Introdução às Redes Neuronais Artificiais: Redes Feed-forward em Matlab

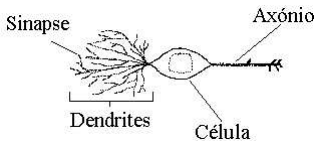
João Paulo Teixeira
joaopt@ipb.pt
www.ipb.pt/~joaopt
Departamento de Electrotecnia
ESTiG – Instituto Politécnico de Bragança

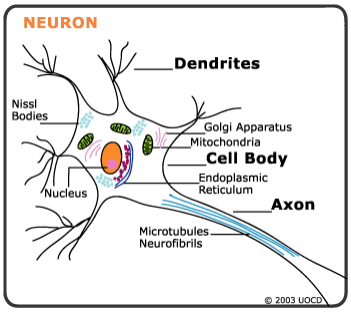
Resumo

- Evolução das RNA
- Tipos de Redes
  - Perceptrão
  - Rede Linear
  - Redes Recorrentes
  - Redes Radiais
- Redes Feed-Forward
  - Tipologia
  - Algoritmo BackPropagation
  - Funções de Treino (Matlab)
  - Paragem do treino (Conjuntos de treino, validação e teste)
  - Dados de entrada (escolha e pré-processamento)
- Aplicações
- RNA em Matlab
  - Nntool
    - Exemplo do Xor
  - Linha de comandos
    - Exemplo do Xor, seguimento de uma curva, série temporal, identificação de números
  - Proposta de trabalho
- Bibliografia

Neurónio biológico

- Estima-se que um cérebro humano tenha cerca de 10^{14} sinapses
- Efectua processamento paralelo
- É tolerante a falhas
- Aprende
- É inteligente e (às vezes) tem consciência





Evolução Histórica das RNA (1)

- 1943 – McCulloch (Psiquiatra e neuro-anatomista) e Pitts (Matemático) apresentaram o modelo matemático do neurónio artificial, e defendiam que uma rede destes neurónios poderia determinar qualquer função computável.
- 1949 – Hebb publicou o livro "The Organization of Behavior" em que explicita a regra de aprendizagem para a modificação sináptica.
- 1956 – Uttley demonstrou que a rede neuronal pode aprender a classificar grupos de padrões binários.
- 1958 – Rosenblatt apresentou o perceptrão – um método para a aprendizagem supervisionada para problemas de reconhecimento de padrões.
- 1960 – Widrow e Hoff introduziram o algoritmo LMS e usaram-no para criar ADALINE (Adaptative Linear Element).
- 1969 – Minsky e Pepert mostraram a incapacidade de um perceptrão resolver um problema simples (XOR). Juntamente com a falta de capacidade computacional levou a um afastamento das RNA.

Evolução Histórica das RNA (2)

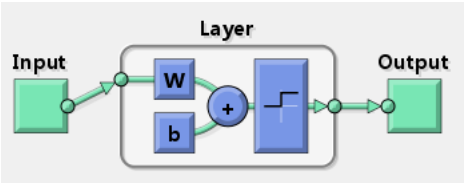
- 1982 – Hopfield usando a ideia de uma função energia, desenvolveu o conceito de rede realimentada, (ligações sinapticas simétricas) – criou as Redes Recorrentes.
- 1985 – Ackley, Hinton e Sejnowski desenvolveram a "Boltzmann machine" que na verdade foi a primeira rede neuronal multi-camada. Contudo os algoritmos de treino não eram eficientes.
- 1986 – Rumelhart, Hinton, Williams e McClelland desenvolveram e publicaram o algoritmo de "back-propagation" que ajusta os pesos das redes da frente para trás. O algoritmo permite treinar redes neuronais em multi-camada.
- 1986-> várias modificações/optimizações ao algoritmo de "back-propagation"

Paradigmas da Aprendizagem

- Supervisionada – existe um "professor" que ensina a rede qual deve ser a saída para cada vector de entrada. (Abordagem seguida nesta apresentação)
- Não supervisionada – não há uma saída para um vector de entrada. A rede aprende a agrupar os vectores de entrada em classes – 'clusters'.

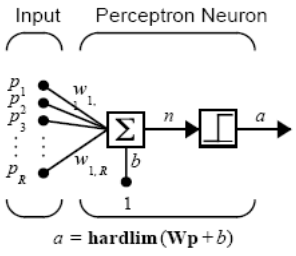
Tipologia de Redes Neuronais

- Perceptrão
  - apenas resolve problemas separáveis de forma linear
  - Adequado para resolver problemas simples de classificação de padrões;
  - A saída é 0 ou 1 (função hardlim);
  - Hoje é importante a sua referencia para ajudar a perceber as redes mais complexas (além da sua importância histórica).
  - (ver demos de nnd2n1 e nnd2n2, em Matlab)

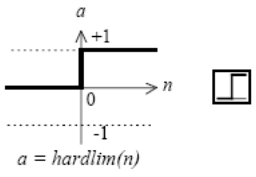


$$Output = f(\sum(input_i \times w_i) + b)$$

Tipologia de Redes Neuronais
Perceptrão – Regra de Aprendizagem



$a = \mathbf{hardlim}(\mathbf{Wp} + b)$



$a = hardlim(n)$

The Perceptron Learning Rule

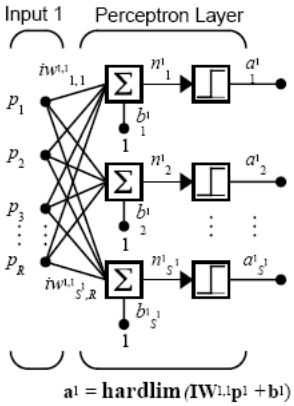
$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

where

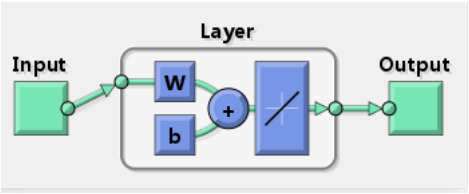
$$\mathbf{e} = \mathbf{t} - \mathbf{a}$$

Tipologia de Redes Neuronais
Rede de Perceptrons



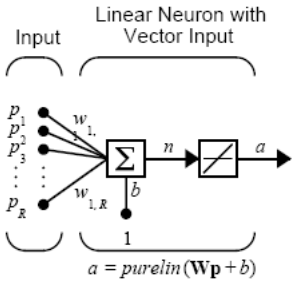
Tipologia de Redes Neuronais

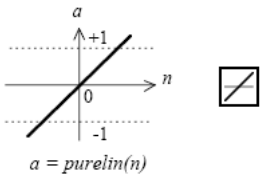
- ADALINE (Rede Linear)
  - igual à perceptrão mas com a função linear na saída;
  - pode ter vários nós;
  - apenas resolve problemas separáveis de forma linear;
  - é treinado pelo algoritmo de minimização do erro quadrático médio (Widrow-Hoff).



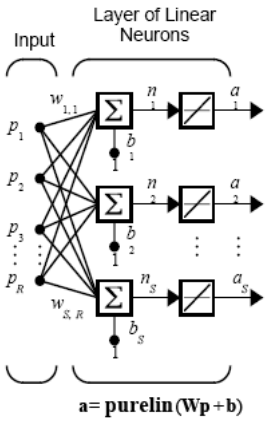
$$Output = \sum (input_i \times w_i) + b$$

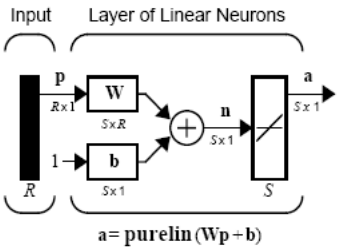
Tipologia de Redes Neuronais
Nó Linear





Tipologia de Redes Neuronais
Rede Linear





Where... R = numberof elements in input vector

S = numberof neurons in layer

Tipologia de Redes Neuronais
Regra de Aprendizagem

Minimização do erro quadrático médio:

Para os pares de entrada (p_i) saída (t_i)

$$\{\mathbf{p}_1,\mathbf{t}_1\},\{\mathbf{p}_2,\mathbf{t}_2\},\ldots,\{\mathbf{p}_Q,\mathbf{t}_Q\}$$

O erro quadrático médio (mse)

$$mse = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} (t(k)-a(k))^2$$

O algoritmo ajusta os pesos (W) e os desvios (b) de forma a minimizar este erro.

Como a função do erro é quadrática pode haver um mínimo global, vários mínimos locais ou mesmo nenhum mínimo, dependendo dos valores das entradas (p).

Aprendizagem
Regra Delta ou Widrow-Hoff

- Erro

$$e_k(n) = t_k(n) - y_k(n)$$

n-iteração
t-target (alvo)
y-saída
k-nó

Minimizar a função de custo ou performance (Regra delta)

$$E_k(n) = \frac{1}{2}e_k^2(n)$$

Variação do peso w

$$\Delta w_{kj}(n) = \eta E_k(n) x_j(n)$$

η – taxa de aprendizagem

Aprendizagem
Regra Delta ou Widrow-Hoff

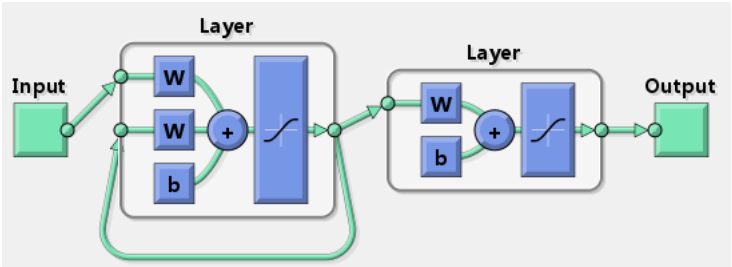
O ajuste ao peso w de uma sinapse de um nó é proporcional ao produto dos sinais de erro e de entrada da sinapse em questão.

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$

O novo valor do peso w é igual ao antigo mais a variação.

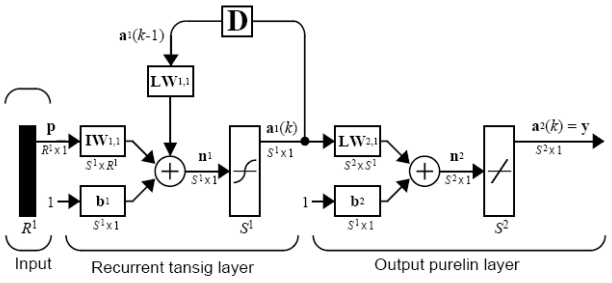
Tipologia de Redes Neuronais

- Rede Recorrente
  - Trata-se de uma rede feed-forward com pelo menos uma ligação para trás.
  - Dividem-se entre redes de Elman e redes de Hopfield.



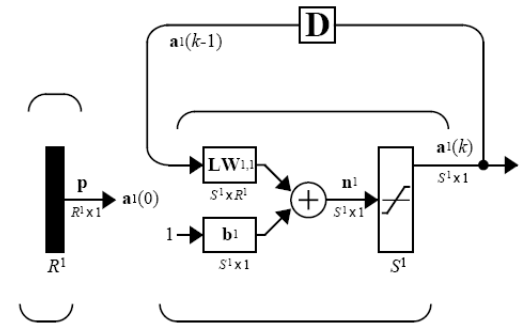
Rede Recorrente – Rede de Elman

- Permitem gerar e reconhecer padrões temporais e padrões espaciais.



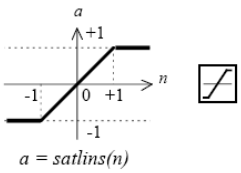
$\mathbf{a}^1(k) = \mathbf{tansig}(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{LW}^{1,1}\mathbf{a}^1(k-1) + \mathbf{b}^1)$ $\quad \mathbf{a}^2(k) = \mathbf{purelin}(\mathbf{LW}^{2,1}\mathbf{a}^1(k) + \mathbf{b}^2)$

Rede Recorrente – Rede de Hopfiel



$\mathbf{a}^1(0) = \mathbf{p}$ **and then for** $k = 1, 2, \ldots$

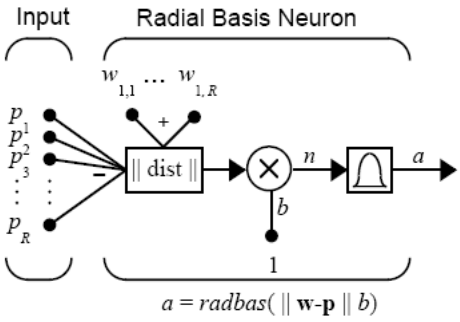
$\mathbf{a}^1(k) = \mathbf{satlins}(\mathbf{LW}^{1,1}\mathbf{a}^1(k-1)) + \mathbf{b}^1)$

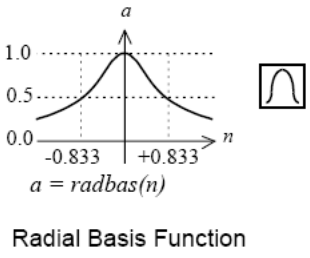


a = satlins(n)

Tipologia de Redes Neuronais

- Redes de função de base radial
  - Requerem mais nós que uma rede feed-forward standard mas é treinada em menos tempo.
  - Funciona bem com muitos vectores de entrada.

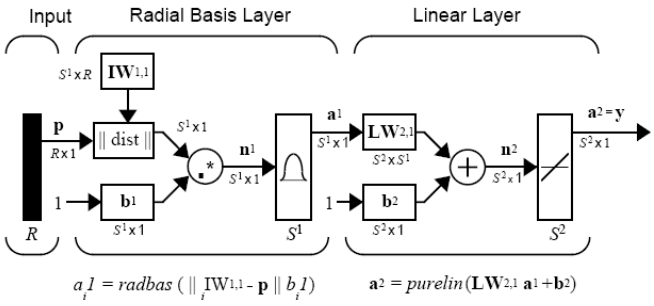




Radial Basis Function

Rede Radial

- Rede sempre com duas camadas



Where...

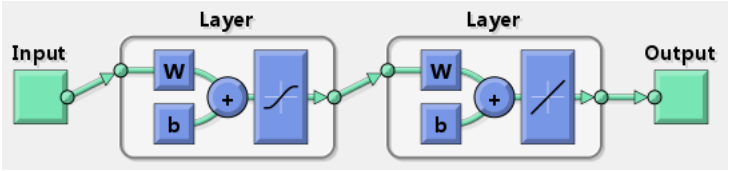
R = number of elements in input vector

$S1$ = number of neurons in layer 1

$S2$ =number of neurons in layer 2

Tipologia de Redes Neuronais

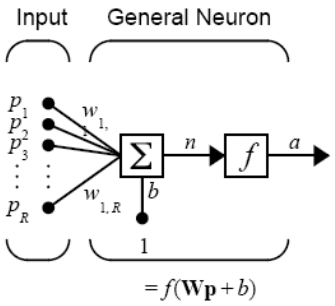
- Rede feed-forward
  - Com os algoritmos backpropagation passou a ser possível usar redes com ligação de neurónios em multi-camada.
  - Usam funções de activação não lineares que permitem resolver problemas não lineares.



Rede Feed-forward

- A soma pesada das entradas mais o desvio forma a entrada de uma função f.

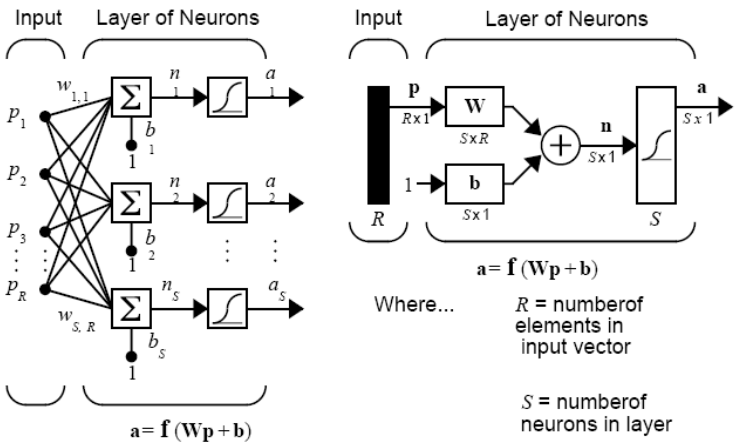
Rede com um nó



$$= f(\mathbf{W}\mathbf{p} + b)$$

Rede Feed-forward

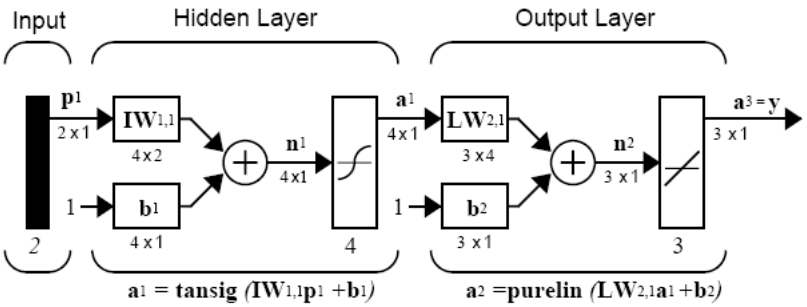
Rede com uma camada e vários nós



$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

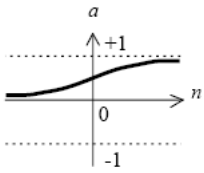
Rede Feed-forward

Rede Multi-camada



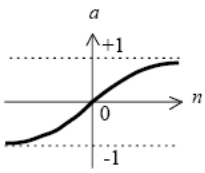
$$\mathbf{a}^1 = \text{tansig}\,(\mathbf{IW}^{1,1}\mathbf{p}^1 + \mathbf{b}^1) \qquad \mathbf{a}^2 = \text{purelin}\,(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2)$$

Rede Feed-forward – Funções de Activação



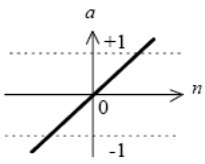
$a = logsig(n)$

Log-Sigmoid Transfer Function



$a = tansig(n)$

Tan-Sigmoid Transfer Function



$a = purelin(n)$

Linear Transfer Function

Backpropagation

- Os algoritmos de backpropagation (retro-propagação do erro) vieram permitir a generalização da regra Widrow-Hoff a redes multi-camada com funções não lineares mas deriváveis.
- Backpropagation é um algoritmo de gradiente descendente, tal como a regra Widrow-Hoff, em que os pesos das sinapses se movem na direcção do gradiente negativo da função performance (custo).
- O termo backpropagation refere-se ao sentido em que o gradiente é determina em redes multi-camada (feed-forward) não lineares.
- Outras variações do algoritmo standard baseiam-se noutras técnicas de optimização como o gradiente conjugado e os métodos de Newton.
- Quando treinada adequadamente a rede tende a dar respostas razoáveis perante situações de entradas nunca vistas.
- Tipicamente, uma nova entrada tende a ter uma resposta similar à resposta adequada de uma entrada similar da usada no treino.

Algoritmo Backpropagation

$$w(n+1) = w(n) - \eta g_k(n)$$

- Em que $g_k(n)$ é o gradiente da função performance no nó k. Obtido por num processo de derivadas parciais para os nós das camada intermédias.
- O sinal – faz o algoritmo percorrer a função performance no sentido do gradiente negativo.
- Os desvios b_k são ajustados como mais uma peso w_k com entrada unitária.
- O algoritmo pode ser implementado em dois modos:
  - Modo incremental – os pesos são actualizados após a aplicação de cada vector de entrada.
  - Modo em lote (batch) – os pesos são actualizados após a aplicação do conjunto completo de vectores de entrada .

Função de Treino (em Matlab)

- Net.TrainFcn=função de treino.

- **traingd** – gradiente descendente em lote

  Os pesos são ajustados na direcção do gradiente negativo da função performance, em lote.
- **traingdm** – gradiente descendente com momento em lote

  O momento permite que a rede responda não apenas ao gradiente local mas também aos valores recentes da superfície do erro.

  Isto dá a capacidade à rede de procurar além dos mínimos locais.

Funções de Treino Rápidas

- Técnicas Heurísticas – baseadas na análise da performance dos algoritmos do gradiente descendente:
  - Momento – traingdm
  - Taxa de aprendizagem variável – traingda
  - Resilient backpropagation – trainrp
- Técnicas de Optimização:
  - Gradiente conjugado – traincgf, traincgp, traincgb, trainscg
  - Quasi-Newton – trainbfg, trainoss
  - Levenberg-Marquardt – trainlm

Escolha da função de treino

- **trainlm** – se o número de pesos não é muito elevado (<100). É muito rápido (iterações mais demoradas, mas converge em menos iterações). Usa muita memória (é possível trocar memória por tempo de computação). Pode atingir um menor erro que as restantes funções de treino. O seu interesse diminui com o aumento do número de pesos. Não é adequada para problemas de reconhecimento de padrões. (usada por defeito no Matlab)
- **trainrp** – a mais rápida para reconhecimento de padrões. Menor desempenho quando o erro é muito baixo. Baixos requisitos de memória.
- **trainsgc** – funciona bem para a generalidade de problemas, particularmente para redes com muitos pesos.
- Etc…

Critérios de Paragem do Treino

1. Por a rede ter atingido o erro alvo.
2. Por o treino ter atingido o número de iterações
3. A rede não aprende mais. Apesar disso o erro diminui no conjunto de treino (overfiting). Usa-se um conjunto de validação (também representativo do problema) para parar o treino quando o erro nesse conjunto deixar de diminuir. – Validação Cruzada

- Adicionalmente usa-se um conjunto de teste para avaliar a performance da rede.

Tratamento dos Dados de Entrada

- Factores que influenciam o problema
  - Ex: determinação do risco de contrair diabetes (factores: índice de massa corporal, perímetro abdominal, idade, hábitos alimentares, hereditariedade, prática de exercício físico, etc.)

- Representação por vectores
  - Como representar por números cada um dos factores acima?
  - Devem-se processar alguns dados de entrada de forma a que os seus números sejam mais relevantes (pensar numa escala entre 0 e 1 ou -1 e 1)

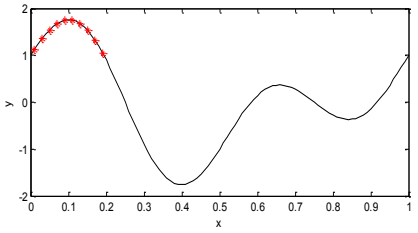
Aplicações

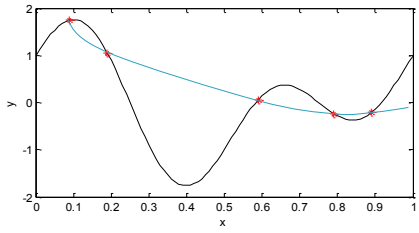
- Todas em que exista uma relação, desconhecida, entre a entrada e a saída, e existam dados para caracterizar essa relação.
- Problemas de previsão
- Previsão de séries temporais
- Classificação de padrões
- Áreas:
  - Engenharia
  - Medicina
  - Economia
  - Turismo
- etc.

Condições para a Utilização de RNA

1. Existência de um conjunto de dados exemplo do problema que serão transformados em vectores de entrada/saída.
2. Estes dados devem ser representativos do problema.
3. O seu número deve ser grande para treinar a rede (pelo menos várias vezes o número de pesos da rede).

Exemplos de 'maus' conjuntos de treino – 'más' representações do problema





Redes Neuronais no Matlab – nntool

- Usando o nntool
  - Exemplo do Xor

a	b	Y=a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

  - Vectores de entrada – cada situação corresponde a um vector vertical.
  - Redes:
    - Perceptrão (não aprende)
    - Rede Linear (train) (não aprende)
    - Rede Recorrente (aprende)
    - Rede Feed-Forward (aprende)

Redes Neuronais no Matlab (script)

- Exemplo do Xor
- Seguimento de uma curva (nnd11gn)
- Rnfuncao
- Série Temporal
- Reconhecimento de dígitos

Redes Neuronais no Matlab (script)

- Criar a RNA

```
net=newff(p,t,6)
 ou   net=newff(p,t,6,{'tansig','purelin'},'trainlm');
```

- Treinar a RNA

```
[net,TR]=train(net,p,t);
```

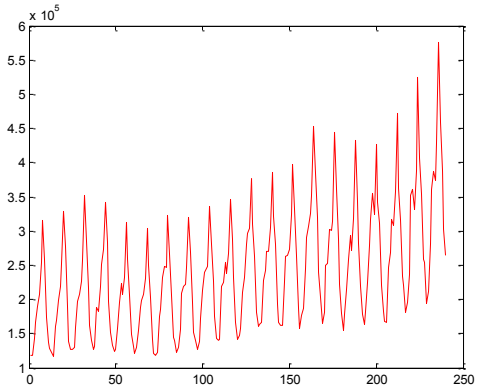
- Aplicar (simular) a RNA

```
saida=sim(net,entrada));
```

Série temporal

Dormidas mensais nos estabelecimentos Hoteleiros da Região Norte de Portugal entre Jan-87 e Dez 2007

Problema: Previsão de valores futuros



Série temporal

- Entradas: últimos 12 valores
- Saída: valor seguinte
- Conjuntos:
  - Treino: valores entre Jan87 e Dez2005
  - Validação: valores do ano 2006
  - Teste: valores do ano 2007

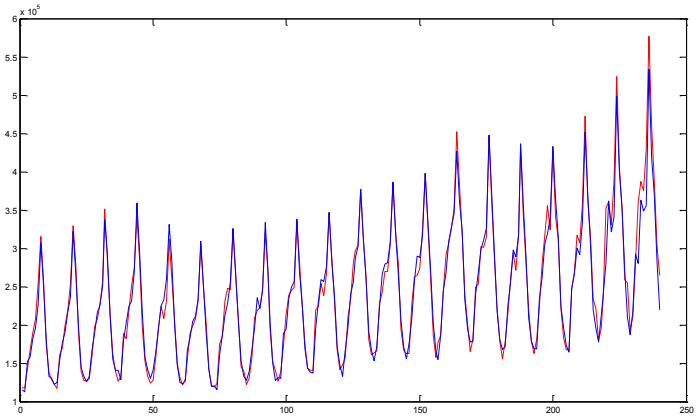
RNA Para a Série Temporal

```
fid=fopen('dormidas_2007_norte.txt','rt')
    ;
[dormidas,namostras]=fscanf(fid,'%f');
f=fclose(fid);
dormidas=dormidas';

% Preparaçao de p e t
p=zeros(12,namostras-12);
for i=1:namostras-12
   for j=1:12
      p(j,i)=dormidas(i+12-j);
   end
end
t=dormidas(13:namostras);
[linhasP,colunasP]=size(p),
```

```
% Divide dados em treino, validaçao e teste

ptreino=p(:,1:colunasP-24);
ttreino=t(1:colunasP-24);

Val.P=p(:,colunasP-23:colunasP-12);
Val.T=t(colunasP-23:colunasP-12);

Tes.P=p(:,colunasP-11:colunasP);
Tes.T=t(colunasP-11:colunasP);

net=newff(p,t,6); % Cria a rn
[net,TR]=train(net,ptreino,ttreino,[],[],Val,Tes);  %treina
saida=sim(net,p);    %simula

plot(1:colunasP,t,'r',1:colunasP,saida,'b');
```


Resultados da previsão da RNA Para a Série Temporal

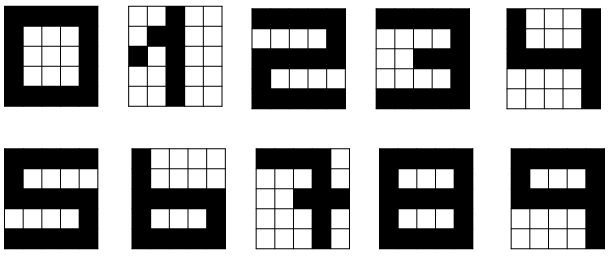
Target - Previstos



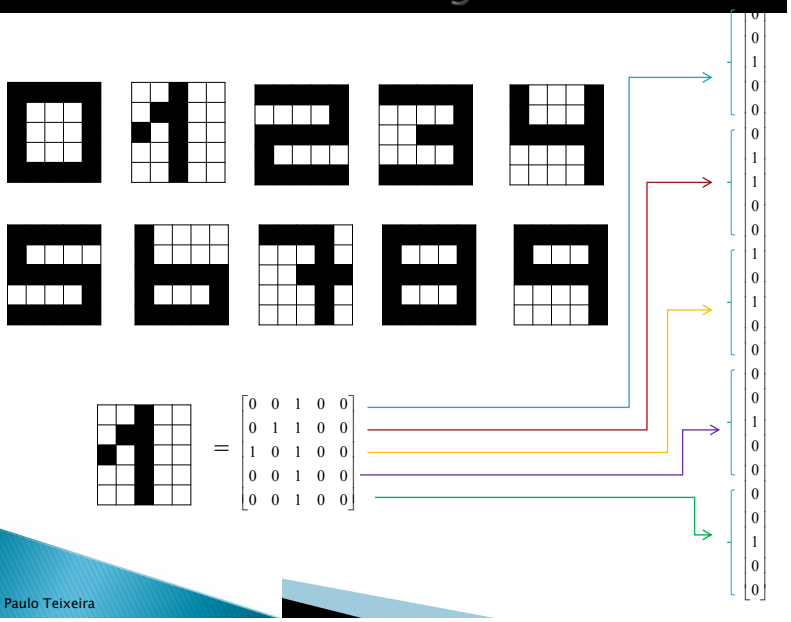
Trabalho para alunos

- Desenvolver uma rede para efectuar o reconhecimento de dígitos (letras e números)
- Criar interface gráfica com visualização da matriz de entrada e do dígito identificado.

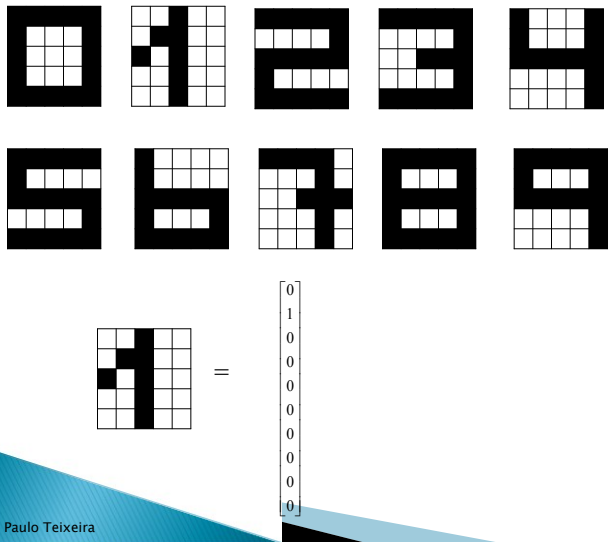
Reconhecimento de Dígitos



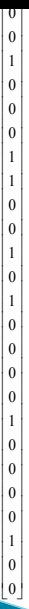
Reconhecimento de Dígitos – Entrada

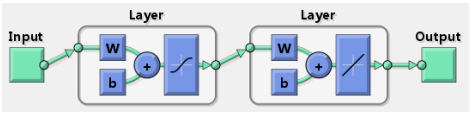


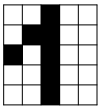
Reconhecimento de Dígitos – Saída



Reconhecimento de Dígitos

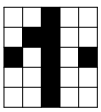




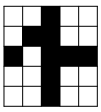


Recuperação a Falhas

- Verificar o resultado da rede para entradas com alguns bits errados:



- 1 bit (pixel) errado



- 2 bits (pixeis) errados

Bibliografia

- *Neural Networks – A Comprehensive Foundation, Simon Haykin, 1999, Prentice-Hall.*
- Neural Network Toolbox For Use with MATLAB®, *Howard Demuth, Mark Beale. Matworks, 1992-2002.*