SCC0276 - Aprendizado de Máquin

Aula - *Multi-Layer Perceptron* (MLP)

Profa. Dra. Roseli Aparecida Francelin Romero
SCC - ICMC - USP

2019

Sumário

Perceptron multicamadas

- Redes de apenas uma camada representam somente funções linearmente separáveis.
- Redes de múltiplas camadas solucionam essa restrição.
- O desenvolvimento do algoritmo *backpropagation* foi um dos motivos para o ressurgimento da área de redes neurais [Rumelhart *et. al*, 1986].

Sumário

Modelo de rede neural com múltiplas camadas.

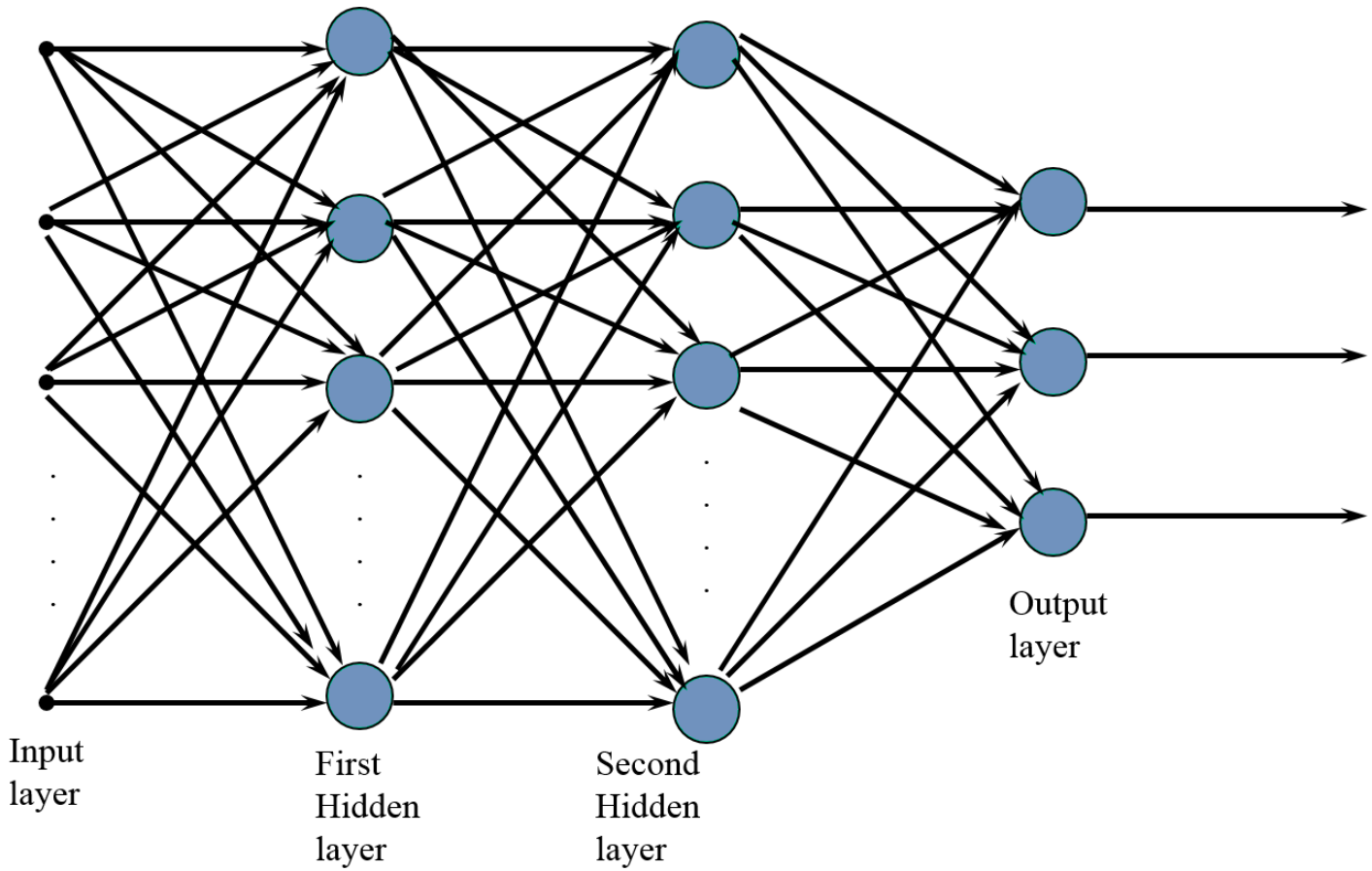


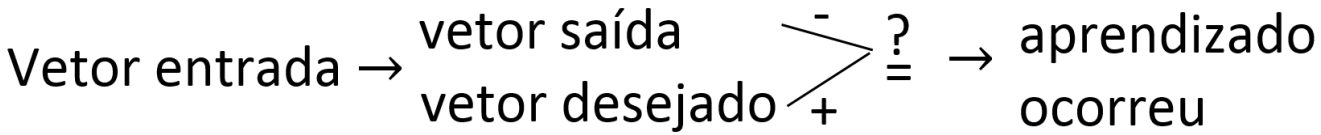
Figura 1: Rede neural *feed-forward* com múltiplas camadas.

Sumário

Sumário

Aprendizado da rede

- O esquema de aprendizado da rede pode ser descrito do seguinte modo:



Aprendizado da rede

- Caso contrário, os pesos são modificados para minimizar o erro:

$$E(w) = \sum_{p=1}^{N} E_p(w)$$

onde N é o no. total de padrões e E_p é o erro quadrático referente a cada par p apresentado à rede, sendo dado por:

$$E_p = \frac{1}{2} \sum_{j} (t_{pj} - y_{pj})^2$$

onde:

- t_{pj}: j-ésima componente do vetor saída desejada.
- y_{pj}: j-ésima componente do vetor obtido pela rede.

Aprendizado da rede

Pesos (*Gradient Descent Method*)

$$w_{ji}(k+1) = w_{ji}(k) - \eta \frac{\partial E_p(w)}{\partial w_{ji}}\Big|_{w(k)}$$

Onde η é uma constante positiva (velocidade de aprendizado).

- Calculando a derivada parcial do E_p, tem-se:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial y_{pj}} \cdot \frac{\partial y_{pj}}{\partial v_{pj}} \cdot \frac{\partial v_{pj}}{\partial w_{ji}}$$

- Para se calcular $\frac{\partial E_p}{\partial y_{pj}}$, dois casos devem ser considerados:

Aprendizado da rede

1. Neurônio j está na camada de saída.

$$\frac{\partial E_p}{\partial y_{pj}} = -(t_{pj} - y_{pj})$$

$$\therefore \frac{\partial E_p}{\partial w_{ji}} = \underbrace{-(t_{pj} - y_{pj}) \cdot \overbrace{y_{pj}(1 - y_{pj})}^{\frac{\partial y_{pj}}{\partial v_{pj}}}}_{\delta_{pj}} \cdot y_{pi}$$

$$\boxed{\frac{\partial E_p}{\partial w_{ji}} = -\delta_{pj} \cdot y_{pi}} \rightarrow \text{erro na camada de saída}$$

onde $-\delta_{pj} = \frac{\partial E_p}{\partial v_{pj}}$

Aprendizado da rede

2. Neurônio j está na camada oculta (escondida).
   - Nesse caso, não se conhece a expressão do erro.
   - Para obtermos $\frac{\partial E_p}{\partial y_{pj}}$, usamos mais uma vez a **regra da cadeia**.

$$\frac{\partial E_p}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial v_{pk}} \cdot \frac{\partial v_{pk}}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial v_{pk}} \cdot \frac{\partial \left(\sum_j w_{kj} y_{pj} \right)}{\partial y_{pj}}$$

$$= \sum_k \frac{\partial E_p}{\partial v_{pk}} \cdot w_{kj} = \sum_k \left(-\delta_{pk} \cdot w_{kj} \right)$$

$$\therefore \boxed{\frac{\partial E_p}{\partial w_{ji}} = \left(\sum_k \left(-\delta_{pk} w_{kj} \right) \right) \cdot y_{pj}(1 - y_{pj}) \cdot y_{pi}}$$

erro na camada oculta

Aprendizado da rede

- **Observação:** os erros são computados no sentido *backward*. O erro foi chamado de *back-propagado* $\rightarrow$ algoritmo de aprendizado **backpropagation** (BP).

Algoritmo *Backpropagation*

- **Inicialização:** pesos iniciados com valores aleatórios e pequenos ($[-1, +1]$).
- **Treinamento - Repita:**
  - Considere um novo padrão de entrada x_i e seu respectivo vetor de saída t_i desejado do conjunto de treinamento.
  - **Repita:**
    - Apresentar o par (x_i, t_i). **(modo padrão)**
    - Calcular as saídas dos processadores, começando da primeira camada escondida até a camada de saída.
    - Calcular o erro na camada de saída.
    - Atualizar os pesos de cada processador, começando pela camada de saída, até a camada de entrada.
  - **Até que o erro quadrático médio para esse padrão seja** $<= tol1$.
- **Até que o erro quadrático médio seja** $<= tol2$ **para todos os padrões do conjunto de treinamento.**

Sumário

Processo de aprendizado

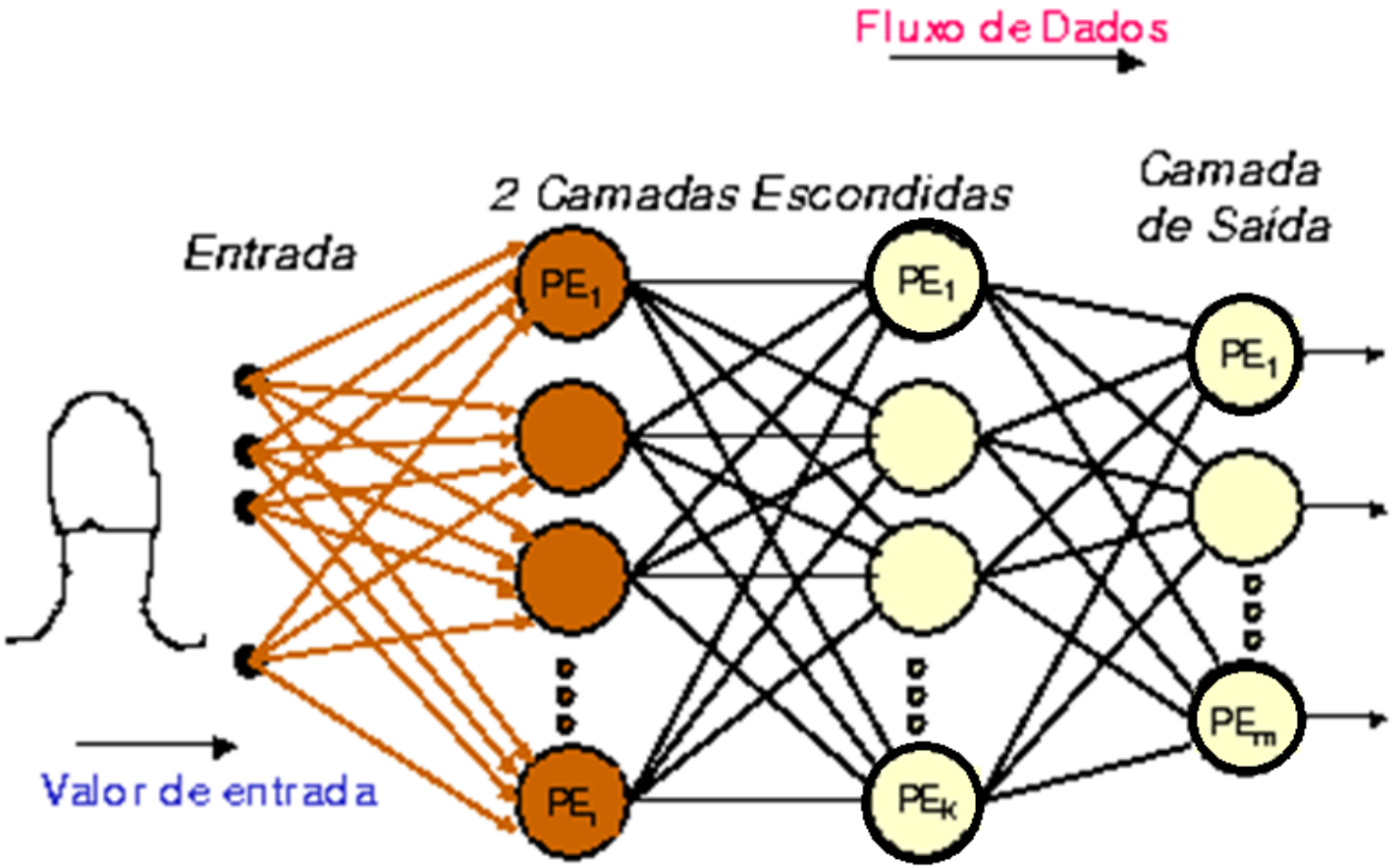


Figura 2: *Feed-forward* (fase 1), primeira camada escondida.

Processo de aprendizado

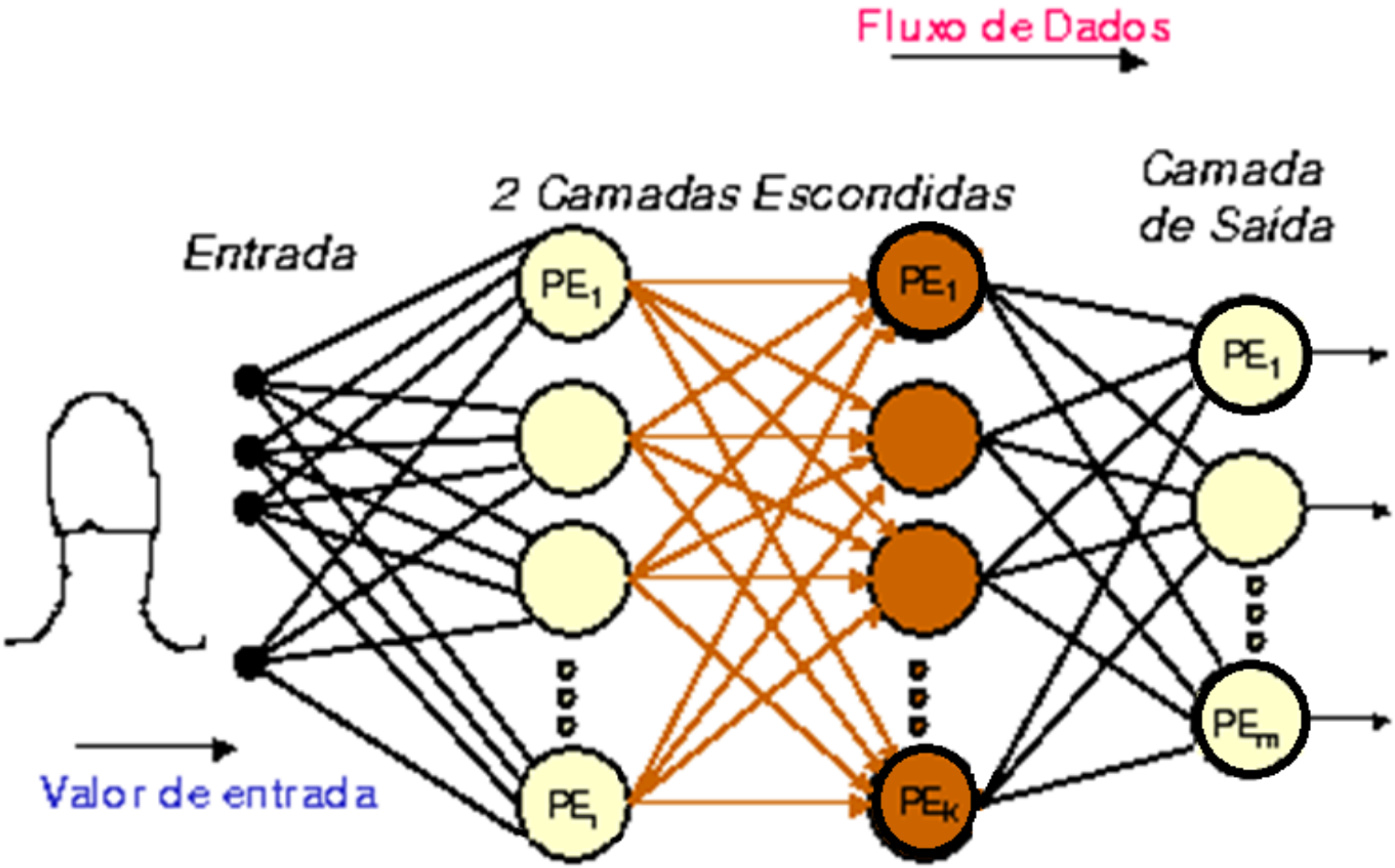


Figura 3: *Feed-forward* (fase 1), segunda camada escondida.

Processo de aprendizado

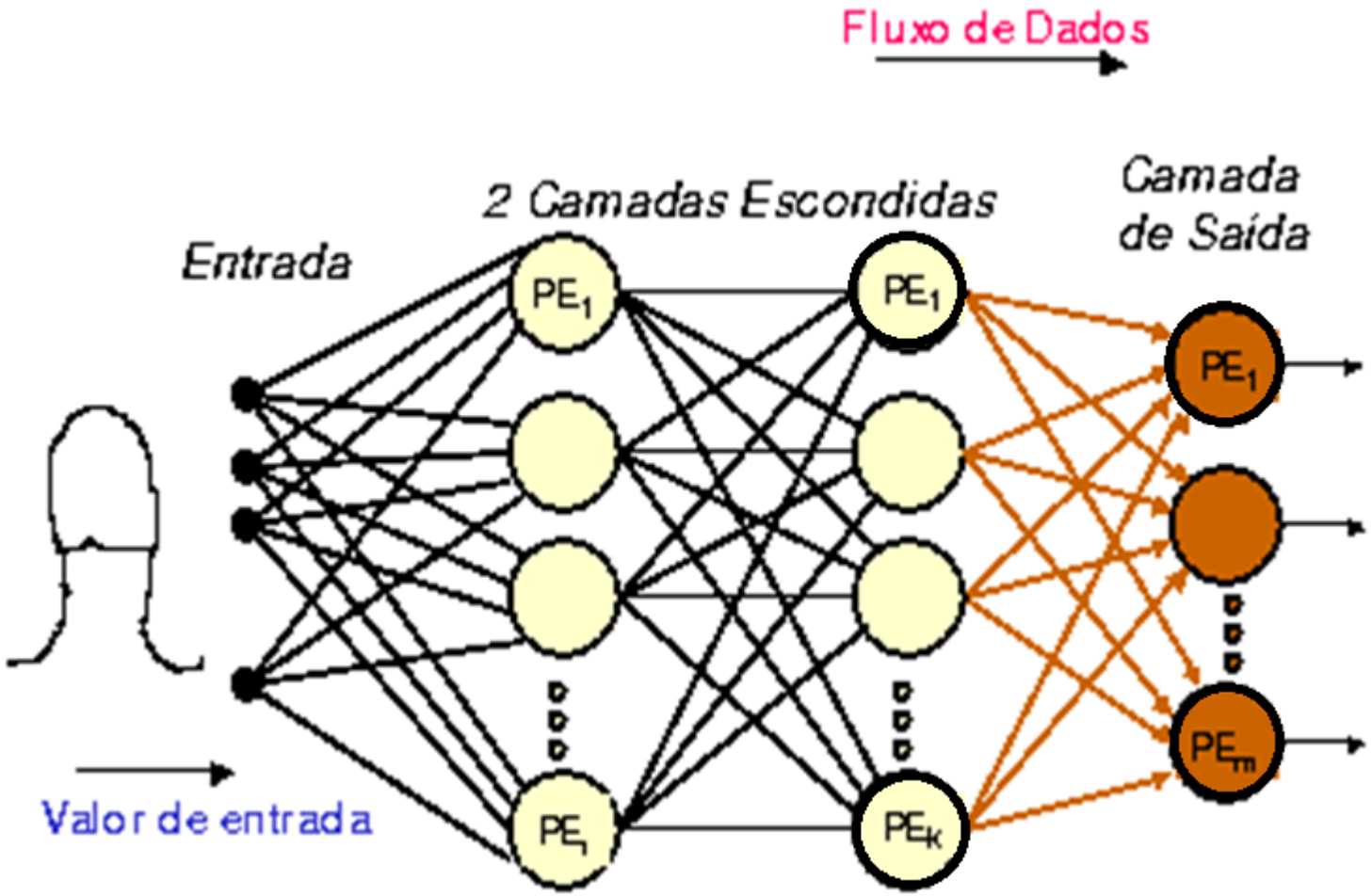


Figura 4: *Feed-forward* (fase 1), camada de saída.

Processo de aprendizado

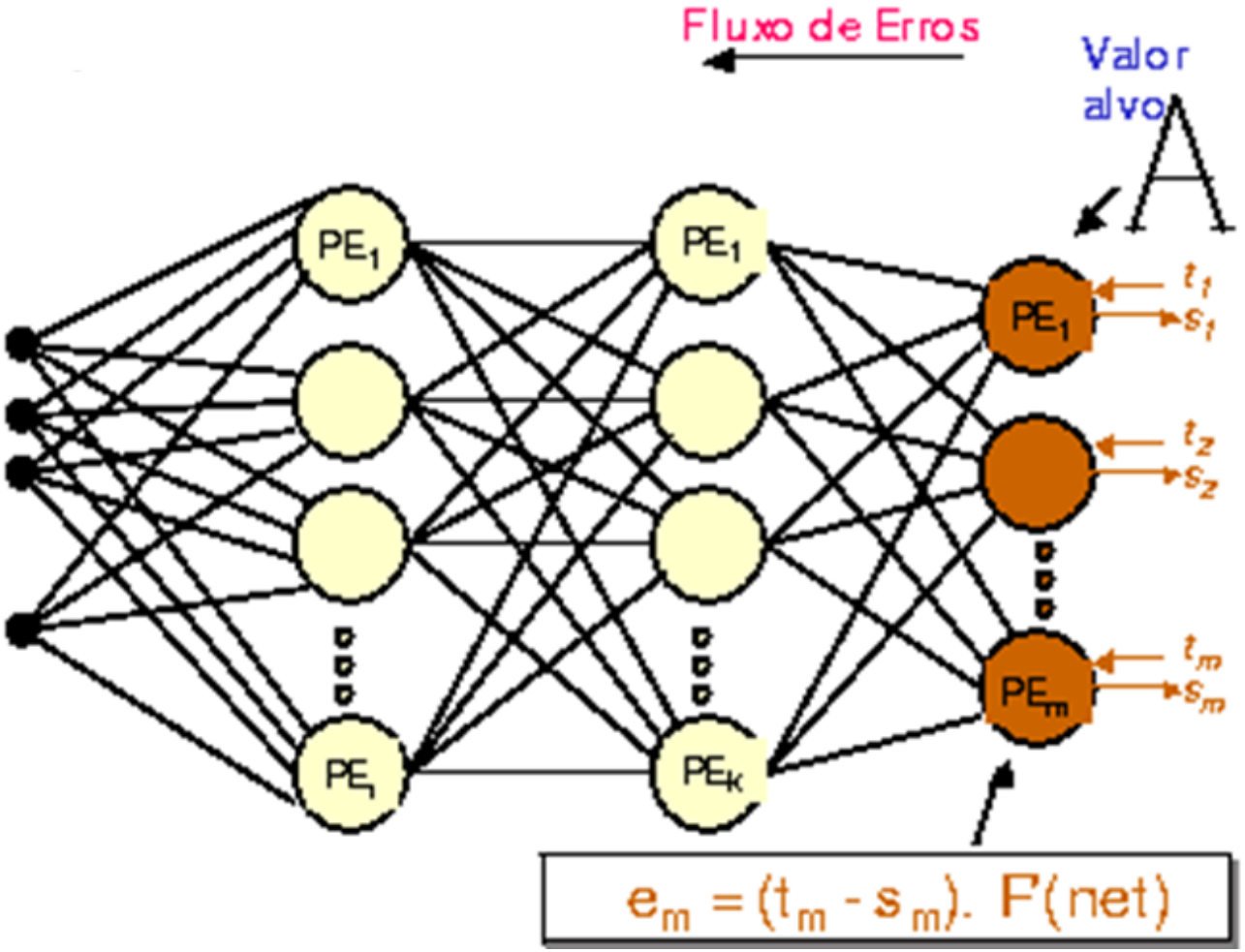


Figura 5: *Feed-backward* (fase 2), cálculo do erro da camada de saída.

Processo de aprendizado

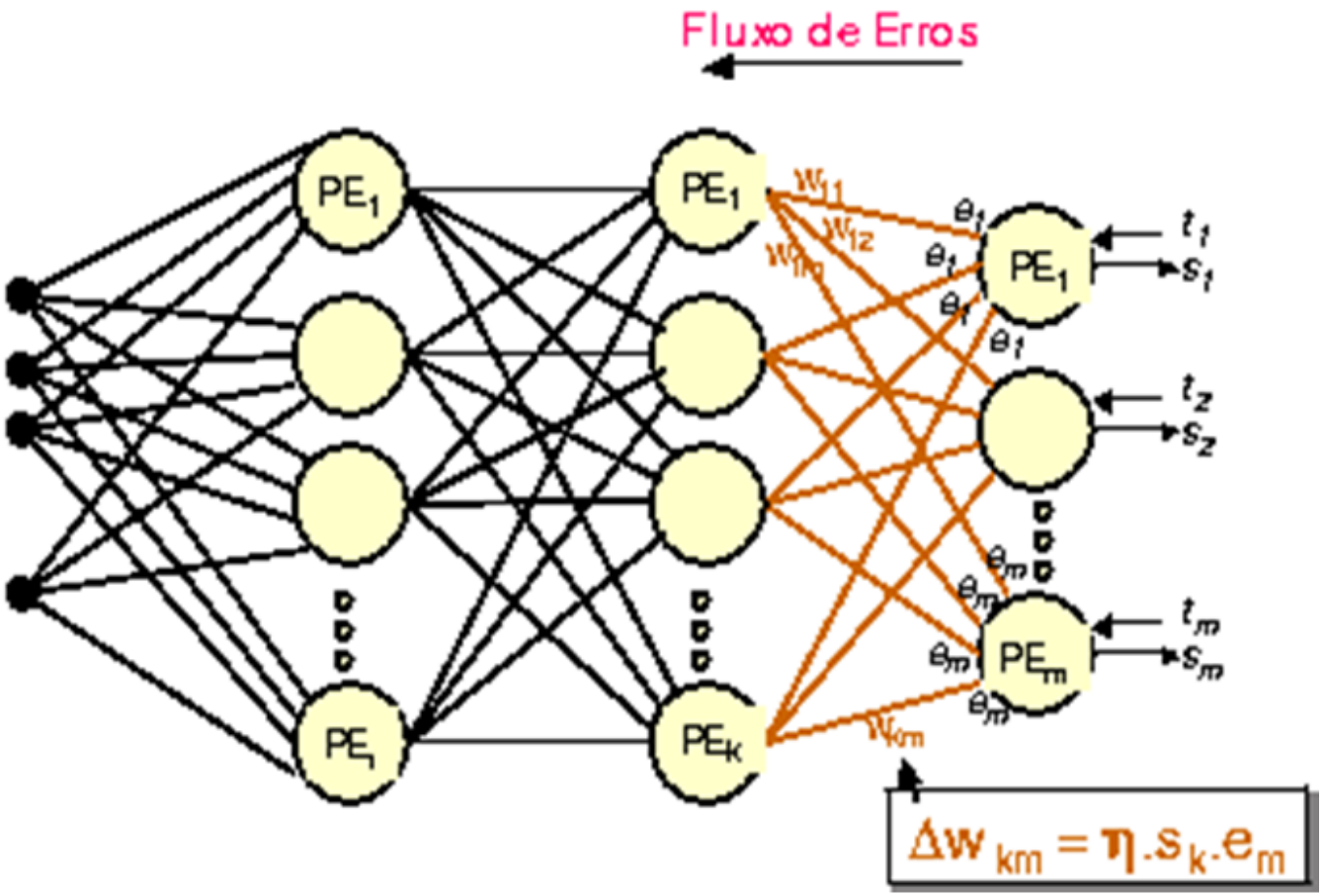


Figura 6: *Feed-backward* (fase 2), atualização dos pesos da camada de saída.

Processo de aprendizado

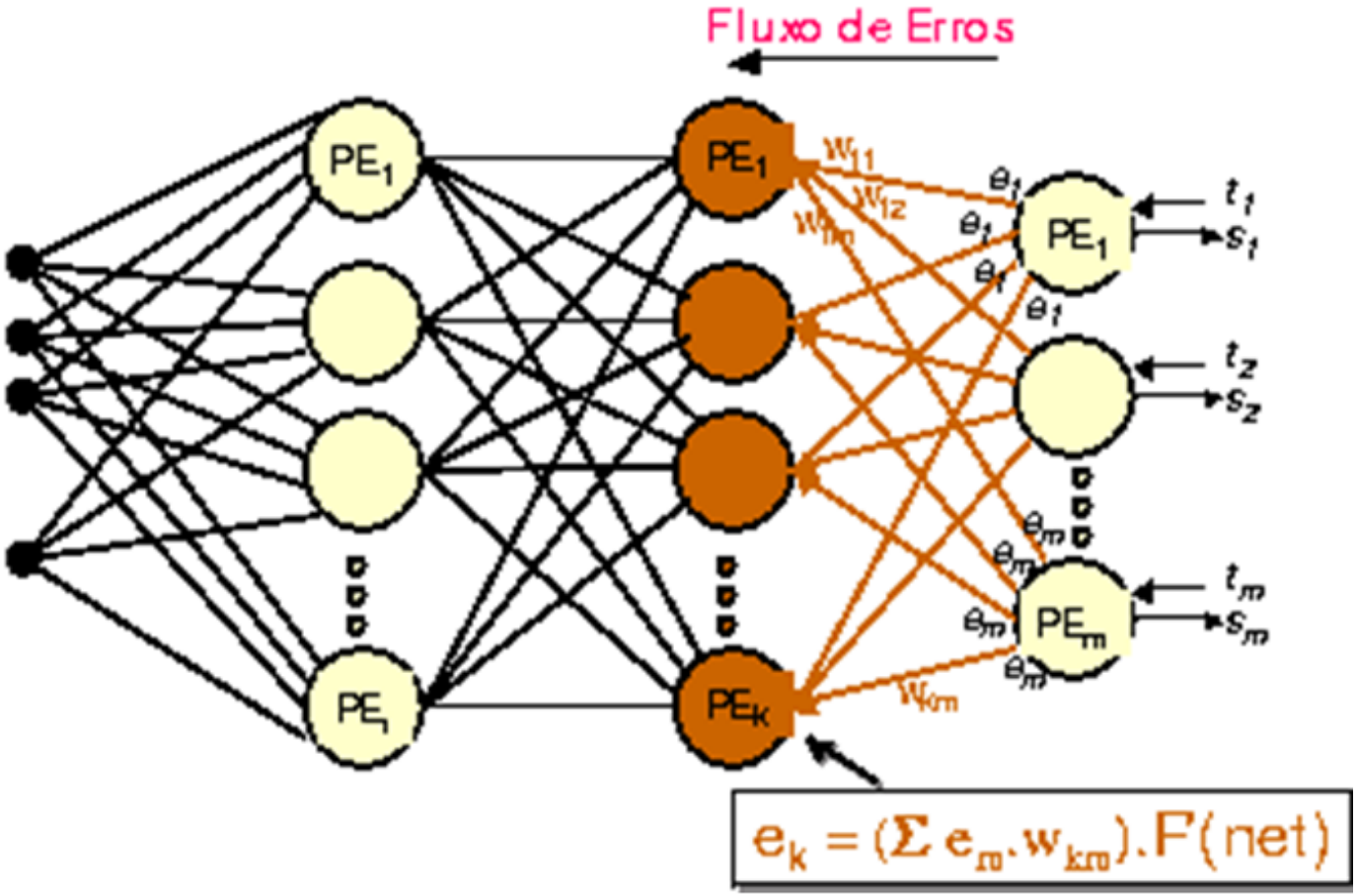


Figura 7: *Feed-backward* (fase 2), cálculo do erro da segunda camada escondida.

Processo de aprendizado

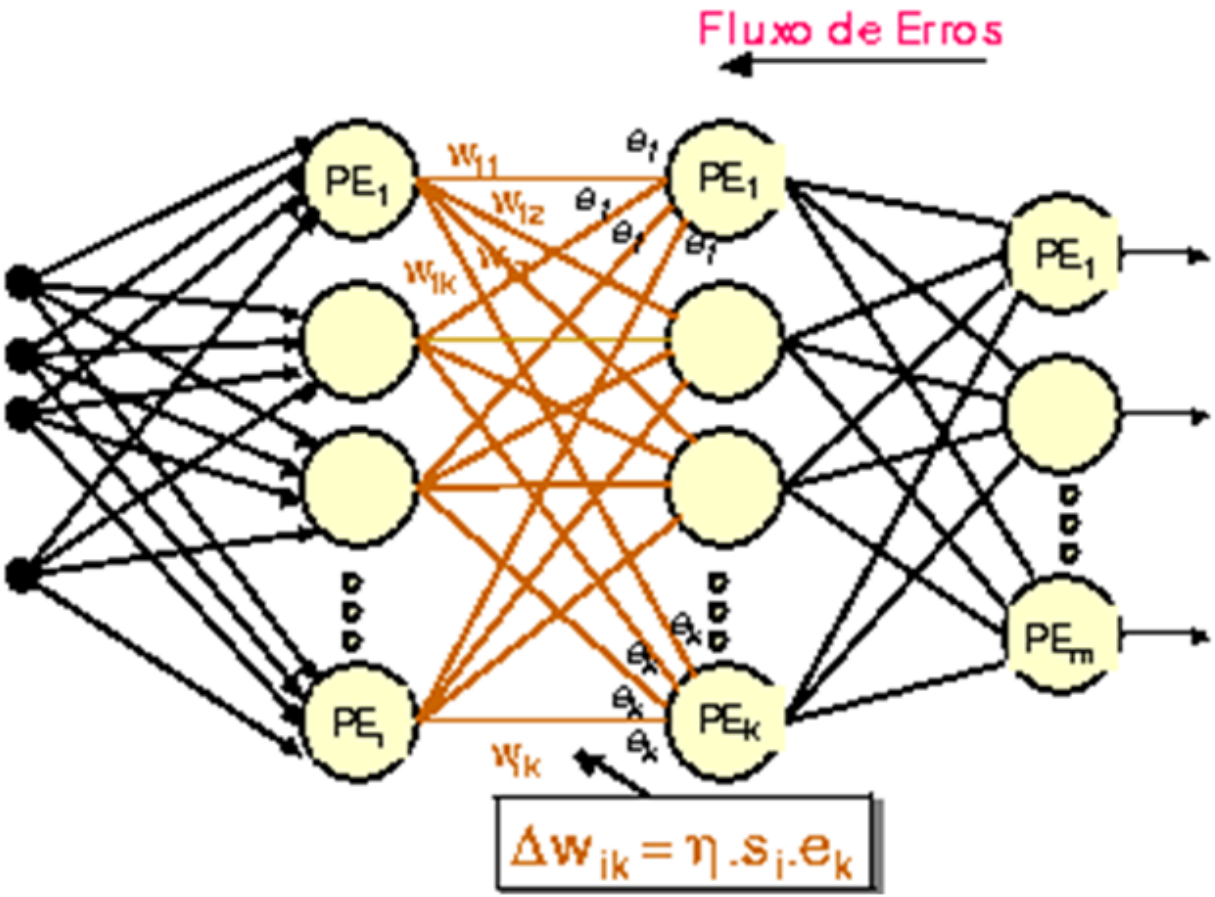


Figura 8: *Feed-backward* (fase 2), atualização dos pesos da segunda camada escondida.

Processo de aprendizado

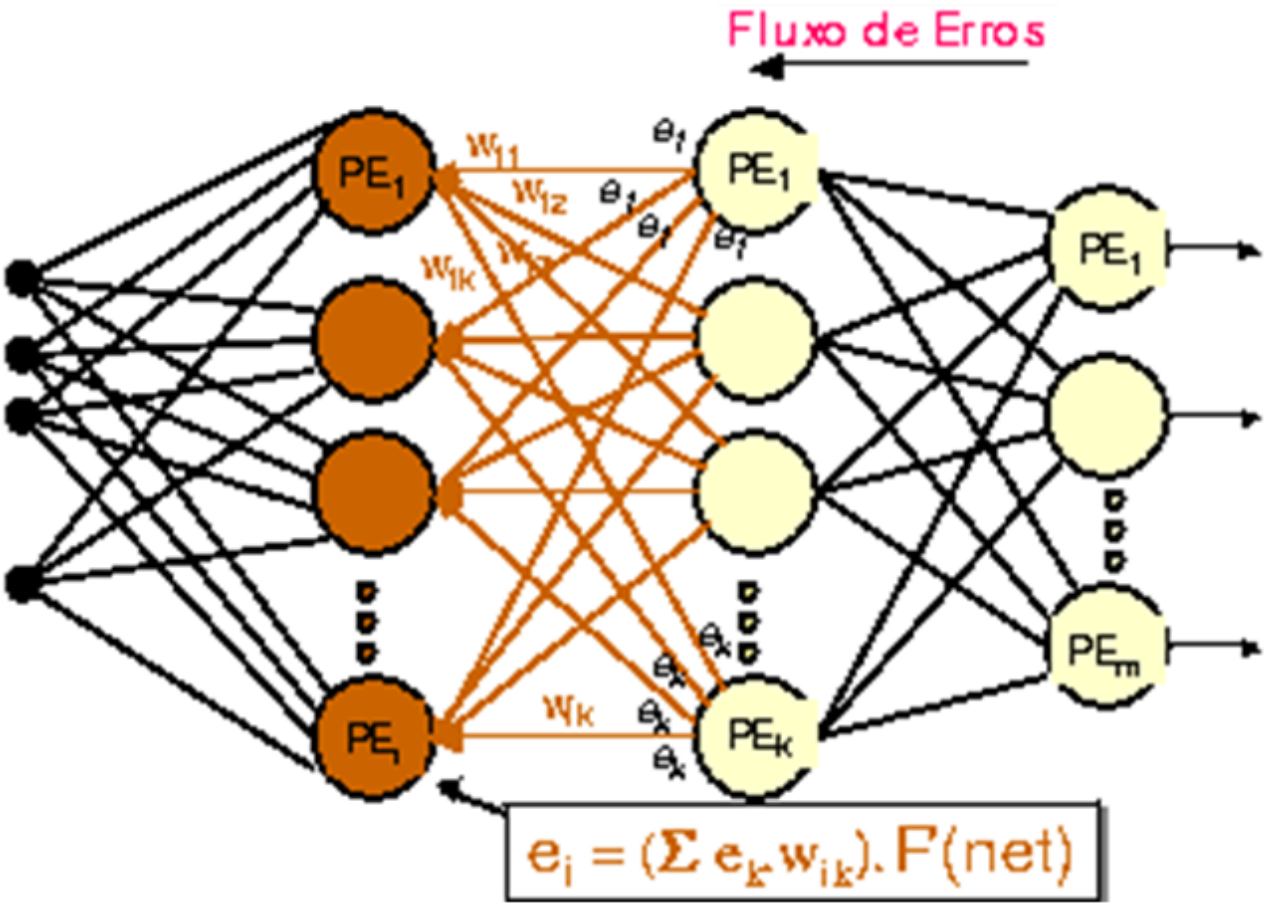


Figura 9: *Feed-backward* (fase 2), cálculo do erro da primeira camada escondida.

Processo de aprendizado

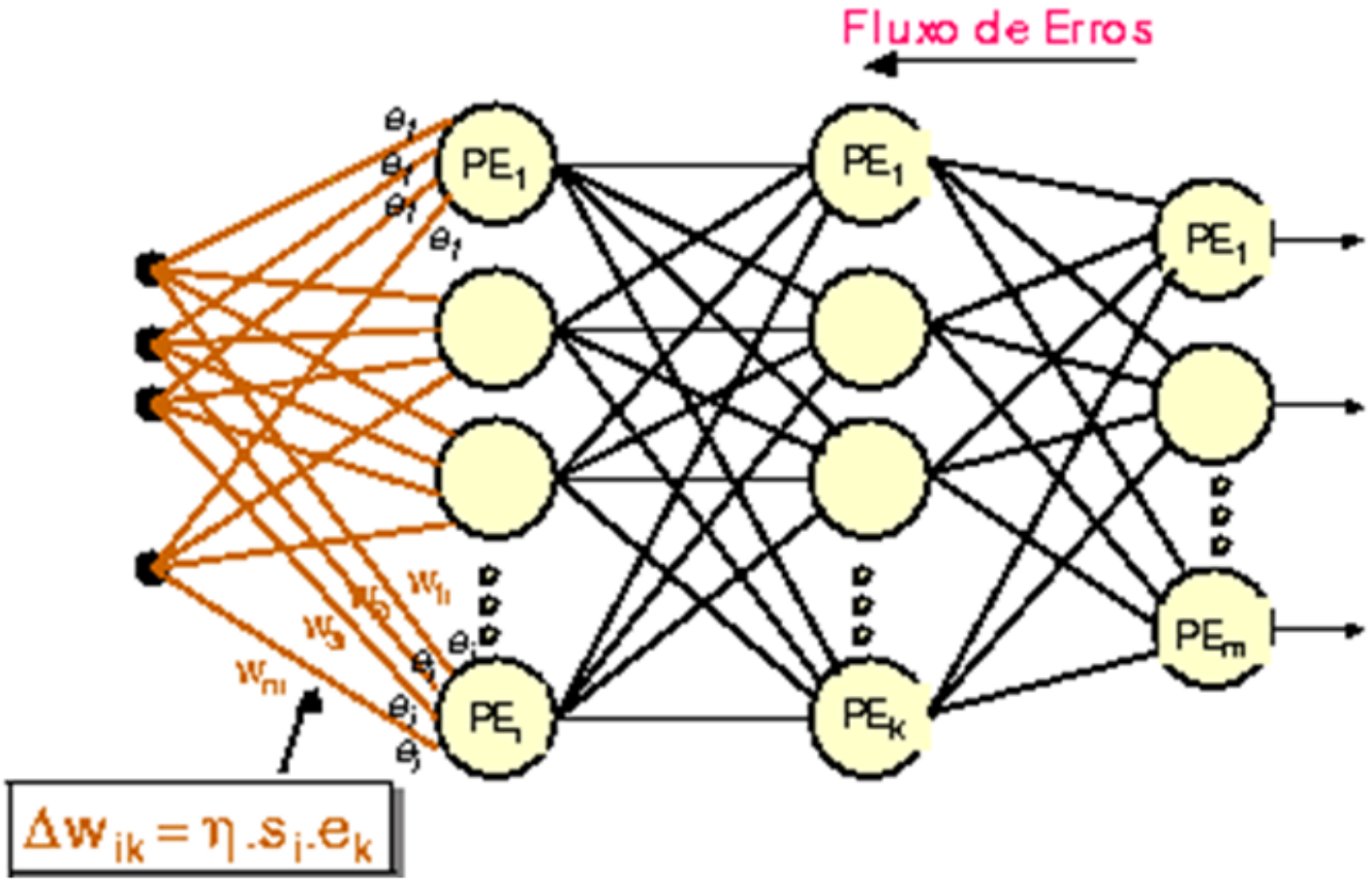


Figura 10: *Feed-backward* (fase 2), atualização dos pesos da primeira camada escondida.

Processo de aprendizado

- Este procedimento de aprendizado é repetido diversas vezes, até que, **para todos processadores de camada de saída e para todos padrões de treinamento**, o erro seja menor do que o especificado.

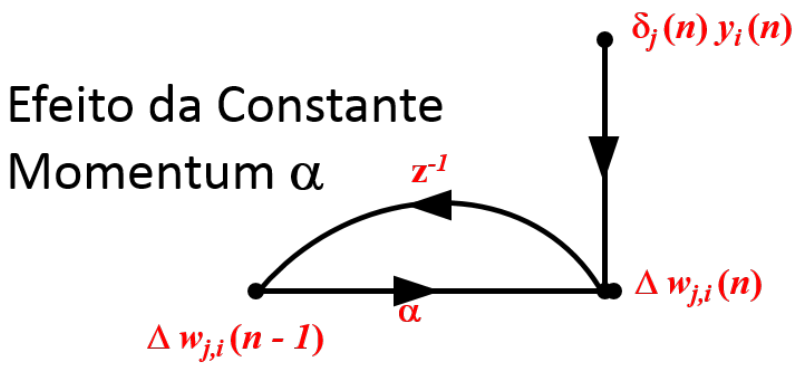
Sumário

Efeito da constante α

- É um método simples de aumentar a velocidade do aprendizado e evitar o perigo de instabilidade, como mostrado por Rumelhart *et al.*, 1986.

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1) \qquad (1)$$

- Onde α é geralmente um número positivo chamado **constante momentum**.



A equação (α) é chamada *REGRA DELTA GENERALIZADA*. Se α =0 $\Rightarrow$ REGRA DELTA

Sumário

Velocidade de aprendizado

- O algoritmo BP fornece uma aproximação para a trajetória no espaço dos pesos.
- Quanto menor o valor de η, menores as mudanças nos pesos e mais suave será a trajetória.
  - **Aprendizado lento.**
- Se η é muito grande, o aprendizado torna-se rápido, porém a rede pode tornar-se **instável**.

Sumário

Modos de treinamento

- Aprendizado BP resulta de muitas apresentações de um conjunto de treinamento de exemplos.
- Uma apresentação **completa** do conjunto de treinamento corresponde a 1 ciclo (*epoch*).
- O processo de aprendizado é repetido ciclo após ciclo, até que os pesos sinápticos e níveis *threshold* se **estabilizem**.
- Tomar os pesos em uma forma aleatória $\rightarrow$ pesquisa no espaço dos pesos **estocástica**.

Modo padrão

- **(1) Modo padrão:**
  - Atualização nos pesos é feita após a apresentação de cada exemplo de treinamento.
  - Um ciclo consistindo de N exemplos de treinamento, arranjados na ordem:

$$\{[\mathbf{x_1}, \mathbf{d_1}], [\mathbf{x_2}, \mathbf{d_2}], \cdots, [\mathbf{x_N}, \mathbf{d_N}]\}$$

  - $[\mathbf{x_1}, \mathbf{d_1}] \rightarrow$ cálculos *forward/backward* e atualização dos pesos.
  - $[\mathbf{x_2}, \mathbf{d_2}] \rightarrow$ cálculos *forward/backward* e atualização dos pesos.

$$\vdots$$

  - $[\mathbf{x_N}, \mathbf{d_N}] \rightarrow$ cálculos *forward/backward* e atualização dos pesos.

Modo padrão

- Dessa forma, a variação média nas mudanças dos pesos é:

$$\Delta\hat{w}_{ji} = \frac{1}{N}\sum_{n=1}^{N}\Delta w_{ji}(n)$$

$$= -\frac{\eta}{N}\sum_{n=1}^{N}\frac{\partial E(n)}{\partial w_{ji}(n)} \implies$$

$$\boxed{\Delta\hat{w}_{ji} = -\frac{\eta}{N}\sum_{n=1}^{N} e_j(n)\frac{\partial e_j(n)}{\partial w_{ji}(n)}} \qquad (2)$$

Modo *batch*

- **(2) Modo batch**:
  - Atualização dos pesos é feita depois da apresentação de todos os exemplos de treinamento que constituem um ciclo.
  - Para um ciclo particular, função custo com o erro quadrático médio:

$$\mathcal{E}_{av} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n) \tag{3}$$

  - Onde C denota o conjunto de índices correspondentes aos neurônios da camada de saída e e_j é o sinal do erro do neurônio j correspondente ao exemplo de treinamento w.

$$\Delta w_{ji} = -\eta \frac{\partial \mathcal{E}_{av}}{\partial w_{ji}} \implies$$

$$\boxed{\Delta w_{ji} = \frac{\eta}{N} \sum_{n=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}} \tag{4}$$

Sumário

Generalização

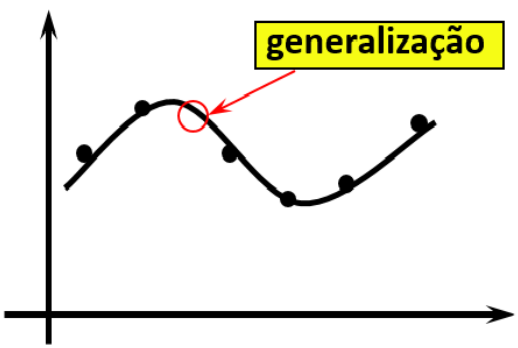
Aprendizado **BP**

conjunto treinamento + algoritmo BP ⇒ pesos sinápticos

GENERALIZAR

"GENERALIZAÇÃO": termo da psicologia.

Processo de aprendizado pode ser visto como um Método de Aproximação de Funcões



: efeito de uma boa aproximação não linear dos dados de entrada, tamanho e eficiência do conjunto treinamento, arquitetura da rede, complexidade física do problema

Complexidade da rede

- **Problema:** determinar o melhor número de nós na camada intermediária.
- Estatisticamente, esse problema é equivalente a determinar o tamanho do conjunto de parâmetros usado para modelar o conjunto de dados. Existe um limite no tamanho da rede.
- Esse limite deve ser tomado lembrando que é melhor treinar a rede para **produzir a melhor generalização** do que treinar a rede para representar perfeitamente um conjunto de dados.
- Isso pode ser feito usando **validação cruzada**.

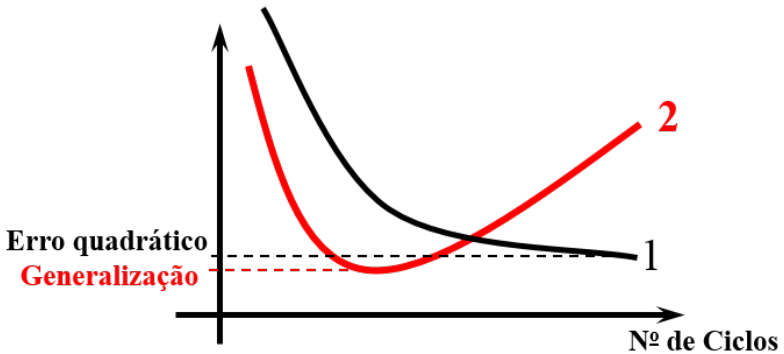
Validação cruzada

- Conjunto de dados:
  - Treinamento (75%)
  - Teste (25%)
- Conjunto de treinamento:
  - Um subconjunto para validação do modelo.
  - Um subconjunto para treinamento.
- Validar o modelo em um conjunto diferente do usado para estimá-lo.

Validação cruzada

- Usa-se o subconjunto de validação para avaliar o desempenho de diferentes candidatos do modelo (diferentes topologias) e, então, escolhe-se uma delas.
- O modelo escolhido é treinado sobre o conjunto de treinamento inteiro e a capacidade de generalização é medida no conjunto de teste.
- A validação cruzada pode ser usada para decidir quando o treinamento de uma rede deve ser encerrado.

Tamanho do conjunto de treinamento



Curva 1: poucos parâmetros (*underfitting*)

Curva 2: muitos parâmetros (*overfitting*)

- Em ambos os casos:
  1. O desempenho do erro na generalização exibe um mínimo.
  2. O mínimo no caso *overfitting* é menor e mais definido.
- Pode-se obter boa generalização se a rede é projetada com muitos neurônios, contanto que o treinamento seja cessado após um número de ciclos correspondente ao mínimo da curva do **erro** obtida na **validação cruzada**.

Sumário

Sumário

Problema

- Em redes neurais *feedforward*, o aprendizado pode tornar-se lento ou pouco efetivo, devido a alguns fatores
- A otimização da função *erro quadrático médio* provoca saturação dos neurônios de saída, tornando o aprendizado mais lento.
- Quando uma entrada x_j é próxima de zero, o peso correspondente w_j se ajusta muito devagar.
- Otimizar a função *cross-entropy*, em vez do erro quadrático médio.
- Usar a função de ativação *softmax* nos neurônios de saída.

Função cross-entropy

- Pode ser a função a ser otimizada, alternativamente ao erro quadrático médio.

$$\text{cross-entropy} = -\sum y' \log y$$

Função Softmax

- A função *softmax* pode ser utilizada como função de ativação na última camada de uma rede neural.
- Seja v_i a combinação linear das entradas e pesos do neurônio de saída i, define-se:

$$softmax(v_i) = \frac{\exp(v_i)}{\sum_j \exp(v_j)}$$

- Aplicando a função *softmax* na camada de saída, obtém-se uma **distribuição de probabilidade** válida, em que a soma das saídas dos neurônios de saída é igual a 1.
  - A saída de cada neurônio representa a probabilidade de a classe correspondente ser a classificação correta.