Sistemas Inteligentes

UNIDADE 5 – Redes Neurais Artificiais
(Perceptron Multicamadas – Conceitos)

Prof. Ivan Nunes da Silva

1. *Rede Perceptron Multicamadas*
Aspectos de arquitetura

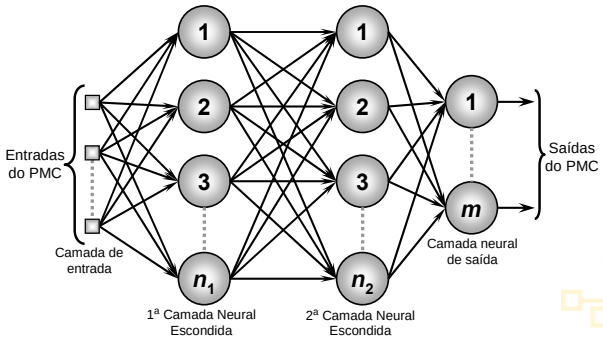
- Redes ***Perceptron de Múltiplas Camadas*** (***PMC***), também conhecidas como redes ***MLP*** (***Multiple Layer Perceptron***), são caracterizadas pela presença de pelo menos uma camada intermediária (escondida) de neurônios.
  - As camadas intermediárias são aquelas situadas entre a camada de entrada e a respectiva camada neural de saída.
  - Conseqüentemente, as redes ***PMC*** possuem no mínimo duas camadas de neurônios, os quais estarão distribuídos entre as camadas intermediárias e a camada de saída.
- Redes ***PMC*** é uma das mais versáteis quanto às suas aplicações, podendo ser utilizadas nos seguintes tipos de problemas:
  - Aproximação universal de funções.
  - Classificação de padrões.
  - Identificação e controle de processos.
  - Previsão de séries temporais.
  - Otimização de sistemas.
- O ***PMC*** pertence à arquitetura *feedforward* de camadas múltiplas.
- O treinamento do ***PMC*** é executado de forma **SUPERVISIONADA**.

2

1. Rede Perceptron Multicamadas

Fluxo de informações

- Síntese do fluxo de informações na estrutura da rede ***PMC***:
  1. Inicia-se na camada de entrada;
  2. Percorre, em seguida, as camadas intermediárias;
  3. Finaliza-se na camada neural de saída.
- No ***PMC*** convencional inexiste qualquer tipo de realimentação de valores produzidos pela camada neural de saída ou pelas próprias camadas neurais intermediárias.

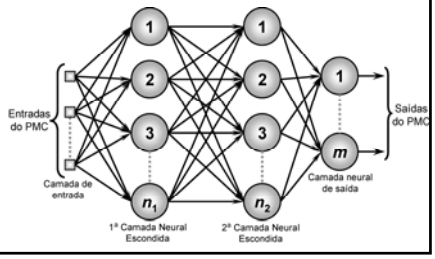


3

1. Rede Perceptron Multicamadas

Princípio de funcionamento

- Síntese do funcionamento da rede ***PMC***:
  1. As entradas do ***PMC***, representando os sinais advindos de determinada aplicação, será propagada camada-a-camada em direção à sua camada neural de saída.
  2. As saídas dos neurônios da primeira camada neural de saída serão as próprias entradas daqueles neurônios pertencentes à segunda camada neural escondida.
  3. As saídas dos neurônios da segunda camada neural escondida serão as respectivas entradas dos neurônios pertencentes à sua camada neural de saída.
- Diferentemente do ***Perceptron*** e ***ADALINE***, além da presença de camadas escondidas, a camada neural de saída do ***PMC*** pode ser composta por diversos neurônios:
  - Cada um destes neurônios de saída representaria uma das saídas do processo a ser mapeado.
  - As camadas intermediárias, por sua vez, extraem a maioria das informações referentes ao seu comportamento e as codificam por meio dos pesos sinápticos e limiares de seus neurônios.
- O projeto de um ***PMC*** depende dos seguintes aspectos:
  - Classe de problema a ser tratado.
  - Disposição espacial das amostras de treinamento.
  - Valores iniciais atribuídos tanto aos parâmetros de treinamento como para as matrizes de pesos.
  - Nível de ruídos presentes nas amostras de treinamento.

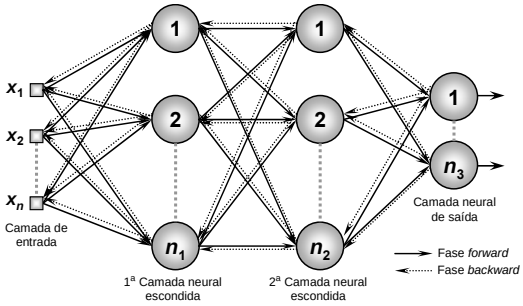


4

2. Processo de Treinamento

Introdução ao algoritmo backpropagation

- O processo de treinamento do ***PMC*** é feito mediante o algoritmo ***backpropagation***, conhecido também como *regra delta generalizada.*
  - O processo é realizado por meio das aplicações sucessivas de **duas fases** bem específicas.
- Como ilustração, considera-se um ***PMC*** constituído de duas camadas escondidas, tendo-se a seguinte composição:
  - n sinais em sua camada de entrada.
  - n_1 neurônios na primeira camada neural escondida.
  - n_2 neurônios na segunda camada neural escondida.
  - n_3 sinais associados à camada neural de saída (terceira camada neural).

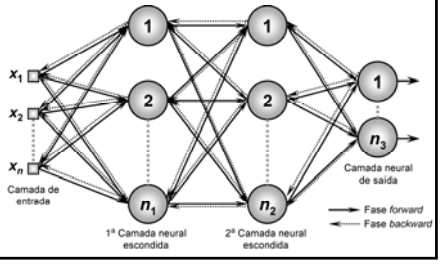


5

2. Processo de Treinamento

Fases do algoritmo backpropagation

- **Primeira Fase ➔ *Forward* (propagação adiante)**
  - Os sinais $\{x_1, x_2, \ldots, x_n\}$ de uma amostra de treinamento são inseridos nas entradas da rede.
  - Estes são propagados camada-a-camada até a produção das respectivas saídas.
  - Leva-se em consideração apenas valores atuais de pesos sinápticos e limiares de seus neurônios, os quais permanecerão inalterados durante cada execução desta fase.
  - **CONCLUSÃO ➔** A aplicação desta fase visa tão somente obter as respostas da rede.
- As respostas produzidas pelas saídas do ***PMC*** são comparadas com as respectivas respostas desejadas (aprendizado supervisionado).
- **Segunda Fase ➔ *Backward* (propagação reversa)**
  - Baseados nos desvios (erros) entre às respostas desejadas e àquelas produzidas pelos neurônios de saída, ajustam-se os pesos e limiares dos neurônio do ***PMC***.
  - **CONCLUSÃO ➔** A aplicação desta fase visa então ajustar pesos e limiares de todos os neurônios.
- Em suma, tem-se:
  - As aplicações sucessivas de ambas as fazem com que os pesos sinápticos e limiares dos neurônios se ajustem automaticamente em cada iteração.
  - Conseqüentemente, ter-se-á então uma gradativa diminuição da soma dos erros produzidos pelas respostas da rede frente àquelas desejadas.
  - O processo cessa quando essa soma dos erros já estiver dentro de valores aceitáveis.

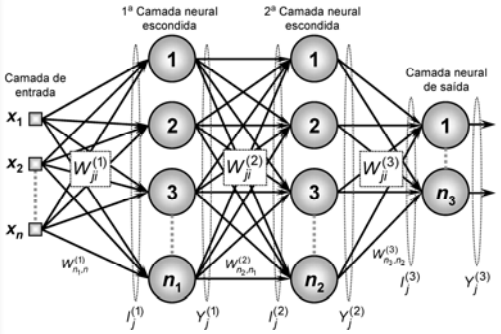


6

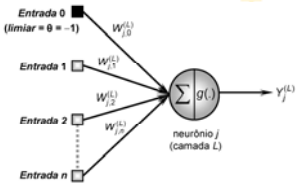
2. Processo de Treinamento

Derivação do algoritmo backpropagation (I)

- **Definindo variáveis e parâmetros (Matrizes de Pesos):**
  - $W_{ji}^{(L)}$ são matrizes de pesos cujos elementos denotam o valor do peso conectando o *j*-ésimo neurônio da camada (*L*) ao *i*-ésimo neurônio da camada (*L*-1). Para a topologia ilustrada, tem-se:
    - $W_{ji}^{(3)}$ é o peso sináptico conectando o *j*-ésimo neurônio da camada de saída ao *i*-ésimo neurônio da camada 2.
    - $W_{ji}^{(2)}$ é o peso sináptico conectando o *j*-ésimo neurônio da camada escondida 2 ao *i*-ésimo neurônio da camada 1.
    - $W_{ji}^{(1)}$ é o peso sináptico conectando o *j*-ésimo neurônio da camada 1 ao *i*-ésimo sinal da camada de entrada.



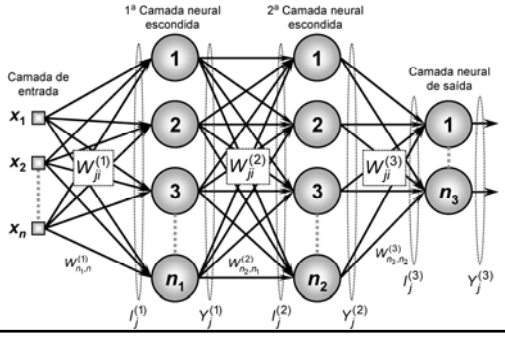
- Cada neurônio acima tem a seguinte configuração:



2. Processo de Treinamento

Derivação do algoritmo backpropagation (II)

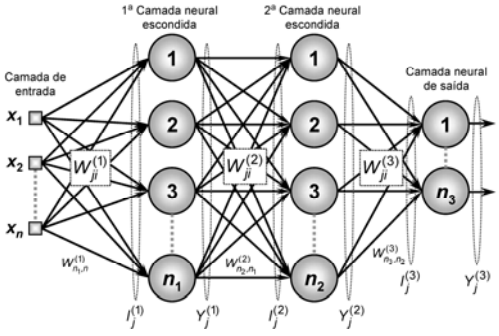
- **Definindo variáveis e parâmetros (Vetores de Entradas):**
  - $I_j^{(L)}$ são vetores cujos elementos denotam a entrada ponderada em relação ao *j*-ésimo neurônio da camada *L*, os quais são definidos por:
    - $I_j^{(1)} = \sum_{i=0}^{n} W_{ji}^{(1)} \cdot x_i \Leftrightarrow I_j^{(1)} = W_{10}^{(1)} \cdot x_0 + W_{11}^{(1)} \cdot x_1 + \cdots + W_{1n}^{(1)} \cdot x_n$ **(1)**
    - $I_j^{(2)} = \sum_{i=0}^{n_1} W_{ji}^{(2)} \cdot Y_i^{(1)} \Leftrightarrow I_j^{(2)} = W_{10}^{(2)} \cdot Y_0^{(1)} + W_{11}^{(2)} \cdot Y_1^{(1)} + \cdots + W_{1n_1}^{(2)} \cdot Y_{n_1}^{(1)}$ **(2)**
    - $I_j^{(3)} = \sum_{i=0}^{n_2} W_{ji}^{(3)} \cdot Y_i^{(2)} \Leftrightarrow I_j^{(3)} = W_{10}^{(3)} \cdot Y_0^{(2)} + W_{11}^{(3)} \cdot Y_1^{(2)} + \cdots + W_{1n_2}^{(3)} \cdot Y_{n_2}^{(2)}$ **(3)**



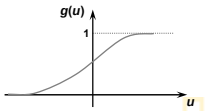
2. Processo de Treinamento

Derivação do algoritmo backpropagation (III)

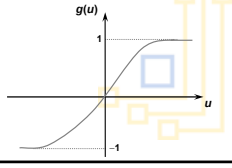
- **Definindo variáveis e parâmetros (Vetores de Saídas):**
  - $Y_j^{(L)}$ são vetores cujos elementos denotam a saída do *j*-ésimo neurônio em relação à camada *L*, os quais são definidos por:
    - $Y_j^{(1)} = g(I_j^{(1)})$ **(4)**
    - $Y_j^{(2)} = g(I_j^{(2)})$ **(5)**
    - $Y_j^{(3)} = g(I_j^{(3)})$ **(6)**
  - O funcional *g*(.) representa uma função de ativação que deve ser contínua e diferenciável em todo o seu domínio, tais como a função de ativação logística ou tangente hiperbólica.



- **Função logística**



- **Função tangente hiperbólica**

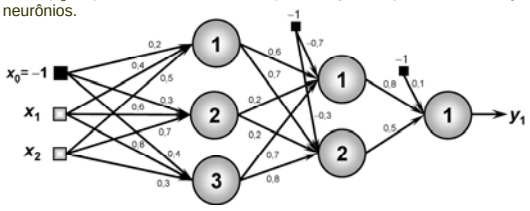


9

2. Processo de Treinamento

Derivação do algoritmo backpropagation (IV)

- **Definindo variáveis e parâmetros (Exemplo):**
  - Considera-se um ***PMC*** composto de duas entradas x_1 e x_2 (n = 2), 3 neurônios na primeira camada escondida (n_1 = 3), 2 neurônios na segunda camada escondida (n_2 = 2) e um neurônio de saída (n_3 = 1). Considera-se também que a tangente hiperbólica é ativação para todos os neurônios.



$$W_{ji}^{(1)} = \begin{bmatrix} 0,2 & 0,4 & 0,5 \\ 0,3 & 0,6 & 0,7 \\ 0,4 & 0,8 & 0,3 \end{bmatrix}$$

$$W_{ji}^{(2)} = \begin{bmatrix} -0,7 & 0,6 & 0,2 & 0,7 \\ -0,3 & 0,7 & 0,2 & 0,8 \end{bmatrix}$$

$$W_{ji}^{(3)} = \begin{bmatrix} 0,1 & 0,8 & 0,5 \end{bmatrix}$$

- **Cálculo de $I_j^{(1)}$ e $Y_j^{(1)}$ para x_1=0,3 e x_2 = 0,7:**

$$I_j^{(1)} = \begin{bmatrix} I_1^{(1)} \\ I_2^{(1)} \\ I_3^{(1)} \end{bmatrix} = \begin{bmatrix} W_{10}^{(1)} \cdot x_0 + W_{11}^{(1)} \cdot x_1 + W_{12}^{(1)} \cdot x_2 \\ W_{20}^{(1)} \cdot x_0 + W_{21}^{(1)} \cdot x_1 + W_{22}^{(1)} \cdot x_2 \\ W_{30}^{(1)} \cdot x_0 + W_{31}^{(1)} \cdot x_1 + W_{32}^{(1)} \cdot x_2 \end{bmatrix} = \begin{bmatrix} 0,2 \cdot (-1) + 0,4 \cdot 0,3 + 0,5 \cdot 0,7 \\ 0,3 \cdot (-1) + 0,6 \cdot 0,3 + 0,7 \cdot 0,7 \\ 0,4 \cdot (-1) + 0,8 \cdot 0,3 + 0,3 \cdot 0,7 \end{bmatrix} = \begin{bmatrix} 0,27 \\ 0,37 \\ 0,05 \end{bmatrix} \quad Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(1)}) \\ g(I_2^{(1)}) \\ g(I_3^{(1)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,27) \\ \tanh(0,37) \\ \tanh(0,05) \end{bmatrix} = \begin{bmatrix} 0,26 \\ 0,35 \\ 0,05 \end{bmatrix} \xrightarrow{Y_0^{(1)} = -1} Y_j^{(1)} = \begin{bmatrix} Y_0^{(1)} \\ Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0,26 \\ 0,35 \\ 0,05 \end{bmatrix}$$

- **Cálculo de $I_j^{(2)}$ e $Y_j^{(2)}$:**

$$I_j^{(2)} = \begin{bmatrix} I_1^{(2)} \\ I_2^{(2)} \end{bmatrix} = \begin{bmatrix} W_{10}^{(2)} \cdot Y_0^{(1)} + W_{11}^{(2)} \cdot Y_1^{(1)} + W_{12}^{(2)} \cdot Y_2^{(1)} + W_{13}^{(2)} \cdot Y_3^{(1)} \\ W_{20}^{(2)} \cdot Y_0^{(1)} + W_{21}^{(2)} \cdot Y_1^{(1)} + W_{22}^{(2)} \cdot Y_2^{(1)} + W_{23}^{(2)} \cdot Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0,96 \\ 0,59 \end{bmatrix} \quad Y_j^{(2)} = \begin{bmatrix} Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(2)}) \\ g(I_2^{(2)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,96) \\ \tanh(0,59) \end{bmatrix} = \begin{bmatrix} 0,74 \\ 0,53 \end{bmatrix} \xrightarrow{Y_0^{(2)} = -1} Y_j^{(2)} = \begin{bmatrix} Y_0^{(2)} \\ Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0,74 \\ 0,53 \end{bmatrix}$$

- **Cálculo de $I_j^{(3)}$ e $Y_j^{(3)}$ (saída da rede):**

$$I_j^{(3)} = \begin{bmatrix} I_1^{(3)} \end{bmatrix} = \begin{bmatrix} W_{10}^{(3)} \cdot Y_0^{(2)} + W_{11}^{(3)} \cdot Y_1^{(2)} + W_{12}^{(3)} \cdot Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0,76 \end{bmatrix} \quad Y_j^{(3)} = \begin{bmatrix} Y_1^{(3)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(3)}) \end{bmatrix} = \begin{bmatrix} \tanh(0,76) \end{bmatrix} = \begin{bmatrix} 0,64 \end{bmatrix}$$

10

2. Processo de Treinamento

Derivação do algoritmo backpropagation (V)

- **Definindo a função representativa dos erros (desvios):**
  - A sua incumbência será medir o desvio entre as respostas produzidas pelos neurônios de saída da rede em relação aos respectivos valores desejados.
  - Considerando a *k*-ésima amostra de treinamento para a topologia ilustrada abaixo, assume-se a função erro quadrático como aquela a ser utilizada para medir o desempenho local associado aos resultados produzidos pelos neurônios de saída frente à referida amostra, ou seja:

$$E(k) = \frac{1}{2}\sum_{j=1}^{n_3}\left(d_j(k) - Y_j^{(3)}(k)\right)^2 \qquad (7)$$

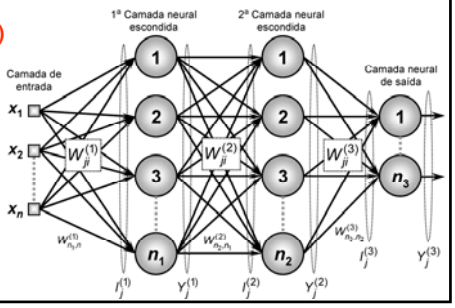
onde $d_j(k)$ é o respectivo valor desejado p/ a *k*-ésima amostra.

  - Conseqüentemente, para um conjunto de treinamento composto por *p* amostras, a evolução do desempenho global do aprendizado pode ser feito por meio da avaliação do "erro quadrático médio", isto é:

$$E_M = \frac{1}{p}\sum_{k=1}^{p} E(k) \qquad (8)$$

onde $E(k)$ é o erro quadrático obtido em (7).

- **Para melhor entendimento, divide-se o algoritmo em duas partes:**
  - **Parte I** ➔ destinada ao ajuste da matriz de pesos sinápticos referente à camada neural de saída.
  - **Parte II** ➔ destinada ao ajuste das matrizes de pesos associadas às camadas intermediárias.



2. Processo de Treinamento

Derivação do algoritmo backpropagation (VI)

- **Parte I ➔ Ajuste da matriz de pesos da camada de saída:**
  - Consiste de ajustar a matriz $W_{ji}^{(3)}$ a fim de minimizar o erro entre a saída da rede frente à saída desejada. Portanto, considerando-se o erro dado em (7), a regra de ajuste se torna similar àquela do ***ADALINE***. Então, pela definição de gradiente e da regra de diferenciação em cadeia, tem-se:

$$\nabla E^{(3)} = \frac{\partial E}{\partial W_{ji}^{(3)}} = \frac{\partial E}{\partial Y_j^{(3)}} \cdot \frac{\partial Y_j^{(3)}}{\partial I_j^{(3)}} \cdot \frac{\partial I_j^{(3)}}{\partial W_{ji}^{(3)}} \qquad (9)$$

  - A partir das definições anteriores, tem-se:

$$\frac{\partial I_j^{(3)}}{\partial W_{ji}^{(3)}} = Y_i^{(2)} \qquad (10) \qquad \frac{\partial Y_j^{(3)}}{\partial I_j^{(3)}} = g'(I_j^{(3)}) \qquad (11) \qquad \frac{\partial E}{\partial Y_j^{(3)}} = -(d_j - Y_j^{(3)}) \qquad (12)$$

  - Substituindo (10), (11) e (12) em (9), obtém-se:

$$\frac{\partial E}{\partial W_{ji}^{(3)}} = -(d_j - Y_j^{(3)}) \cdot g'(I_j^{(3)}) \cdot Y_i^{(2)} \qquad (13)$$

  - Logo, o ajuste de $W_{ji}^{(3)}$ deve ser feito em direção oposta ao gradiente p/ minimizar o erro, ou seja:

$$\Delta W_{ji}^{(3)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{(3)}} \Leftrightarrow \Delta W_{ji}^{(3)} = \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \qquad (14)$$

onde $\delta_j^{(3)}$ o gradiente local em relação ao *j*-ésimo neurônio da camada de saída, isto é:

$$\delta_j^{(3)} = (d_j - Y_j^{(3)}) \cdot g'(I_j^{(3)}) \qquad (15)$$

  - Complementarmente, expressão (15) pode ser convertida no seguinte procedimento iterativo:

$$W_{ji}^{(3)}(t+1) = W_{ji}^{(3)}(t) + \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)} \qquad (16)$$

$$\boxed{W_{ji}^{(3)} \leftarrow W_{ji}^{(3)} + \eta \cdot \delta_j^{(3)} \cdot Y_i^{(2)}} \qquad (17)$$

2. Processo de Treinamento

Derivação do algoritmo backpropagation (VII)

- **Parte II(a) ➔ Ajuste da matriz de pesos da 2ª camada escondida:**
  - Consiste de ajustar a matriz $W_{ji}^{(2)}$ a fim de minimizar o erro. Para tanto, tem-se:

$$\nabla E^{(2)} = \frac{\partial E}{\partial W_{ji}^{(2)}} = \frac{\partial E}{\partial Y_j^{(2)}} \cdot \frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} \cdot \frac{\partial I_j^{(2)}}{\partial W_{ji}^{(2)}} \quad (18)$$

  - A partir das definições anteriores, obtêm-se:

$$\frac{\partial I_j^{(2)}}{\partial W_{ji}^{(2)}} = Y_i^{(1)} \quad (19) \qquad \frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} = g'(I_j^{(2)}) \quad (20) \qquad \frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n_3} \frac{\partial E}{\partial I_k^{(3)}} \cdot \frac{\partial I_k^{(3)}}{\partial Y_j^{(2)}} = \sum_{k=1}^{n_3} \underbrace{\frac{\partial E}{\partial I_k^{(3)}}}_{\text{parcela (i)}} \cdot \underbrace{\frac{\partial (\sum_{k=1}^{n_3} W_{kj}^{(3)} \cdot Y_j^{(2)})}{\partial Y_j^{(2)}}}_{\text{parcela (ii)}} \quad (21)$$

  - O valor da derivada parcial do argumento de (ii) em relação à $Y_j^{(2)}$ é o próprio $W_{kj}^{(3)}$, ou seja:

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{n_3} \underbrace{\frac{\partial E}{\partial I_k^{(3)}}}_{\text{parcela (i)}} \cdot \underbrace{W_{kj}^{(3)}}_{\text{parcela (ii)}} \quad (22) \qquad \frac{\partial E}{\partial Y_j^{(2)}} = -\sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \quad (23)$$

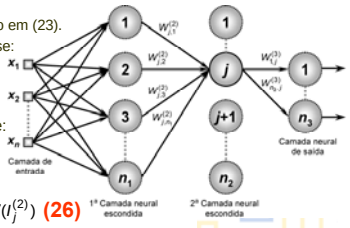
  A parcela (i) foi obtida multiplicando (11) por (12), resultando em (23).
  - Por conseguinte, substituindo (19), (20) e (23) em (18), têm-se:

$$\frac{\partial E}{\partial W_{ji}^{(2)}} = -(\sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)}) \cdot g'(I_j^{(2)}) \cdot Y_i^{(1)} \quad (24)$$

  - Logo, o ajuste de $W_{ji}^{(2)}$ é na direção oposta ao gradiente:

$$\Delta W_{ji}^{(2)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{(2)}} \Leftrightarrow \Delta W_{ji}^{(2)} = \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \quad (25)$$

  onde $\delta_j^{(2)}$ é o gradiente local: $\delta_j^{(2)} = -(\sum_{k=1}^{n_3} \delta_k^{(3)} \cdot W_{kj}^{(3)}) \cdot g'(I_j^{(2)})$ (26)



  - Complementarmente, a expressão (25) poder ser convertida em:

$$W_{ji}^{(2)}(t+1) = W_{ji}^{(2)}(t) + \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \quad (27) \qquad \boxed{W_{ji}^{(2)} \leftarrow W_{ji}^{(2)} + \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)}} \quad (28)$$

2. Processo de Treinamento

Derivação do algoritmo backpropagation (VIII)

- **Parte II(b) ➔ Ajuste da matriz de pesos da 1ª camada escondida:**
  - Consiste de ajustar a matriz $W_{ji}^{(1)}$ a fim de minimizar o erro. Para tanto, tem-se:

$$\nabla E^{(1)} = \frac{\partial E}{\partial W_{ji}^{(1)}} = \frac{\partial E}{\partial Y_j^{(1)}} \cdot \frac{\partial Y_j^{(1)}}{\partial I_j^{(1)}} \cdot \frac{\partial I_j^{(1)}}{\partial W_{ji}^{(1)}} \quad (29)$$

  - A partir das definições anteriores, obtêm-se:

$$\frac{\partial I_j^{(1)}}{\partial W_{ji}^{(1)}} = x_i \quad (30) \qquad \frac{\partial Y_j^{(1)}}{\partial I_j^{(1)}} = g'(I_j^{(1)}) \quad (31) \qquad \frac{\partial E}{\partial Y_j^{(1)}} = \sum_{k=1}^{n_2} \frac{\partial E}{\partial I_k^{(2)}} \cdot \frac{\partial I_k^{(2)}}{\partial Y_j^{(1)}} = \sum_{k=1}^{n_2} \underbrace{\frac{\partial E}{\partial I_k^{(2)}}}_{\text{parcela (i)}} \cdot \underbrace{\frac{\partial (\sum_{k=1}^{n_2} W_{kj}^{(2)} \cdot Y_j^{(1)})}{\partial Y_j^{(1)}}}_{\text{parcela (ii)}} \quad (32)$$

  - O valor da derivada parcial do argumento de (ii) em relação à $Y_j^{(1)}$ é o próprio $W_{kj}^{(2)}$, ou seja:

$$\frac{\partial E}{\partial Y_j^{(1)}} = \sum_{k=1}^{n_2} \underbrace{\frac{\partial E}{\partial I_k^{(2)}}}_{\text{parcela (i)}} \cdot \underbrace{W_{kj}^{(2)}}_{\text{parcela (ii)}} \quad (33) \qquad \frac{\partial E}{\partial Y_j^{(1)}} = -\sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \quad (34)$$

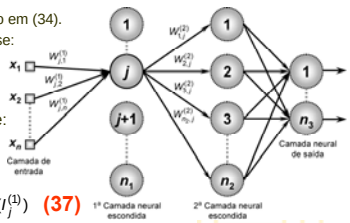
  A parcela (i) foi obtida multiplicando (20) por (21), resultando em (34).
  - Por conseguinte, substituindo (30), (31) e (34) em (29), têm-se:

$$\frac{\partial E}{\partial W_{ji}^{(1)}} = -(\sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)}) \cdot g'(I_j^{(1)}) \cdot x_i \quad (35)$$

  - Logo, o ajuste de $W_{ji}^{(1)}$ é na direção oposta ao gradiente:

$$\Delta W_{ji}^{(1)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{(1)}} \Leftrightarrow \Delta W_{ji}^{(1)} = \eta \cdot \delta_j^{(1)} \cdot x_i \quad (36)$$

  onde $\delta_j^{(1)}$ é o gradiente local: $\delta_j^{(1)} = -(\sum_{k=1}^{n_2} \delta_k^{(2)} \cdot W_{kj}^{(2)}) \cdot g'(I_j^{(1)})$ (37)



  - Complementarmente, a expressão (36) pode ser convertida em:

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} \cdot x_i \quad (38) \qquad \boxed{W_{ji}^{(1)} \leftarrow W_{ji}^{(1)} + \eta \cdot \delta_j^{(1)} \cdot x_i} \quad (39)$$

3. Implementação Computacional

Aspectos de preparação de dados

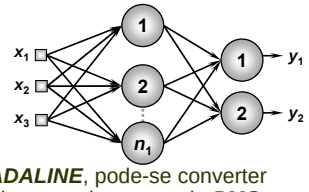
- **Montagem de conjuntos de treinamento:**
  - Supõe-se que um problema a ser mapeado pelo ***PMC*** tenha três entradas $\{x_1, x_2, x_3\}$, e duas saídas $\{y_1, y_2\}$ conforme a figura ao lado (abaixo).
  - Assume-se que se tem quatro amostras, constituída dos seguintes valores de entrada:

Amostra 1 → Entrada: [0,2 0,9 0,4] → Saída desejada: [0,7 0,3]

Amostra 2 → Entrada: [0,1 0,3 0,5] → Saída desejada: [0,6 0,4]

Amostra 3 → Entrada: [0,9 0,7 0,8] → Saída desejada: [0,9 0,5]

Amostra 4 → Entrada: [0,6 0,4 0,3] → Saída desejada: [0,2 0,8]



  - Então, de forma similar ao ***Perceptron*** e ***ADALINE***, pode-se converter tais sinais para que estes possam ser usados no treinamento do ***PMC***:

Conjunto de treinamento

$x^{(1)} = [-1 \quad 0,2 \quad 0,9 \quad 0,4]^T \rightarrow$ com $d^{(1)} = [0,7 \quad 0,3]^T$

$x^{(2)} = [-1 \quad 0,1 \quad 0,3 \quad 0,5]^T \rightarrow$ com $d^{(2)} = [0,6 \quad 0,4]^T$

$x^{(3)} = [-1 \quad 0,9 \quad 0,7 \quad 0,8]^T \rightarrow$ com $d^{(3)} = [0,9 \quad 0,5]^T$

$x^{(4)} = [-1 \quad 0,6 \quad 0,4 \quad 0,3]^T \rightarrow$ com $d^{(4)} = [0,2 \quad 0,8]^T$

forma matricial

$$\Omega^{(x)} = \begin{matrix} & x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} \\ x_0 & -1 & -1 & -1 & -1 \\ x_1 & 0,2 & 0,1 & 0,9 & 0,6 \\ x_2 & 0,9 & 0,3 & 0,7 & 0,4 \\ x_3 & 0,4 & 0,5 & 0,8 & 0,3 \end{matrix}$$

$$\Omega^{(d)} = \begin{matrix} & d^{(1)} & d^{(2)} & d^{(3)} & d^{(4)} \\ d_1 & 0,7 & 0,6 & 0,9 & 0,2 \\ d_2 & 0,3 & 0,4 & 0,5 & 0,8 \end{matrix}$$

  - Geralmente, as amostras de treinamento são disponibilizadas em sua forma matricial (por meio de arquivo texto ou planilha).

15

3. Implementação Computacional

Algoritmo de aprendizagem (fase de treinamento)

- **Pseudocódigo para fase de treinamento:**

Início {Algoritmo PMC – Fase de Treinamento}

<1> Obter o conjunto de amostras de treinamento $\{x^{(k)}\}$;
<2> Associar o vetor de saída desejada $\{d^{(k)}\}$ para cada amostra;
<3> Iniciar $w_{ij}^{(1)}$, $w_{ji}^{(2)}$ e $w_{ji}^{(3)}$ com valores aleatórios pequenos;
<4> Especificar taxa de aprendizagem $\{\eta\}$ e precisão requerida $\{\varepsilon\}$;
<5> Iniciar o contador de número de épocas $\{época \leftarrow 0\}$;
<6> Repetir as instruções:
  <6.1> $E_M^{anterior} \leftarrow E_M$; {conforme (8)}
  <6.2> Para todas as amostras de treinamento $\{x^{(k)}, d^{(k)}\}$, fazer:
    <6.2.1> Obter $I_j^{(1)}$ e $Y_j^{(1)}$; {conforme (1) e (4)}
    <6.2.2> Obter $I_j^{(2)}$ e $Y_j^{(2)}$; {conforme (2) e (5)}
    <6.2.3> Obter $I_j^{(3)}$ e $Y_j^{(3)}$; {conforme (3) e (6)}
    } Passo *Forward*
    <6.2.4> Determinar $\delta_j^{(3)}$; {conforme (15)}
    <6.2.5> Ajustar $w_{ji}^{(3)}$; {conforme (17)}
    <6.2.6> Determinar $\delta_j^{(2)}$; {conforme (26)}
    <6.2.7> Ajustar $w_{ji}^{(2)}$; {conforme (28)}
    <6.2.8> Determinar $\delta_j^{(1)}$; {conforme (37)}
    <6.2.9> Ajustar $w_{ji}^{(1)}$; {conforme (39)}
    } Passo *Backward*
  <6.3> Obter $Y_j^{(3)}$ ajustado; {conforme <6.2.1>, <6.2.2> e <6.2.3>}
  <6.4> $E_M^{atual} \leftarrow E_M$; {conforme (8)}
  <6.5> $época \leftarrow época + 1$;
  Até que: $|E_M^{atual} - E_M^{anterior}| \leq \varepsilon$

Fim {Algoritmo PMC – Fase de Treinamento}

16

3. Implementação Computacional

Algoritmo de aprendizagem (fase de operação)

- **Pseudocódigo para fase de operação:**

Início {Algoritmo PMC – Fase de Operação}

<1> Obter uma amostra { $\boldsymbol{x}$ };
<2> Assumir $W_{ji}^{(1)}$, $W_{ji}^{(2)}$ e $W_{ji}^{(3)}$ já ajustadas no treinamento;
<3> Execute as seguintes instruções:
  <3.1> Obter $I_j^{(1)}$ e $Y_j^{(1)}$; {conforme (1) e (4)}
  <3.2> Obter $I_j^{(2)}$ e $Y_j^{(2)}$; {conforme (2) e (5)}
  <3.3> Obter $I_j^{(3)}$ e $Y_j^{(3)}$; {conforme (3) e (6)}
  } Passo *Forward*
<4> Disponibilizar as saídas da rede, as quais são dadas pelos elementos contidos em $Y_j^{(3)}$

Fim {Algoritmo PMC – Fase de Operação}

Obs. 1 ➔ A "fase de operação" é usada somente após a "fase de treinamento", pois aqui a rede já está apta para ser usada no processo.

Obs. 2 ➔ Lembrar de incluir o valor -1 dentro do vetor $\boldsymbol{x}$.

$$\boldsymbol{x} = [-1 \quad x_1 \quad x_2 \ldots x_n]^T$$

17

4. Técnicas de Validação Cruzada

Conceitos introdutórios

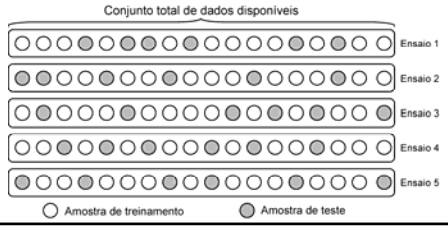
- **Aspectos de seleção topológica de redes *PMC*:**
  - A especificação da topologia de rede ***PMC*** mais apropriada para mapear um problema específico é usualmente efetuada de forma empírica, pois tal dimensionamento depende (entre outros) dos seguintes fatores:
    - Algoritmo de aprendizado utilizado.
    - Maneira como as matrizes de pesos foram iniciadas.
    - Complexidade do problema a ser mapeado.
    - Disposição espacial das amostras.
    - Qualidade do conjunto de treinamento disponível (relacionado aos níveis de ruídos presentes nas amostras).
  - Como exemplo ilustrativo, considera-se que para um determinado problema se tem 4 topologias candidatas de ***PMC***, constituídas todas de apenas uma camada escondida, e que podem ser capazes de mapear o seu comportamento. São elas as seguintes:
    - Topologia Candidata 1 ➔ 05 neurônios na camada escondida.
    - Topologia Candidata 2 ➔ 10 neurônios na camada escondida.
    - Topologia Candidata 3 ➔ 15 neurônios na camada escondida.
    - Topologia Candidata 4 ➔ 20 neurônios na camada escondida.
  - **O objetivo agora colocado está em saber qual delas seria a mais indicada para executar o mapeamento do referido problema.**

18

4. Técnicas de Validação Cruzada

Validação cruzada por amostragem aleatória

- **Princípios da validação cruzada (amostragem aleatória):**
  - O conjunto total de dados (amostras) disponíveis é aleatoriamente dividido em duas partes, isto é, subconjunto de treinamento e subconjunto de teste (validação).
    - **Subconjunto de treinamento** ➔ utilizado para treinar todas as topologias candidatas.
    - **Subconjunto de teste** ➔ utilizado para selecionar aquela que estará apresentando os melhores resultados de generalização.
  - As amostras do subconjunto de teste não participaram do treinamento, o que possibilita avaliar o desempenho da generalização proporcionada em cada uma das topologias candidatas.
    - Para tanto, basta-se comparar os resultados produzidos em suas saídas frente aos respectivos valores desejados.
  - A partir do conjunto total de amostras, cerca de 60 a 90% delas são aleatoriamente escolhidas para o subconjunto de treinamento, enquanto o restante ficará alocado ao subconjunto de teste.
  - Esta sistemática de partição é repetida várias vezes durante o aprendizado das topologias candidatas, permitindo-se (em cada ensaio) a possibilidade de contemplação de amostras diferentes tanto no subconjunto de treinamento como naquele de teste.
  - O desempenho global de cada topologia candidata será então compilado a partir da média dos desempenhos individuais em cada experimento.



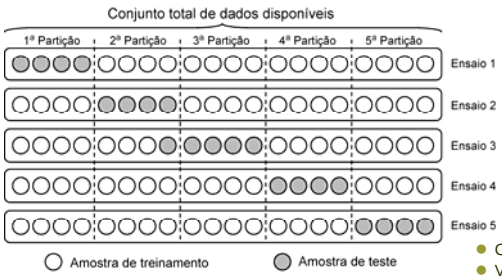
- Conjunto total de amostras ➔ 18
- Conjunto de treinamento ➔ 12
- Conjunto de teste ➔ 6

19

4. Técnicas de Validação Cruzada

Validação cruzada por k-partições

- **Princípios da validação cruzada (*k*-partições):**
  - Realiza-se aqui a divisão do conjunto total de amostras em *k* partições, sendo que (*k*-1) delas serão usadas para compor o subconjunto de treinamento, ao passo que a partição restante constituirá o subconjunto de teste.
  - Por conseguinte, o processo de aprendizado se repete *k* vezes até que todas as partições tenham sido utilizadas como subconjunto de teste.
  - O valor do parâmetro *k* está atrelado à quantidade total de amostras disponíveis, sendo usualmente atribuído um número compreendido entre 5 e 10.
  - O desempenho global de cada topologia candidata será agora também obtido em função da média entre os desempenhos individuais observados quando da aplicação das *k* partições.



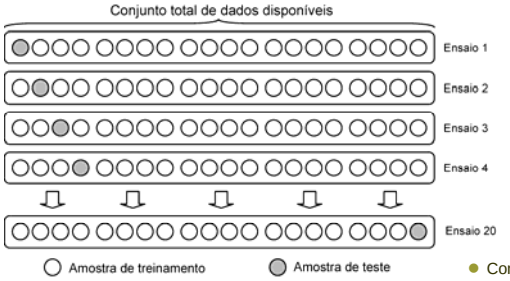
- Conjunto total de amostras ➔ 20
- Valor do parâmetro *k* ➔ 5

20

4. Técnicas de Validação Cruzada

Validação cruzada por unidade

- **Princípios da validação cruzada (por unidade):**
  - Consiste da utilização de uma única amostra para o subconjunto de teste, sendo todas as demais alocadas para o subconjunto de treinamento.
  - O processo de aprendizado é então repetido até que todas as amostras sejam individualmente utilizadas como subconjunto de teste.
  - Esta técnica acaba sendo um caso particular do método de *k*-partições, pois se basta atribuir ao parâmetro *k* o valor que corresponde ao número total de amostras disponíveis.
  - Contudo, tem-se aqui um elevado esforço computacional, pois o processo de aprendizagem será repetido, considerando cada uma das topologias candidatas, um número de vezes que será igual ao tamanho do conjunto total de amostras.



- Conjunto total de amostras ➔ 20

21

4. Técnicas de Validação Cruzada

Aspectos de implementação

- **Pseudocódigo para efetuar validação cruzada:**

Início {Algoritmo Validação Cruzada}

<1> Definir para o problema as topologias candidatas de PMC;
<2> Disponibilizar os subconjuntos de treinamento e de teste;
<3> Aplicar o algoritmo de treinamento do PMC para todas as topologias candidatas usando os subconjuntos de treinamento;
<4> Aplicar os subconjuntos de teste nas topologias candidatas (já treinadas) visando avaliar os potenciais de generalização;
<5> Obter o desempenho final de cada topologia candidata em função do número de ensaios utilizados;
<6> Selecionar aquela topologia candidata que obteve o melhor desempenho global;
<7> Se o desempenho global desta melhor topologia candidata estiver de acordo com a precisão requerida ao problema,
  <7.1> Então: Terminar o processo de validação cruzada.
  <7.2> Senão: Especificar novo conjunto de topologias candidatas e voltar ao passo <3>.

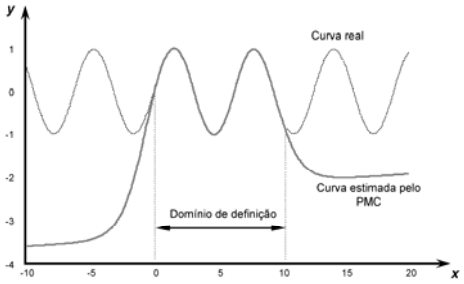
Fim {Algoritmo Validação Cruzada}

22

4. Técnicas de Validação Cruzada

Aspectos de subconjuntos de treinamento e teste

- **Alocação de amostras nos subconjuntos de treinamento:**
  - Assegurar que todas as amostras, que carregam os valores mínimos e máximos de cada variável de entrada, estejam também dentro desses subconjuntos.
    - Caso contrário, se tais valores forem inadvertidamente alocados aos subconjuntos de teste, o ***PMC*** poderia então gerar erros significativos, pois tentaria generalizar valores que estão fora dos domínios de definição de suas variáveis de entrada (nos quais foi treinado).
  - Durante toda a fase de operação, deve-se ainda garantir que os atuais sinais, referentes a cada uma das variáveis de entrada, estejam novamente compreendidos dentro daqueles domínios de definição que foram obtidos a partir dos valores mínimos e máximos dos subconjuntos de treinamento.
    - Realiza-se um procedimento de pré-checagem a fim de verificar se os sinais estão dentro dos domínios de definição.

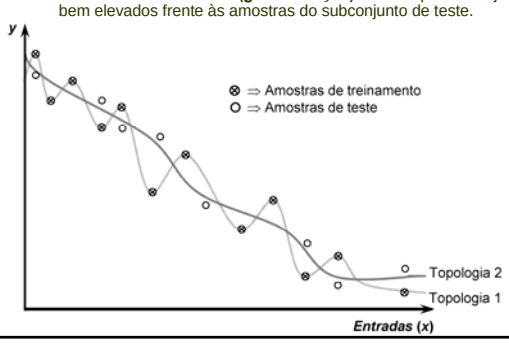


- PMC treinado para mapear a função seno.
- Amostras de treinamento estavam compreendidas no domínio entre 0 e 10.
- As respostas da rede fora do domínio são totalmente incompatíveis.

4. Técnicas de Validação Cruzada

Aspectos de situações de overfitting/underfitting (I)

- **Aspectos de ocorrência de *overfitting (sobre-treinamento)*:**
  - O aumento indiscriminado de neurônios, assim como de camadas intermediárias, não assegura a generalização apropriada do ***PMC*** frente às amostras pertencentes aos subconjuntos de teste.
  - Esse aumento indiscriminado tende a levar a saída do ***PMC*** para a circunstância de memorização excessiva (***overfitting***), em que o mesmo acaba decorando as suas respostas frente aos estímulos introduzidos em suas entradas. Aqui, verifica-se os seguintes aspectos:
    - **Durante a fase de aprendizado** ➔ Erro quadrático tende a ser bem baixo.
    - **Durante a fase de teste (generalização)** ➔ Erro quadrático já tende a assumir valores bem elevados frente às amostras do subconjunto de teste.



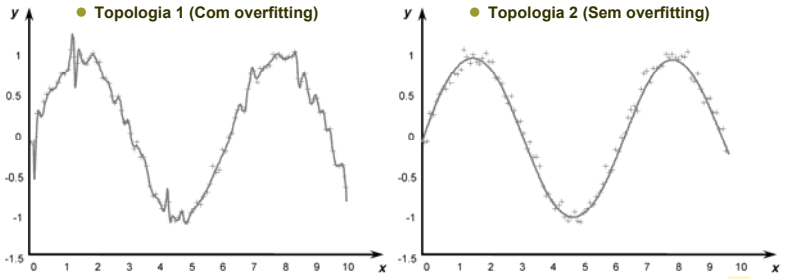
Topologia 1 (Com overfitting)
- Composta de uma camada escondida.
- 20 neurônios nesta camada.
- Apresenta menor Erro frente às amostras de treinamento.
- Apresenta maior Erro frente às amostras de teste.

Topologia 2 (Sem overfitting)
- Composta de uma camada escondida.
- 10 neurônios nesta camada.
- Apresenta maior Erro frente às amostras de treinamento.
- Apresenta menor Erro frente às amostras de teste.

4. Técnicas de Validação Cruzada

Aspectos de situações de overfitting/underfitting (II)

- **Ilustração de ocorrência de *overfitting (sobre-treinamento)*:**
  - Mapeamento da função seno (que foi afetada por ruídos).



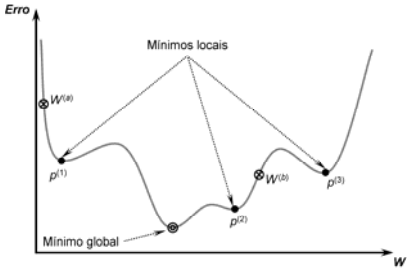
- **Aspectos de ocorrência de *underfitting (sub-treinamento)*:**
  - Em contrapartida, frente à precisão requerida, uma topologia de ***PMC*** com número muito reduzido de neurônios pode ser insuficiente para a extração e armazenamento de características que permitam à rede implementar as hipóteses a respeito do comportamento do processo.
  - Nesses casos, por sua vez, o erro quadrático tanto na fase de aprendizado como na fase de teste serão bem significativos.

25

4. Técnicas de Validação Cruzada

Aspectos de convergência para mínimos locais

- **Superfície da função erro quadrático e mínimos locais:**
  - Como a superfície de erro produzida pelo ***PMC*** é não-linear, há a possibilidade de que o treinamento leve a matriz de pesos da rede p/ um ponto de mínimo local.
  - Este ponto pode não corresponder aos valores mais apropriados aos propósitos de generalização de resultados.
  - Esta tendência de convergência fica condicionada à posição em que ***W*** foi iniciada, pois o treinamento das redes é baseado em métodos de gradiente descendente



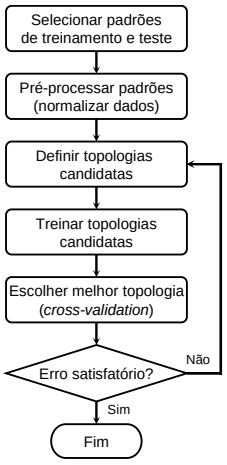
  - No exemplo, se a rede for iniciada em $\boldsymbol{W}^{(a)}$, a tendência é convergir p/ o ponto de mínimo $p^{(1)}$; ao passo que se for iniciada em $\boldsymbol{W}^{(b)}$, a propensão seria p/ o ponto $p^{(2)}$.
  - Entretanto, a solução dada por $p^{(2)}$ é mais favorável que aquela dada por $p^{(1)}$, pois o valor do erro para $p^{(2)}$ é menor que aquele de $p^{(1)}$.
  - **CONCLUSÃO ➔** Uma forma para contornar o problemas de mínimos locais seria executar o treinamento da topologia várias vezes, a fim de selecionar o melhor deles.

26

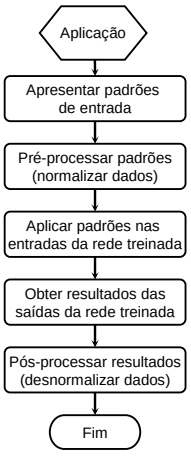
5. Aspectos de Projeto de PMC

Principais etapas de projeto

- **Diagrama de blocos para projeto de redes *PMC*:**



(a) Fase de Treinamento

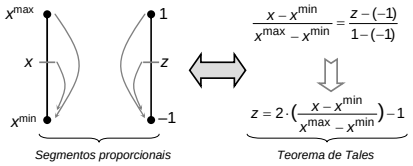


(b) Fase de Operação

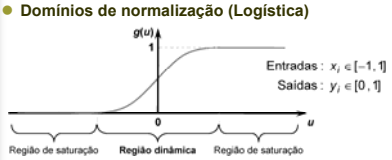
5. Aspectos de Projeto de PMC

Pré-processamento de dados (normalização)

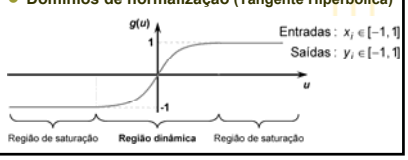
- **Princípios de normalização de dados:**
  - Conforme observado no slide anterior, há a necessidade de pré-processamento dos padrões de treinamento/teste visando aspectos de melhoria do desempenho de treinamento.
  - Isto implica geralmente em escalar as respectivas amostras p/ a faixa de variação dinâmica das funções de ativação dos neurônios, evitando-se assim a saturação de suas saídas.
  - Uma das técnicas de escalamento mais utilizada é aquela baseada no princípio dos segmentos proporcionais (Teorema de Tales) ilustrado na figura seguinte, isto é:
    - **Antes de Normalizar →** valores inicialmente compreendidos entre a faixa delimitada por x^{min} e x^{max}, ou seja, $x \in [x^{min}, x^{max}]$.
    - **Depois de Normalizar →** valores estarão convertidos para um domínio proporcional entre –1 e 1, o qual representa as faixas de variações dinâmicas das funções de ativação.



- **Domínios de normalização (Logística)**



- **Domínios de normalização (Tangente Hiperbólica)**



6. Aplicabilidade do PMC

Problemas de aproximação funcional

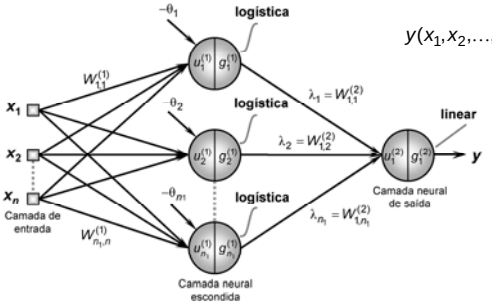
- **Caracterização de problemas de aproximação funcional:**
  - É a classe de problemas em que as redes ***PMC*** podem usufruir de maior destaque.
  - Consiste de mapear o comportamento de um processo se baseando somente em diversas medições efetivadas em suas entradas e saídas (sem conhecer a modelagem matemática).
  - Observa-se aqui uma das principais características intrínsecas das redes neurais artificiais, ou seja, o aprendizado a partir de exemplos.
  - No caso de aproximação de funções, traduz-se na disponibilização de um conjunto de entradas/saídas que reproduzem o comportamento do sistema a ser tratado.
  - De fato, há muitas aplicações em que as únicas informações disponíveis se resumem a uma coleção de dados de entradas/saídas.
  - Nesta direção, constata-se que as ***RNA*** têm sido extensivamente aplicados nas seguintes situações:
    - O processo a ser modelado é de certa forma complexo.
    - Naqueles casos em que as utilizações de métodos convencionais produzem resultados insatisfatórios.
    - Naqueles casos em que os sistemas convencionais exigem requisitos computacionais bem sofisticados.

29

6. Aplicabilidade do PMC

Teorema da aproximação universal

- **Aspectos do teorema da aproximação universal:**
  - Baseado nas demonstrações de Kolmogorov, estas fornecem as bases para se definir as configurações de redes ***PMC*** p/ finalidade de mapear funções algébricas.
  - Assumindo que *g*(.) a ser adotada nas redes ***PMC*** sejam contínuas e limitadas em suas imagens, tais como a logística e tangente hiperbólica, demonstra-se então que:
    - Um ***PMC***, **composto de apenas uma camada escondida**, é capaz de mapear qualquer função contínua no espaço real. Em termos matemáticos, tem-se:



$$y(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n_1} \underbrace{\lambda_i}_{\text{parcela (i)}} \cdot \underbrace{g_i^{(1)}(u_i^{(1)})}_{\text{parcela (ii)}}$$

$$u_i^{(1)} = \sum_{j=1}^{n} W_{ji}^{(1)} \cdot x_j - \theta_i$$

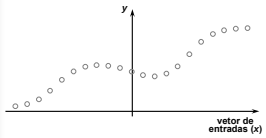
  - O neurônio de saída (ativação linear) realiza tão somente a combinação linear das funções de ativação logística implementadas pelos neurônios da camada intermediária.
  - A função *y* a ser mapeada será constituída por superposição de logísticas {parcela (ii)}, representadas pelos termos $g_i^{(1)}(u_i^{(1)})$, que são ponderadas por fatores λ_i {parcela (i)}.

30

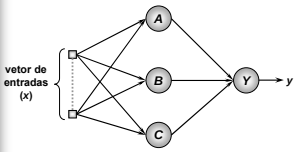
6. Aplicabilidade do PMC

Teorema da aproximação universal (Ilustração)

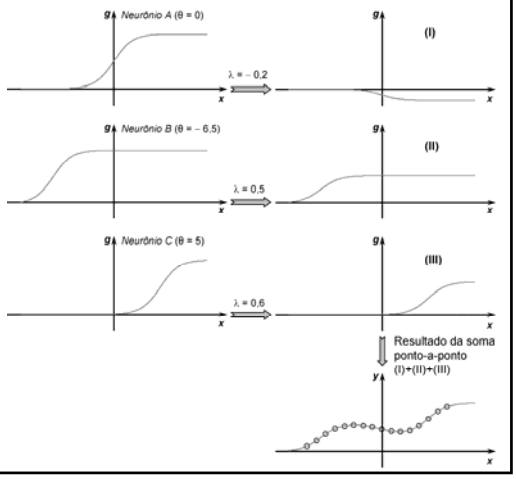
- **Conjunto de amostras relacionando entradas/saídas referente ao processo (função) a ser mapeado.**



- ***PMC* aplicado para mapear a função representada pelas amostras de treinamento acima:**



- **Configuração de *PMC* após o ajuste de seus pesos:**
  - **Parâmetro θ** ➔ responsável pela translação das funções de ativação.
  - **Parâmetro λ** ➔ responsável pelo escalamento das funções de ativação.



31

7. Questões Sobre o PMC

Reflexões, observações e aspectos práticos

Aspectos Práticos

- Embora um ***PMC*** com apenas uma camada escondida seja suficiente para mapear qualquer função não-linear contínua definida num domínio compacto (fechado), há situações em que se utilizam mais de duas camadas delas.
- A adoção de mais camadas escondidas podem ser apropriadas tanto para o propósito de incrementar o desempenho do treinamento como de reduzir a topologia estrutural da rede.

Exercícios de Reflexão

1) Explique se é possível realizar o treinamento da rede PMC, por meio do algoritmo *backpropagation*, quando se inicializa todas as matrizes de pesos com elementos nulos. Discorra também se há então alguma implicação quando se inicializa todos os elementos das matrizes de pesos com valores iguais (diferentes de zeros). ***{Exercício 1}***

2) Considerando os problemas envolvendo aproximação de funções, discorra então se há alguma vantagem e/ou desvantagem em se utilizar a função de ativação linear para os neurônios da camada de saída da rede ao invés do uso da tangente hiperbólica. ***{Exercício 2}***

32