times -0.1285 \times 1 = & -0.0642 \\
\Delta w_{12} &= \alpha.\delta_2.z_1 = 0.5 \times -0.1285 \times 0.4750 = & -0.0305 \\
\Delta w_{22} &= \alpha.\delta_2.z_2 = 0.5 \times -0.1285 \times 0.5250 = & -0.0337 \\
\Delta w_{32} &= \alpha.\delta_2.z_3 = 0.5 \times -0.1285 \times 0.4750 = & -0.0305
\end{aligned}$$

3.3. Ajuste dos Pesos da Camada Escondida (*backward step*)

Esta seção cobre o passo 7 do algoritmo da página 294 do livro da Fausset, em que a **informação de erro** é definida como $\delta_j = \delta\_in_j.f'(z\_in_j)$, sendo $\delta\_in_j = \sum_k \delta_k.w_{jk}$ e $z\_in_j = \sum_i x_i.v_{ij}$.

Como você pode notar, o símbolo δ está sendo empregado para se referir à informação de erro tanto da camada de saída (δ_k, no passo 6) quanto da camada escondida (δ_j, no passo 7), e isso pode causar confusão quando o subscrito é substituído por um número. Exemplo: δ_j representa um dos termos de ajuste da camada de saída e δ_k representa um dos termos de ajuste da camada escondida, na nomenclatura do livro texto. Entretanto, temos δ_1 tanto na camada escondida quanto na camada de saída. Para evitar a ambiguidade na representação, empregaremos a seguinte distinção: δ_1^s se refere a informação de erro computada para o primeiro neurônio da camada de saída, e δ_1^h se refere a informação de erro computada para o primeiro neurônio da camada escondida. Assim, procedemos com o cálculo da informação de erro para os neurônios da camada escondida:

$$\delta_j^h = \delta\_in_j^h.f'(z\_in_j) = \sum_k \delta_k^s.w_{jk}.f'(\sum_i x_i.v_{ij}) = \sum_{k=1}^{m} \delta_k^s.w_{jk}.f'(\sum_{i=0}^{n} x_i.v_{ij}) \tag{9}$$

$$\delta_1^h = \sum_{k=1}^{2} \delta_k^s.w_{1k}.f'(\sum_{i=0}^{2} x_i.v_{i1}) = \sum_{k=1}^{2} \delta_k^s.w_{1k}.f'(x_0.v_{01} + x_1.v_{11} + x_2.v_{21}) =$$

$$\begin{aligned}
\delta_1^h &= f'(x_0.v_{01} + x_1.v_{11} + x_2.v_{21}).\sum_{k=1}^{2} \delta_k^s.w_{1k} = \\
&= f'(x_0.v_{01} + x_1.v_{11} + x_2.v_{21}).(\delta_1^s.w_{11} + \delta_2^s.w_{12}) = \\
&= f'(1 \times -0.1 + 1 \times 0.1 + 1 \times -0.1) \times (0.1253 \times 0.1 - 0.1285 \times -0.1) = \\
&= f'(-0.1) \times (0.0254) = 0.2494 \times 0.0254 = 0.0063
\end{aligned}$$

$$\begin{aligned}
\delta_2^h &= f'(x_0.v_{02} + x_1.v_{12} + x_2.v_{22}).(\delta_1^s.w_{21} + \delta_2^s.w_{22}) = \\
&= f'(1 \times -0.1 + 1 \times 0.1 + 1 \times 0.1) \times (0.1253 \times 0.0 - 0.1285 \times 0.1) = \\
&= f'(0.1) \times (-0.0128) = 0.2494 \times -0.0128 = -0.0032
\end{aligned}$$

$$\begin{aligned}
\delta_3^h &= f'(x_0.v_{03} + x_1.v_{13} + x_2.v_{23}).(\delta_1^s.w_{31} + \delta_2^s.w_{32}) = \\
&= f'(1 \times 0.1 + 1 \times -0.1 + 1 \times -0.1) \times (0.1253 \times 0.1 - 0.1285 \times -0.1) = \\
&= f'(-0.1) \times (0.0254) = 0.2494 \times 0.0254 = 0.0063
\end{aligned}$$

Ainda no passo 7, o **termo de correção** é definido como $\Delta v_{ij} = \alpha.\delta_j.x_i$. Logo, os

termos de ajuste de pesos associados a cada neurônio da camada escondida são:

$$\begin{aligned}
\Delta v_{01} = \alpha.\delta_1^h.x_0 = 0.5 \times 0.0063 \times 1 = & \quad 0.0032 \\
\Delta v_{11} = \alpha.\delta_1^h.x_1 = 0.5 \times 0.0063 \times 1 = & \quad 0.0032 \\
\Delta v_{21} = \alpha.\delta_1^h.x_2 = 0.5 \times 0.0063 \times 1 = & \quad 0.0032
\end{aligned}$$

$$\begin{aligned}
\Delta v_{02} = \alpha.\delta_2^h.x_0 = 0.5 \times -0.0032 \times 1 = & \quad -0.0016 \\
\Delta v_{12} = \alpha.\delta_2^h.x_1 = 0.5 \times -0.0032 \times 1 = & \quad -0.0016 \\
\Delta v_{22} = \alpha.\delta_2^h.x_2 = 0.5 \times -0.0032 \times 1 = & \quad -0.0016
\end{aligned}$$

$$\begin{aligned}
\Delta v_{03} = \alpha.\delta_3^h.x_0 = 0.5 \times 0.0063 \times 1 = & \quad 0.0032 \\
\Delta v_{13} = \alpha.\delta_3^h.x_1 = 0.5 \times 0.0063 \times 1 = & \quad 0.0032 \\
\Delta v_{23} = \alpha.\delta_3^h.x_2 = 0.5 \times 0.0063 \times 1 = & \quad 0.0032
\end{aligned}$$

3.4. Ajuste dos Pesos

Esta seção cobre o passo 8 do algoritmo descrito na página 294 do livro da Fausset. No passo 8, cada peso da camada de saída é ajustado segundo a regra delta, $w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$. Logo, os novos valores atribuídos aos pesos da camada de saída são:

$$\begin{aligned}
w_{01} \leftarrow w_{01} + \Delta w_{01} = -0.1 + 0.0627 = & \quad -0.0373 \\
w_{11} \leftarrow w_{11} + \Delta w_{11} = 0.1 + 0.0298 = & \quad 0.1298 \\
w_{21} \leftarrow w_{21} + \Delta w_{21} = 0.0 + 0.0329 = & \quad 0.0329 \\
w_{31} \leftarrow w_{31} + \Delta w_{31} = 0.1 + 0.0298 = & \quad 0.1298
\end{aligned}$$

$$\begin{aligned}
w_{02} \leftarrow w_{02} + \Delta w_{02} = 0.1 - 0.0642 = & \quad 0.0358 \\
w_{12} \leftarrow w_{12} + \Delta w_{12} = -0.1 - 0.0305 = & \quad -0.1305 \\
w_{22} \leftarrow w_{22} + \Delta w_{22} = 0.1 - 0.0337 = & \quad 0.0663 \\
w_{32} \leftarrow w_{32} + \Delta w_{32} = -0.1 - 0.0305 = & \quad -0.1305
\end{aligned}$$

Ainda no passo 8, cada peso da camada escondida é ajustado segundo à regra delta, $v_{ij} \leftarrow v_{ij} + \Delta v_{ij}$. Logo, os novos valores atribuídos aos pesos da camada escondida são:

$$\begin{aligned}
v_{01} \leftarrow v_{01} + \Delta v_{01} = -0.1 + 0.0032 = & \quad -0.0968 \\
v_{11} \leftarrow v_{11} + \Delta v_{11} = 0.1 + 0.0032 = & \quad 0.1032 \\
v_{21} \leftarrow v_{21} + \Delta v_{21} = -0.1 + 0.0032 = & \quad -0.0968
\end{aligned}$$

$$
\begin{aligned}
v_{02} \leftarrow v_{02} + \Delta v_{02} &= -0.1 - 0.0016 = \quad -0.1016 \\
v_{12} \leftarrow v_{12} + \Delta v_{12} &= 0.1 - 0.0016 = \quad\ \ 0.0984 \\
v_{22} \leftarrow v_{22} + \Delta v_{22} &= 0.1 - 0.0016 = \quad\ \ 0.0984
\end{aligned}
$$

$$
\begin{aligned}
v_{03} \leftarrow v_{03} + \Delta v_{03} &= 0.1 + 0.0032 = \quad\ \ 0.1032 \\
v_{13} \leftarrow v_{13} + \Delta v_{13} &= -0.1 + 0.0032 = \quad -0.0968 \\
v_{23} \leftarrow v_{23} + \Delta v_{23} &= -0.1 + 0.0032 = \quad -0.0968
\end{aligned}
$$

PS: Você pode estar imaginando se existe uma formulação matricial, como aquela que mostramos no final da seção 3.1, mais compacta, simples de executar e que possa ser aplicada para computar os cálculos de ajuste dos pesos e os próprios ajustes de peso. A resposta é sim!