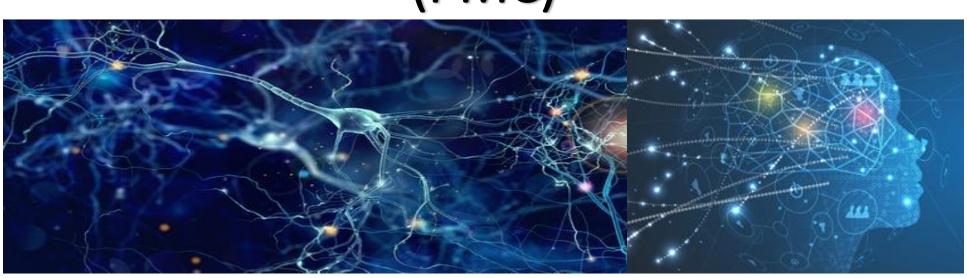
PGCC015 Inteligência Computacional

Prof. Matheus Giovanni Pires
mgpires@ecomp.uefs.br

1

Redes Perceptron Multicamadas (PMC)



2

2

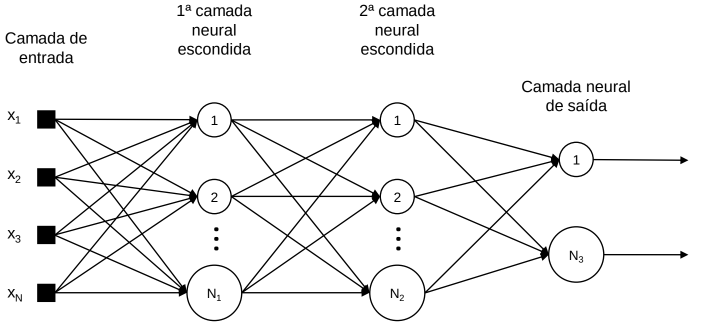
PMC: Aplicações

- As PMC podem ser aplicadas em diversos tipos de problemas, as principais áreas são:
  - Aproximação universal de funções
  - Reconhecimento de padrões
    - Fronteiras não-lineares
    - Conjuntos convexos e não-convexos
    - Conjuntos desconexos
  - Identificação e controle de processos
  - Previsão de séries temporais
  - Otimização de sistemas

PMC: Arquitetura e topologia

- As Redes Perceptron Multicamadas (PMC) são caracterizadas pela presença de pelo menos uma camada escondida (intermediária) de neurônios
- Possui arquitetura *feedforward* de camadas múltiplas
- Possui aprendizado supervisionado!
- Diferentemente do Perceptron e Adaline, além das camadas escondidas, a camada de saída pode ser composta por diversos neurônios
- A definição da topologia depende de vários aspectos
  - Tipo do problema, disposição espacial das amostras de treinamento, nível de ruído presente nos dados de treinamento

PMC: Arquitetura e topologia

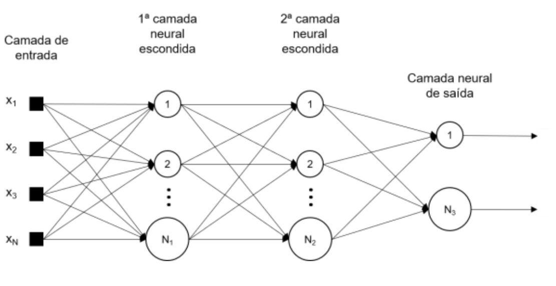


5

5

PMC: Funcionamento

- Os sinais (entradas) são propagados um a um em direção à camada neural de saída
- As saídas dos neurônios da primeira camada neural escondida serão as entradas dos neurônios da segunda camada neural escondida, e assim por diante.
- A propagação dos sinais de entrada é sempre realizada em um único sentido, ou seja, em direção à camada neural de saída



6

6

PMC: Treinamento

- O ajuste dos pesos e do limiar é realizado por meio de um treinamento supervisionado
  - As saídas do PMC são comparadas com as respectivas respostas desejadas
- O processo de treinamento é realizado através do algoritmo Backpropagation, conhecido também como Regra Delta Generalizada
- O algoritmo Backpropagtion consiste de 2 fases:
  - ***Forward***: um padrão é aplicado nas entradas da rede e as informações são propagadas camada a camada até as suas saídas
    - As respostas obtidas da rede considera apenas os valores atuais dos pesos sinápticos e limiares de seus neurônios, os quais permanecerão inalterados durante esta fase
  - ***Backward***: a partir das saídas da rede, calcula-se o erro que será propagado (de volta), objetivando o ajuste dos pesos e dos limiares de todos os neurônios

7

7

PMC: Treinamento

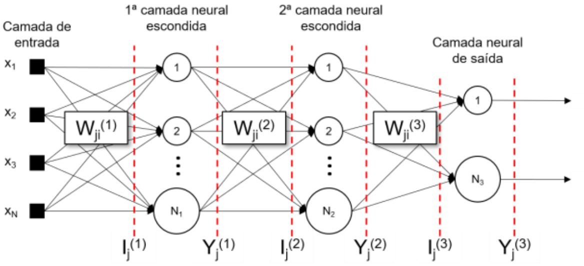
- As aplicações sucessivas das fases *forward* e *backward* fazem com que os pesos sinápticos e limiares se ajustem automaticamente em cada iteração, implicando na gradativa diminuição da soma dos erros
- O treinamento é finalizado quando a soma dos erros estiver dentro de valores aceitáveis

8

8

Backpropagation

- Definindo variáveis e parâmetros (matrizes de pesos)
  - $W_{ji}^{(L)}$ são matrizes pesos que conectam o j-ésimo neurônio da camada (***L***) ao i-ésimo neurônio da camada (***L-1***)
    - $W_{ji}^{(3)}$ → peso sináptico conectando o j-ésimo neurônio da camada de saída ao i-ésimo neurônio da camada 2
    - $W_{ji}^{(2)}$ → peso sináptico conectando o j-ésimo neurônio da camada 2 ao i-ésimo neurônio da camada 1
    - $W_{ji}^{(1)}$ → peso sináptico conectando o j-ésimo neurônio da camada 1 ao i-ésimo neurônio da camada de entrada



9

9

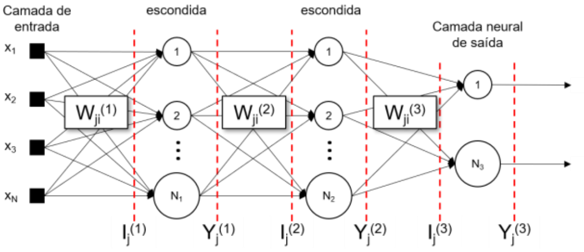
Backpropagation

- Definindo variáveis e parâmetros (vetores de entrada)
  - $I_j^{(L)}$ são vetores de entrada ponderada do j-ésimo neurônio da camada L

$$I_j^{(1)} = \sum_{i=0}^{N} W_{ji}^{(1)}.x_i \Leftrightarrow I_j^{(1)} = W_{j0}^{(1)}.x_0 + W_{j1}^{(1)}.x_1 + \ldots + W_{jN}^{(1)}.x_N \qquad (1)$$

$$I_j^{(2)} = \sum_{i=0}^{N_1} W_{ji}^{(2)}.Y_i^{(1)} \Leftrightarrow I_j^{(2)} = W_{j0}^{(2)}.Y_0^{(1)} + W_{j1}^{(2)}.Y_1^{(1)} + \ldots + W_{jN_1}^{(2)}.Y_{N_1}^{(1)} \qquad (2)$$

$$I_j^{(3)} = \sum_{i=0}^{N_2} W_{ji}^{(3)}.Y_i^{(2)} \Leftrightarrow I_j^{(3)} = W_{j0}^{(3)}.Y_0^{(2)} + W_{j1}^{(3)}.Y_1^{(2)} + \ldots + W_{jN_2}^{(3)}.Y_{N_2}^{(2)} \qquad (3)$$



10

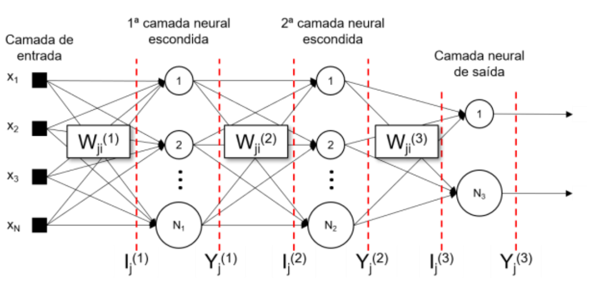
10

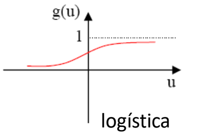
Backpropagation

- Definindo variáveis e parâmetros (vetores de saída)
  - $Y_j^{(L)}$ são vetores de saída do j-ésimo neurônio da camada L

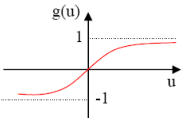
$$Y_j^{(1)} = g(I_j^{(1)}) \quad (4) \qquad Y_j^{(2)} = g(I_j^{(2)}) \quad (5) \qquad Y_j^{(3)} = g(I_j^{(3)}) \quad (6)$$

  - g(.) deve ser uma função contínua e diferenciável em todo seu domínio, como a função logística (sigmóide) ou tangente hiperbólica





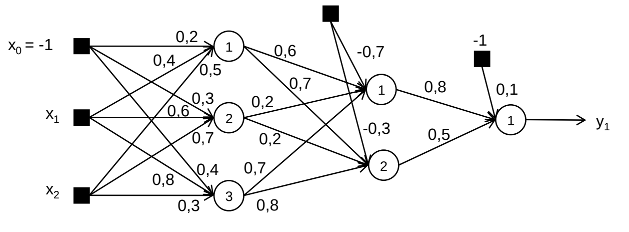
logística



tangente hiperbólica

Backpropagation

- Exemplo



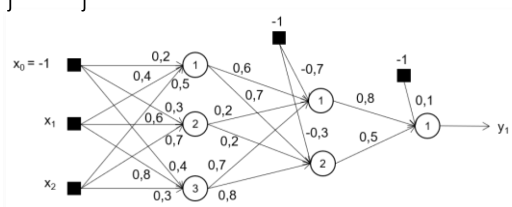
$$W_{ji}^{(1)} = \begin{bmatrix} 0,2 & 0,4 & 0,5 \\ 0,3 & 0,6 & 0,7 \\ 0,4 & 0,8 & 0,3 \end{bmatrix}$$

$$W_{ji}^{(3)} = [0,1 \quad 0,8 \quad 0,5]$$

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

Backpropagation

- Assumindo-se um sinal de entrada definido por x_1=0,3 e x_2=0,7, os vetores $I_j^{(1)}$ e $Y_j^{(1)}$ são:



$$I_j^{(1)} = \begin{bmatrix} I_1^{(1)} \\ I_2^{(1)} \\ I_3^{(1)} \end{bmatrix} = \begin{bmatrix} w_{10}^{(1)}.x_0 + w_{11}^{(1)}.x_1 + w_{12}^{(1)}.x_2 \\ w_{20}^{(1)}.x_0 + w_{21}^{(1)}.x_1 + w_{22}^{(1)}.x_2 \\ w_{30}^{(1)}.x_0 + w_{31}^{(1)}.x_1 + w_{32}^{(1)}.x_2 \end{bmatrix} = \begin{bmatrix} 0{,}2.(-1) + 0{,}4 \,.\, 0{,}3 + 0{,}5 \,.\, 0{,}7 \\ 0{,}3.(-1) + 0{,}6 \,.\, 0{,}3 + 0{,}7 \,.\, 0{,}7 \\ 0{,}4.(-1) + 0{,}8 \,.\, 0{,}3 + 0{,}3 \,.\, 0{,}7 \end{bmatrix} = \begin{bmatrix} 0{,}27 \\ 0{,}37 \\ 0{,}05 \end{bmatrix}$$

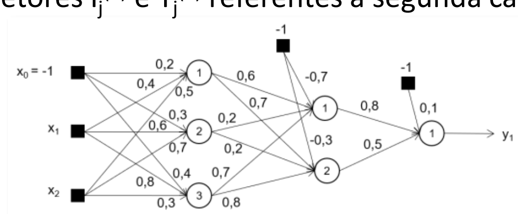
$$Y_j^{(1)} = \begin{bmatrix} Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(1)}) \\ g(I_2^{(1)}) \\ g(I_3^{(1)}) \end{bmatrix} = \begin{bmatrix} \tanh(0{,}27) \\ \tanh(0{,}37) \\ \tanh(0{,}05) \end{bmatrix} = \begin{bmatrix} 0{,}26 \\ 0{,}35 \\ 0{,}05 \end{bmatrix} \xrightarrow{Y_0^{(1)}=-1} Y_j^{(1)} = \begin{bmatrix} Y_0^{(1)} \\ Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0{,}26 \\ 0{,}35 \\ 0{,}05 \end{bmatrix}$$

13

13

Backpropagation

- Os vetores $I_j^{(2)}$ e $Y_j^{(2)}$ referentes à segunda camada são:



$$\begin{bmatrix} Y_0^{(1)} \\ Y_1^{(1)} \\ Y_2^{(1)} \\ Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0{,}26 \\ 0{,}35 \\ 0{,}05 \end{bmatrix}$$

$$I_j^{(2)} = \begin{bmatrix} I_1^{(2)} \\ I_2^{(2)} \end{bmatrix} = \begin{bmatrix} w_{10}^{(2)}.Y_0^{(1)} + w_{11}^{(2)}.Y_1^{(1)} + w_{12}^{(2)}.Y_2^{(1)} + w_{13}^{(2)}.Y_3^{(1)} \\ w_{20}^{(2)}.Y_0^{(1)} + w_{21}^{(2)}.Y_1^{(1)} + w_{22}^{(2)}.Y_2^{(1)} + w_{23}^{(2)}.Y_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0{,}96 \\ 0{,}59 \end{bmatrix}$$

$$Y_j^{(2)} = \begin{bmatrix} Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} g(I_1^{(2)}) \\ g(I_2^{(2)}) \end{bmatrix} = \begin{bmatrix} \tanh(0{,}96) \\ \tanh(0{,}53) \end{bmatrix} = \begin{bmatrix} 0{,}74 \\ 0{,}53 \end{bmatrix} \xrightarrow{Y_0^{(2)}=-1} Y_j^{(2)}$$

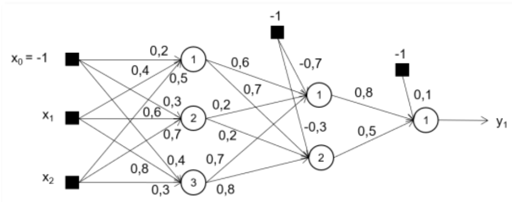
$$Y_j^{(2)} = \begin{bmatrix} Y_0^{(2)} \\ Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0{,}74 \\ 0{,}53 \end{bmatrix}$$

14

14

Backpropagation

- Os vetores $I_j^{(3)}$ e $Y_j^{(3)}$ referentes à terceira camada são:



$$Y_j^{(2)} = \begin{bmatrix} Y_0^{(2)} \\ Y_1^{(2)} \\ Y_2^{(2)} \end{bmatrix} = \begin{bmatrix} -1 \\ 0{,}74 \\ 0{,}53 \end{bmatrix}$$

$$I_j^{(3)} = \left[I_1^{(3)} \right] = \left[w_{10}^{(3)}.Y_0^{(2)} + w_{11}^{(3)}.Y_1^{(2)} + w_{12}^{(3)}.Y_2^{(2)} \right] = [0{,}76]$$

$$Y_j^{(3)} = \left[Y_1^{(3)} \right] = \left[g(I_1^{(3)}) \right] = [\tanh(0{,}76)] = [0{,}64]$$

- Nesta última expressão, dispensa-se a inserção do termo $Y_0^{(3)}$=-1, pois já se trata da última camada neural, sendo que o valor de $Y_1^{(3)}$ é a própria saída y_1 produzida por esta rede

Backpropagation

- O objetivo do processo de aprendizagem é ajustar as matrizes de pesos da rede, a fim de minimizar o Erro Quadrático Médio.
- Erro Quadrático referente a cada padrão *k* de entrada:

$$E(k) = \frac{1}{2}\sum_{j=1}^{N_3}(d_j\,(k) - Y_j^{(3)}(k))^2 \qquad (7)$$

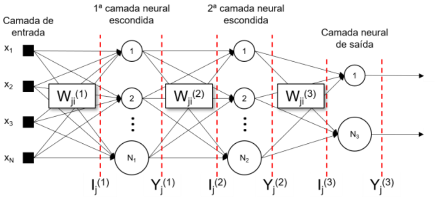
- Logo, o Erro Quadrático Médio, referente ao conjunto de treinamento composto por *p* amostras é definido por:

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

Backpropagation

- Ajustando os pesos da camada de saída
  - O ajuste dos pesos da matriz $W_{ji}^{(3)}$ tem por objetivo minimizar o erro entre a saída da rede e a saída desejada
  - Usando o gradiente descendente e a regra da cadeia, tem-se:

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



Calculando as derivadas parciais:

$$\frac{\partial I_j^{(3)}}{\partial W_{ji}^{(3)}} = \sum_{i=0}^{N_2} \frac{\partial}{\partial W_{ji}^{(3)}} W_{ji}^{(3)} . Y_i^{(2)} = Y_i^{(2)} \qquad (10)$$

$$\frac{\partial Y_j^{(3)}}{\partial I_j^{(3)}} = g'(I_j^{(3)}) \qquad (11)$$

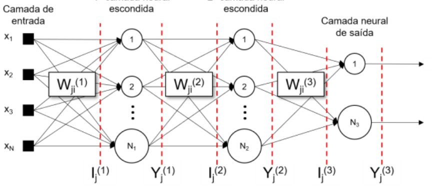
17

17

Backpropagation

- Ajustando os pesos da camada de saída

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



Calculando as derivadas parciais:

$$\frac{\partial E}{\partial Y_j^{(3)}} = \sum_{j=1}^{N_3} \frac{\partial}{\partial Y_j^{(3)}} \frac{1}{2}\left(d_j(k) - Y_j^{(3)}(k)\right)^2 = \sum_{j=1}^{N_3} 2\left(\frac{1}{2}\left(d_j(k) - Y_j^{(3)}(k)\right)\right) \cdot \left(d_j(k) - Y_j^{(3)}(k)\right)'$$

$$\frac{\partial E}{\partial Y_j^{(3)}} = -\left(d_j(k) - Y_j^{(3)}(k)\right) \qquad (12)$$

18

Backpropagation

- Substituindo (10), (11) e (12) em (9)

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

- Para minimizar o erro, o ajuste deve ser realizado na direção oposta ao gradiente

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

- $\delta_j^{(3)}$ é o gradiente local do *j*-ésimo neurônio da camada de saída

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

- Por fim, a expressão que atualiza a matriz de pesos:

$$W_{ji}^{(3)}(t+1) = W_{ji}^{(3)}(t) + \eta \cdot \delta_j^{(3)} \cdot Y_j^{(2)} \Leftrightarrow W_{ji}^{(3)} = W_{ji}^{(3)} + \eta \cdot \delta_j^{(3)} \cdot Y_j^{(2)}$$

(16) (17)

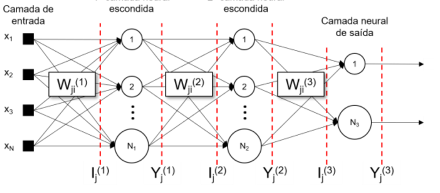
19

19

Backpropagation

- Ajustando os pesos da 2ª camada escondida

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



Calculando as derivadas parciais:

$$\frac{\partial I_j^{(2)}}{\partial W_{ji}^{(2)}} = \sum_{i=0}^{N_1} \frac{\partial}{\partial W_{ji}^{(2)}} W_{ji}^{(2)}.Y_i^{(1)} = Y_i^{(1)} \qquad (19)$$

$$\frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} = g'(I_j^{(2)}) \qquad (20)$$

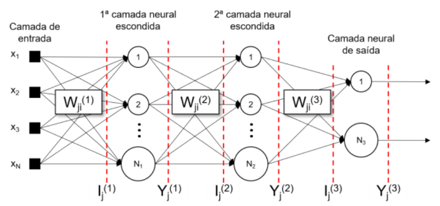
20

20

Backpropagation

- Ajustando os pesos da 2ª camada escondida

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



Calculando as derivadas parciais:

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{N_3} \frac{\partial E}{\partial I_k^{(3)}} \cdot \frac{\partial I_k^{(3)}}{\partial Y_j^{(2)}} = \sum_{k=1}^{N_3} \underbrace{\frac{\partial E}{\partial I_k^{(3)}}}_{\text{(i)}} \cdot \underbrace{\frac{\partial}{\partial Y_j^{(2)}} \sum_{k=1}^{N_3} W_{kj}^{(3)} \cdot Y_j^{(2)}}_{\text{(ii)}} \quad (21)$$

Importante: os pesos de $W_{ji}^{(3)}$ já foram ajustados com valores reais do erro!

$$\frac{\partial E}{\partial Y_j^{(2)}} = \sum_{k=1}^{N_3} \underbrace{\frac{\partial E}{\partial I_k^{(3)}}}_{\text{(i)}} \cdot \underbrace{W_{kj}^{(3)}}_{\text{(ii)}} \quad (22)$$

(i) foi obtido multiplicando (11) por (12)

$$\frac{\partial E}{\partial Y_j^{(2)}} = -\sum_{k=1}^{N_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \quad (23)$$

21

Backpropagation

- Ajustando os pesos da 2ª camada escondida
  - Substituindo (19), (20) e (23) em (18)

$$\frac{\partial I_j^{(2)}}{\partial W_{ji}^{(2)}} = \sum_{i=0}^{N_1} \frac{\partial}{\partial W_{ji}^{(2)}} W_{ji}^{(2)} . Y_i^{(1)} = Y_i^{(1)} \quad (19) \qquad \frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} = g'(I_j^{(2)}) \quad (20)$$

$$\frac{\partial E}{\partial Y_j^{(2)}} = -\sum_{k=1}^{N_3} \delta_k^{(3)} \cdot W_{kj}^{(3)} \quad (23) \qquad \nabla E^{(2)} = \frac{\partial E}{\partial W_{ji}^{(2)}} = \frac{\partial E}{\partial Y_j^{(2)}} \cdot \frac{\partial Y_j^{(2)}}{\partial I_j^{(2)}} \cdot \frac{\partial I_j^{(2)}}{\partial W_{ji}^{(2)}} \quad (18)$$

$$\frac{\partial E}{\partial W_{ji}^{(2)}} = -\left(\sum_{k=1}^{N_3} \delta_k^{(3)} \cdot W_{kj}^{(3)}\right) . g'(I_j^{(2)}) . Y_i^{(1)} \quad (24)$$

- O ajuste deve ser realizado na direção oposta ao gradiente

$$\Delta W_{ji}^{(2)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{(2)}} \Leftrightarrow \Delta W_{ji}^{(2)} = \eta \cdot \delta_j^{(2)} \cdot Y_i^{(1)} \quad (25) \qquad \delta_j^{(2)} = -\left(\sum_{k=1}^{N_3} \delta_k^{(3)} \cdot W_{kj}^{(3)}\right) . g'(I_j^{(2)}) \quad (26)$$

22

Backpropagation

- Por fim, a expressão que atualiza a matriz de pesos:

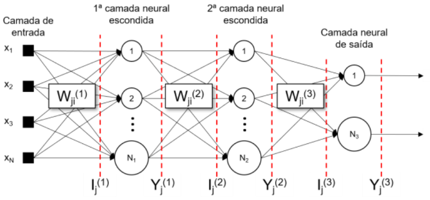
$$W_{ji}^{(2)}(t+1) = W_{ji}^{(2)}(t) + \eta \cdot \delta_j^{(2)} \cdot Y_j^{(1)} \Leftrightarrow W_{ji}^{(2)} = W_{ji}^{(2)} + \eta \cdot \delta_j^{(2)} \cdot Y_j^{(1)}$$

(27) (28)

Backpropagation

- Ajustando os pesos da 1ª camada escondida

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



Calculando as derivadas parciais:

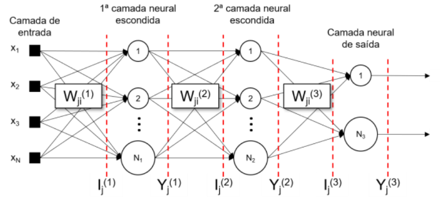
$$\frac{\partial I_j^{(1)}}{\partial W_{ji}^{(1)}} = \sum_{i=0}^{N} \frac{\partial}{\partial W_{ji}^{(1)}} W_{ji}^{(1)} . x_i = x_i \quad (30)$$

$$\frac{\partial Y_j^{(1)}}{\partial I_j^{(1)}} = g'(I_j^{(1)}) \quad (31)$$

Backpropagation

- Ajustando os pesos da 1ª camada escondida

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



Calculando as derivadas parciais:

$$\frac{\partial E}{\partial Y_j^{(1)}} = \sum_{k=1}^{N_2} \frac{\partial E}{\partial I_k^{(2)}} \cdot \frac{\partial I_k^{(2)}}{\partial Y_j^{(1)}} = \underbrace{\sum_{k=1}^{N_2} \frac{\partial E}{\partial I_k^{(2)}}}_{\text{(i)}} \cdot \underbrace{\frac{\partial}{\partial Y_j^{(1)}} \sum_{k=1}^{N_2} W_{kj}^{(2)} \cdot Y_j^{(1)}}_{\text{(ii)}} \quad (32)$$

$$\frac{\partial E}{\partial Y_j^{(1)}} = \underbrace{\sum_{k=1}^{N_2} \frac{\partial E}{\partial I_k^{(2)}}}_{\text{(i)}} \cdot \underbrace{W_{kj}^{(2)}}_{\text{(ii)}} \quad (33)$$

(i) foi obtido multiplicando (20) por (21)

$$\frac{\partial E}{\partial Y_j^{(1)}} = -\sum_{k=1}^{N_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \quad (34)$$

Backpropagation

- Ajustando os pesos da 1ª camada escondida
  - Substituindo (30), (31) e (34) em (29)

$$\frac{\partial I_j^{(1)}}{\partial W_{ji}^{(1)}} = \sum_{i=0}^{N} \frac{\partial}{\partial W_{ji}^{(1)}} W_{ji}^{(1)} . x_i = x_i \quad (30)$$

$$\frac{\partial Y_j^{(1)}}{\partial I_j^{(1)}} = g'(I_j^{(1)}) \quad (31)$$

$$\frac{\partial E}{\partial Y_j^{(1)}} = -\sum_{k=1}^{N_2} \delta_k^{(2)} \cdot W_{kj}^{(2)} \quad (34)$$

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

$$\frac{\partial E}{\partial W_{ji}^{(1)}} = -\left(\sum_{k=1}^{N_2} \delta_k^{(2)} \cdot W_{kj}^{(2)}\right) . g'(I_j^{(1)}) \cdot x_i \quad (35)$$

- O ajuste deve ser realizado na direção oposta ao gradiente

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

$$\delta_j^{(1)} = -\left(\sum_{k=1}^{N_2} \delta_k^{(2)} \cdot W_{kj}^{(2)}\right) . g'(I_j^{(1)}) \quad (37)$$

Backpropagation

- Por fim, a expressão que atualiza a matriz de pesos:

$$W_{ji}^{(1)}(t+1) = W_{ji}^{(1)}(t) + \eta \cdot \delta_j^{(1)} \cdot x_i \Leftrightarrow W_{ji}^{(1)} = W_{ji}^{(1)} + \eta \cdot \delta_j^{(1)} \cdot x_i$$

(38) (39)

27

27

Termo de *momentum*

- A velocidade do algoritmo *backpropagation* pode ser aumentada incluindo um termo de "momentum", ou seja:

$$W_{ji}(t+1) = W_{ji}(t) + \alpha \cdot \left(W_{ji}(t) - W_{ji}(t-1)\right) + \eta \cdot \delta_j^{(1)} \cdot Y_j$$

$\boxed{0 < \alpha \leq 0.9}$

x_i quando for a primeria camada!

28

PMC: Algoritmo de treinamento

```
Inicializar w com valores aleatórios pequenos;
Definir a taxa de aprendizagem {η} e a precisão requerida {ε}
EQM_ant ← INF; EQM_atual ← 0;
Epoca ← 0;
Enquanto | EQM_atual - EQM_ant | > ε
    EQM_ant ← EQM_atual ; (8)
    Para cada par de treinamento {x(k),d(k)} faça
            Calcular Ij(1) e Yj(1); (1) e (4)
            Calcular Ij(2) e Yj(2); (2) e (5)
            CaLcular Ij(3) e Yj(3); (3) e (6)
            Determinar δj(3);   (15)
            Ajustar wji(3);     (17)
            Determinar δj(2);   (26)
            Ajustar wji(2);     (28)
            Determinar δj(1);   (37)
            Ajustar wji(1);     (39)
    fim_para;
    EQM_atual ← EQM; (8)
    Epoca ← Epoca + 1;
fim_enquanto.
```

29

29

PMC: Algoritmo de teste

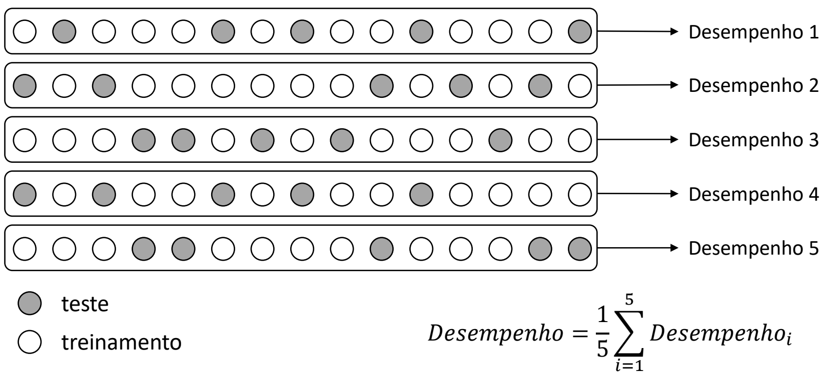
```
Obter conjunto de teste T;
Utilizar a matriz w ajustada no treinamento;
Para cada padrão x ∈ T a ser reconhecido Faça
    Calcular Ij(1) e Yj(1); (1) e (4)
    Calcular Ij(2) e Yj(2); (2) e (5)
    CaLcular Ij(3) e Yj(3); (3) e (6)
fim_para;
Disponibilizar as saídas da rede, as quais são
dadas pelos elementos contidos em Yj(3)
```

30

30

Validação cruzada (cross-validation)

- Validação cruzada Holdout
  - É o tipo mais simples de cross-validation
  - Os dados são separados em dois conjuntos: treinamento e teste
  - Geralmente, 2/3 dos dados são usados para treinamento, e 1/3 para teste



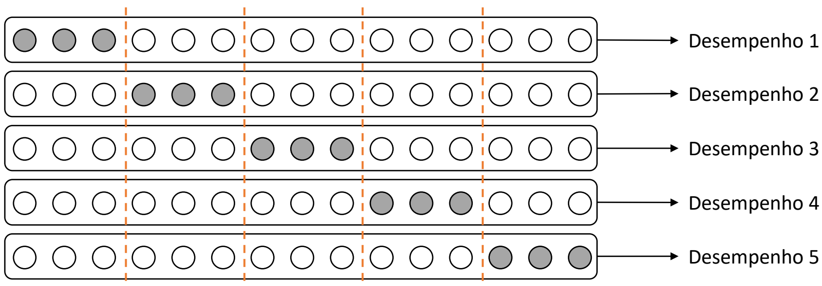
$$Desempenho = \frac{1}{5}\sum_{i=1}^{5} Desempenho_i$$

Validação cruzada (cross-validation)

- Validação cruzada Holdout
  - Cada classe deve ser representada na mesma proporção nos conjuntos de treinamento e teste
  - No momento da escolha dos dados para formar os conjuntos de treinamento e teste, pode acontecer que dados de uma certa classe não estejam presentes no conjunto de treinamento
  - A garantia de que as classes sejam representadas apropriadamente nos conjuntos de treinamento e teste é realizada por um método chamado Estratificação (*Stratification*)

Validação cruzada (cross-validation)

- Validação cruzada por K-partições (K-*fold cross-validation*)
  - Uma forma de melhorar a validação cruzada Holdout
  - Os dados são divididos em *k* sub-conjuntos: *k*-1 conjuntos são usados para treinamento
  - *k* medidas de desempenho são calculadas



teste

treinamento

$$Desempenho = \frac{1}{5}\sum_{k=1}^{5} Desempenho_k$$

33

33

Validação cruzada (cross-validation)

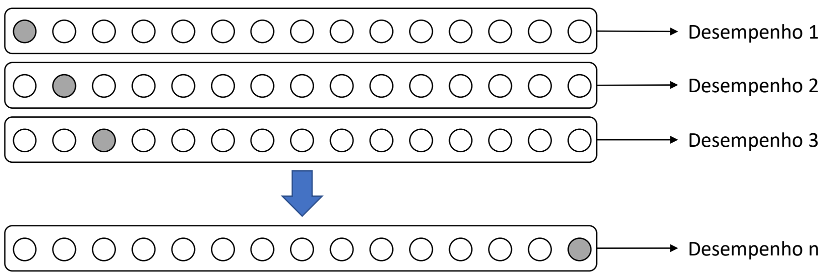
- Validação cruzada por K-partições (K-*fold cross-validation*)
- Geralmente, o valor de *k* é 10. Por que 10?
  - Estudos em várias bases de dados, com diferentes técnicas de aprendizado, tem mostrado que 10 é o número que melhor estima o erro
  - Há também evidências teóricas que também dão suporte a estes resultados
  - No entanto, o debate sobre a melhor forma de avaliação dos modelos de Machine Learning ainda é grande
- Para encontrar uma boa estimativa do erro, o procedimento padrão é executar o *cross-validation* 10 vezes!
  - Ou seja, usando 10-fold cross-validation, o seu modelo será testado 100 vezes!

34

34

Validação cruzada (cross-validation)

- Validação cruzada *Leave-One-Out*
  - É simplesmente *n*-fold cross-validation, onde *n* é o número de instâncias da base de dados
  - Uma instância é testada por vez, o restante são usadas como dados de treinamento



teste

treinamento

$$Desempenho = \frac{1}{n}\sum_{k=1}^{n} Desempenho_k$$

35

35

Somente treinamento e teste?

- O nosso modelo é treinado com dados atuais (dados de treinamento)
- Para estimar o erro futuro, ou seja, como será o desempenho do nosso modelo quando testado com dados NUNCA vistos, utiliza-se os dados de teste.
  - Lembrando que esse erro é um erro estimado, pois nunca saberemos como serão os dados futuros!
- Mas, e se quisermos aprender os hiper parâmetros do nosso modelo?
  - Hiper parâmetros de um modelo são aqueles que devem ser especificados antes do ajuste do próprio modelo
  - Por exemplo, antes de treinar uma rede neural, é necessário definir sua topologia (número de camadas, número de neurônios por camada, tipo das funções de ativação, etc.)

36

36

Somente treinamento e teste?

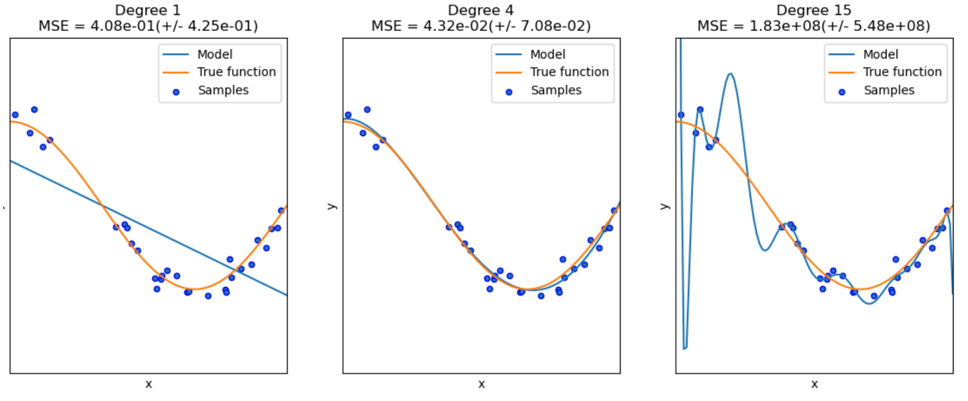
- Importante ressaltar que os dados de teste NÃO PODEM SER USADOS para criar o nosso modelo
- Como então aprender os hiper parâmetros?
  - Usar um conjunto de dados conhecido como Validação.
  - Geralmente, estes dados são selecionados do conjunto de treinamento.
- Os dados de treinamento, validação e teste devem ser diferentes!
  - Os dados de validação devem ser diferentes dos dados treinamento para obter um bom desempenho na otimização do modelo, e o conjunto de teste deve ser diferente do conjunto de validação para obter uma estimativa do erro confiável.
- Após o uso dos dados de validação, estes podem retornar ao conjunto de treinamento, para que o modelo seja retreinado, maximizando o uso dos dados!

Underfitting vs. Overfitting

- Por que precisamos nos importar com underfitting e overfitting?
  - *Overfitting* = memorização
  - *Underfitting* = incapacidade de aprender algo
- O que sempre esperamos dos nossos modelos (classificadores, preditores)?
  - Esperamos que eles generalizem bem!
- O que é generalização?
  - Generalization is the model's ability to give sensible outputs to sets of input that it has never seen before.[1]
- A model that generalizes well is a model that is neither underfit nor overfit.[1]

Underfitting vs. Overfitting

- Este exemplo demonstra o uso de regressão linear com polinômios para aproximar uma função não linear[2]



39

39

Underfitting vs. Overfitting

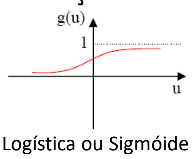
- O *overfitting* / *underfitting* é avaliado quantitativamente (como por exemplo, o Erro Quadrático Médio) através do método *cross-validation*
- No *overfitting*, o erro é baixo no treinamento e alto no teste (validação)
- No *underfitting*, o erro é alto no treinamento e teste
- O aumento exagerado de camadas intermediárias e/ou de neurônios por camada pode resultar em *overfitting*
- Por outro lado, uma rede com poucas camadas e/ou neurônios, pode resultar em *underfitting*

40

40

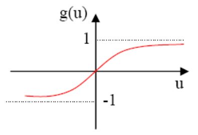
Normalização dos dados

- A normalização dos dados melhora o desempenho do treinamento das redes neurais artificiais, e também da acurácia[3]
- Consiste em transformar os valores dos dados para a mesma faixa de variação das funções de ativação dos neurônios
- Normalização MAX-MIN



Logística ou Sigmóide

$$x^{NORM} = \frac{x - x^{MIN}}{x^{MAX} - x^{MIN}}$$



Tangente Hiperbólica

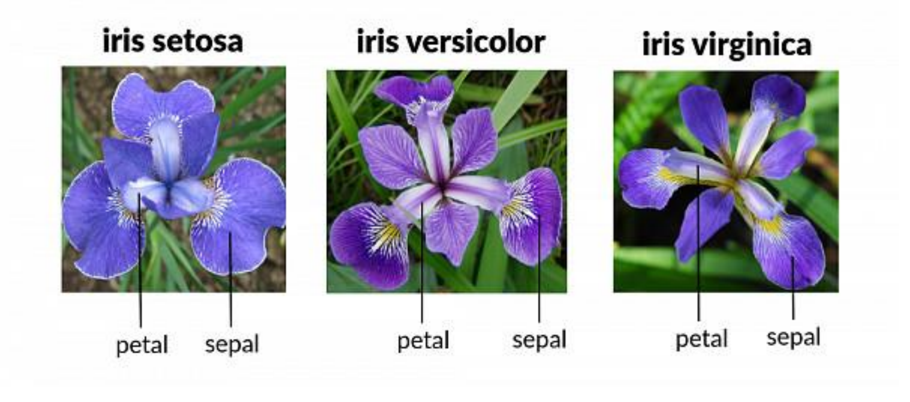
$$x^{NORM} = 2\left(\frac{x - x^{MIN}}{x^{MAX} - x^{MIN}}\right) - 1$$

41

41

PMC: classificação de padrões

- Um problema de classificação de padrões consiste em associar um padrão de entrada (amostra, exemplo, instância) para uma das classes previamente conhecidas.



42

PMC: classificação de padrões

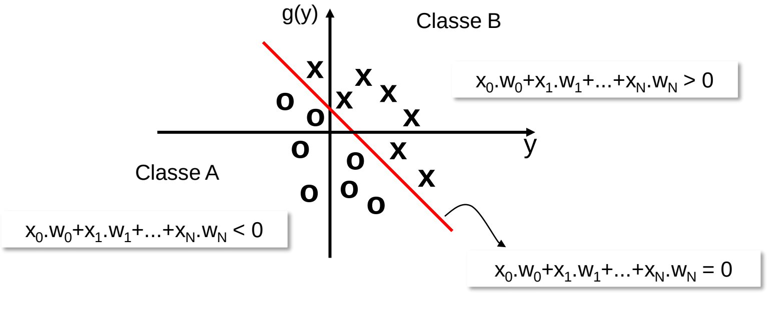
- Iris dataset

```
@attribute PetalLength real [1.0, 6.9]
@attribute PetalWidth real [0.1, 2.5]
@attribute Class {Iris-setosa, Iris-versicolor, Iris-virginica}
@inputs SepalLength, SepalWidth, PetalLength, PetalWidth
@outputs Class
@data
5.1, 3.5, 1.4, 0.2, Iris-setosa
4.9, 3.0, 1.4, 0.2, Iris-setosa
4.6, 3.1, 1.5, 0.2, Iris-setosa
7.0, 3.2, 4.7, 1.4, Iris-versicolor
6.4, 3.2, 4.5, 1.5, Iris-versicolor
6.9, 3.1, 4.9, 1.5, Iris-versicolor
6.3, 3.3, 6.0, 2.5, Iris-virginica
5.8, 2.7, 5.1, 1.9, Iris-virginica
7.1, 3.0, 5.9, 2.1, Iris-virginica
```

43

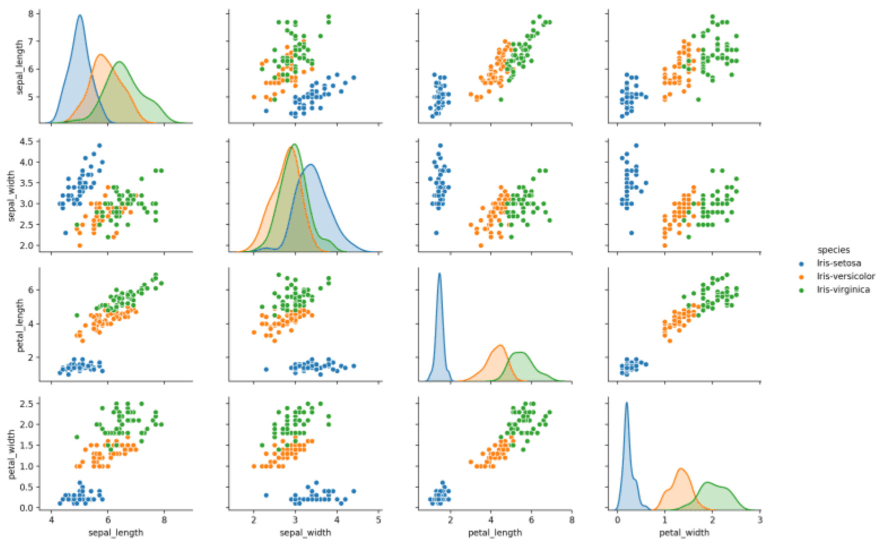
PMC: classificação de padrões

- Recapitulando... Um neurônio consegue classificar apenas um conjunto de dados, com duas classes linearmente separáveis!



44

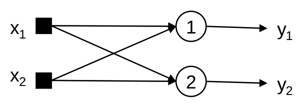
PMC: classificação de padrões

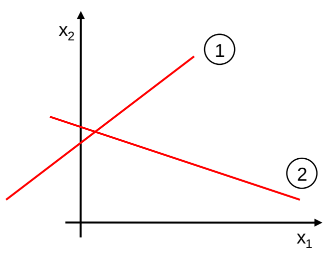


45

PMC: classificação de padrões

- PMC com uma camada e dois neurônios



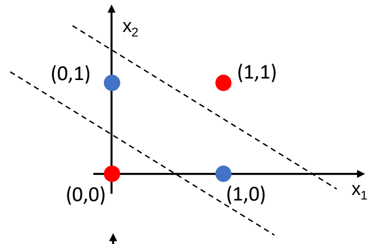


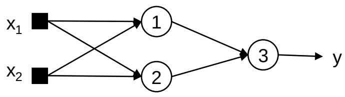
46

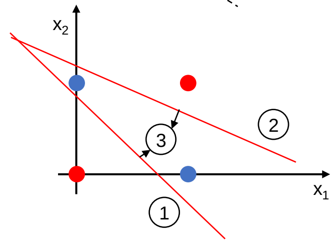
PMC: classificação de padrões

- PMC com uma camada intermediária e uma de saída
  - Problema do Ou-exclusivo

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	0





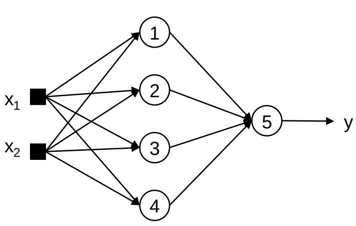


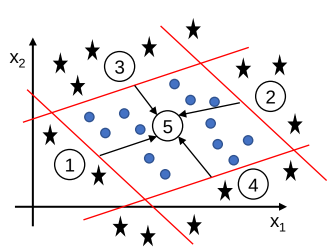
O neurônio 3 faz um combinação lógica das retas produzidas pelos neurônis 1 e 2

47

PMC: classificação de padrões

- PMC com uma camada intermediária e uma de saída



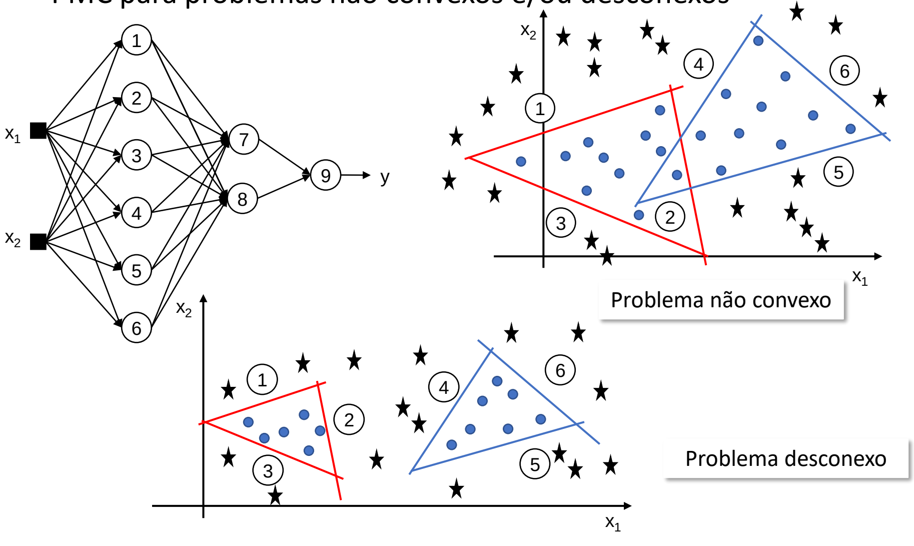


- O neurônio 5 faz a combinação de todas as superfícies de separação

48

PMC: classificação de padrões

- PMC para problemas não convexos e/ou desconexos



49

PMC: aspectos importantes

- O aumento do número de neurônios ou de camadas não significa que a rede irá generalizar melhor.
- Para duas topologias que estão generalizando com o mesmo grau de precisão, deve-se optar por aquela com menor número de neurônios.
- O conjunto de treinamento deve conter os valores mínimos e máximos de quaisquer variáveis de entrada e saída.
- As variáveis referentes às entradas da rede devem ser normalizadas para a faixa [0,1] se estiver usando a função sigmóide, ou então para [-1,1], se estiver utilizando a função tangente hiperbólica

50

50

PMC: aspectos importantes

- Aplicar técnicas de pré-processamento com o objetivo de minimizar redundâncias e reduzir complexidade dimensional dos dados de entrada da rede.
- A velocidade do algoritmo *backpropagation* pode ser aumentada incluindo um termo de *momentum*

Referências

- Silva, Ivan Nunes da; Spatti, Danilo Hernane; Flauzino, Rogério Andrade. **Redes Neurais Artificiais para engenharia e ciências aplicadas**. Artliber, 2010.
- Ian H. Witten; Eibe Frank; Mark A. Hall. **Data Mining: Pratical Machine Learning Tools and Techniques**, 3rd, Elsevier, 2011.
- [1] **What Are Overfitting and Underfitting in Machine Learning?** https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690
- [2] **Underfitting vs. Overfitting**. https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html
- [3] Shanker M, Hu MY, Hung MS, Effect of Data Standardization on Neural Network Training, **Omega**, vol.24, pp,385-397, https://doi.org/10.1016/0305-0483(96)00010-2