SISTEMAS INTELIGENTES

Prática 4 – Implementação de Redes PMC
(Especificação Estrutural Usando Neural Networks Toolbox)

(**Aplicação em Estimação de Parâmetros**)

Ivan Nunes da Silva

Objetivos da Aula

- Fixar a teoria sobre redes PMC estudadas nas aulas anteriores, visando sua aplicação em estimação de parâmetros (aproximação de funções).
- Implementar exemplos aplicativos para realizar tarefas de estimação de parâmetros por meio do Toolbox de Redes Neurais do Matlab (Neural Networks Toolbox).

Problema de Estimação de Parâmetros

1. Consiste em estimar a saída desejada do processo em função de um conjunto específico de entradas.
   - As aplicações são as mais diversas possíveis, sendo que envolvem normalmente o mapeamento de processos cuja modelagem por técnicas convencionais são de difícil obtenção.
2. Diferentemente dos problemas de classificação de padrões, as saídas agora são valores reais ao invés de valores discretos.

3

Problema de Aplicação Prática

(Estimação de Parâmetros // Introdução ao Problema)

- Estudos relatam que a concentração de CO_2 (frente a uma referência) dentro de transformadores de potência pode ser estimado a partir da medição de três grandezas físicas (x_1, x_2, x_3) relacionadas ao óleo mineral isolante, ou sejam:
  - x_1 → tensão interfacial.
  - x_2 → fator de potência.
  - x_3 → quantidade de água.
- No entanto, em função da complexidade do sistema, sabe-se que este mapeamento é de difícil obtenção por técnicas convencionais, na qual o modelo matemático disponível para a sua representação não fornece resultados satisfatórios.

4

Problema de Aplicação Prática

(Estimação de Parâmetros // Definição do Tipo de Rede)

- Neste contexto, a equipe de engenheiros pretende utilizar uma rede *Perceptron* Multicamadas (PMC) como um estimador de parâmetros (aproximador universal), tendo como objetivo final de que dado como entrada os valores de (x_1 , x_2 , x_3) a mesma possa estimar (após o treinamento) o respectivo valor da variável (y) que representa a concentração de CO_2.
- O PMC deverá ter no máximo duas camadas escondidas.
- Os 10 primeiros registros da tabela de amostras de treinamento, contidos no arquivo "*treinamento.txt*", são representados a seguir.

5

Problema de Aplicação Prática

(Estimação de Parâmetros // Tabela de Treinamento PMC)

- Conjunto de treinamento referente às 10 primeiras amostras do arquivo "treinamento.txt".

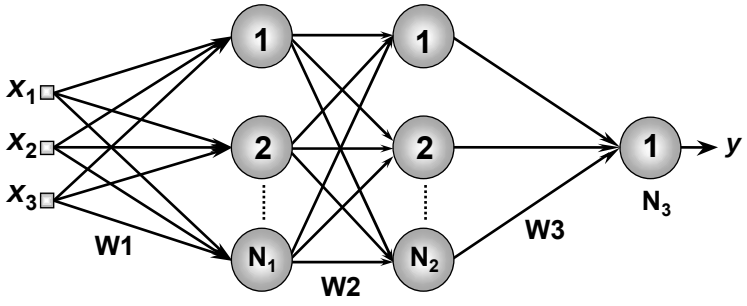
Amostra	x_1	x_2	x_3	d
1	0.1318	0.5704	0.2799	0.5849
2	-0.0365	0.2610	-0.4125	-0.7279
3	0.3952	0.1165	0.3601	0.1800
4	-0.9652	0.7845	0.7750	0.3973
5	0.4320	0.0684	0.8217	0.3008
6	0.1490	0.3721	0.7958	0.0190
7	0.1295	0.9079	-0.9618	0.3197
8	0.9890	0.4632	0.4644	-0.3275
9	-0.6249	-0.3263	0.6228	0.3045
10	-0.2408	-0.3801	-0.5591	-0.5344

6

Problema de Aplicação Prática

(Estimação de Parâmetros // Topologia do PMC)

- Como existe três grandezas físicas que estão sendo medidas, o PMC terá então três entradas { x_1, x_2, x_3 }.
- Consequentemente, a saída { y } do PMC estará então estimando, baseado nas suas três entradas, a concentração de CO_2 dentro do transformador.



Problema de Aplicação Prática

(Estimação de Parâmetros // Atividades // Parte I)

1. Carregue a matriz de treinamento referente ao arquivo "treinamento.txt".
2. Construa o vetor de entrada (vet_entrada) do PMC e o vetor de saída desejada (vet_desejado), conforme a representação a seguir:
   - vet_entrada = [0.1318 -0.0365 (…)
     0.5704 0.2610 (…)
     0.2799 -0.4125 (…)];
   - vet_desejado = [0.5849 -0.7279 (...)];
   - Imprima os vetores para checar se os mesmos estão ok. Verifique também a dimensão de cada um deles.
3. Obtenha os valores mínimos e valores máximos para cada uma das variáveis de entrada.
   - Utilize os comandos "min" e "max".

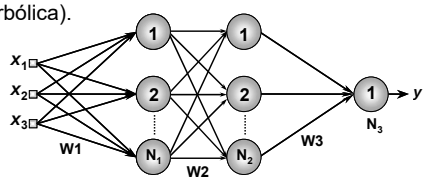
Amostra	x_1	x_2	x_3	d
1	0.1318	0.5704	0.2799	0.5849
2	-0.0365	0.2610	-0.4125	-0.7279
3	0.3952	0.1165	0.3601	0.1800
4	-0.9652	0.7845	0.7750	0.3973
(...)	(...)	(...)	(...)	(...)

Problema de Aplicação Prática

(Estimação de Parâmetros // Atividades // Parte II)

4. Crie o PMC com três camadas neurais (Toolbox, treinada com o algoritmo de "Levenberg-Marquardt", tendo a seguinte topologia:
   - 1ª Camada → 10 neurônios (Tangente hiperbólica).
   - 2ª Camada → 20 neurônios (Tangente hiperbólica).
   - 3ª Camada → 1 neurônio (Rampa).

Topologia: Perceptron Multicamadas (Feed-Forward).



- **Função: `newff`**

```
% 1º Passo: Inicializar a Rede
% ==================================================================
net = newff( [x1min x1max ; x2min x2max ; x3min x3max],... % Função que cria a rede PMC
             [N1 N2 N3],...
             {'ativação1' 'ativação2' 'ativação3'},...
              'algoritmo_treinamento' );
% [x1min x1max ; x2min x2max ; x3min x3max] → Limites das amostras de treinamento
% [N1 N2 N3] → Número de neurônios de cada camada
% {'ativação1' 'ativação2' 'ativação3} → Funções de ativação de cada camada
```


Problema de Aplicação Prática

(Estimação de Parâmetros // Atividades // Parte III)

Descrição de Parâmetros Internos (slide anterior):

```
% TIPOS DE FUNÇÃO DE ATIVAÇÃO
% ------------------------------------------------------------------
% purelin   → Linear (Rampa)
% logsig    → Logística (Sigmóide)
% tansig    → Tangente hiperbólica
% satlin(s) → Linear com saturação
% ------------------------------------------------------------------
```

```
% TIPOS DE ALGORITMOS DE TREINAMENTO
% ------------------------------------------------------------------
% traingd   → Backpropagation com Gradiente Descendente
% traingdm  → Backpropagation com Gradiente Descendente e Momentum
% traingda  → Backpropagation com Gradiente Descendente Adaptativo
% traingdx  → Backpropagation com Gradiente Descendente Adaptativo
%             e Momentum
% trainlm   → Backpropagation com Levenberg-Marquardt (Default)
% trainrp   → Backpropagation com Resiliênica (Rprop)
% ------------------------------------------------------------------
```

Problema de Aplicação Prática
(Estimação de Parâmetros // Atividades // Parte IV)

5. Especifique os parâmetros internos da rede considerando os seguintes valores:
   - 500 épocas de treinamento (*trainParam.epochs*).
   - Precisão de 10^{-4} (*trainParam.goal*).
   - Taxa de aprendizado de 0.01 (*trainParam.lr*).
   - Refresh (atualização) de tela a cada 5 épocas (*trainParam.show*).

```
% 2º Passo: Definir os Parâmetros de Treinamento
% ======================================================
net.trainParam.epochs = 500;  % Número de épocas
net.trainParam.goal = 1e-4;   % Erro final desejado
net.trainParam.lr = 0.01;     % Taxa de aprendizado
net.trainParam.show = 5;      % Refresh da tela (épocas)
```

11

Problema de Aplicação Prática
(Estimação de Parâmetros // Atividades // Parte V)

6. Efetue o treinamento da rede.

Procedimento de Treinamento do PMC

- **Função: `train`**

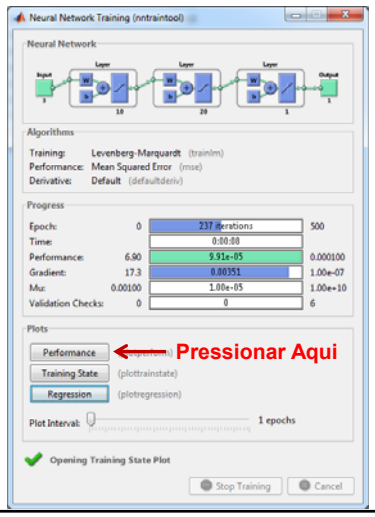
```
% 3º Passo: Treinar a Rede
% ======================================================
net = train(net, vet_entrada, vet_desejado);

% net          → Variável que recebeu a Rede Neural
% vet_entrada  → Vetor de entradas (conjunto de treinamento)
% vet_desejado → Vetor de saída desejadas (conjunto de treinamento)
```

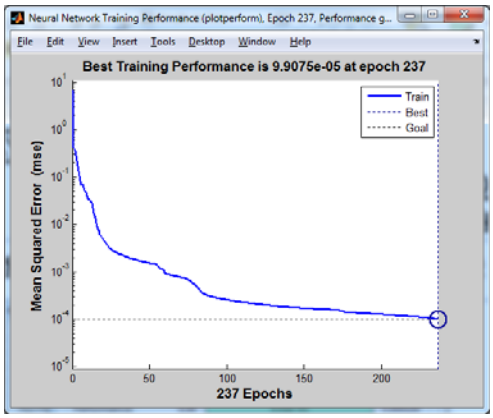
12

Problema de Aplicação Prática

(Estimação de Parâmetros // Atividades // Parte VI)

Visualização de Curva de Treinamento (Performance)





Problema de Aplicação Prática

(Estimação de Parâmetros // Atividades // Parte VII)

7. Visando validar a rede, prepare os vetores de testes baseados nas amostras contidas no arquivo “teste.txt”.

Amostra	x_1	x_2	x_3	d
1	-0.8121	-0.6348	0.5123	-0.1413
2	-0.8290	-0.7158	-0.0089	0.3242
3	-0.6046	0.5158	-0.6853	0.4541
4	0.4994	-0.6433	0.2621	-0.0818
(…)	(…)	(…)	(…)	(…)

- vet_teste_entrada = [-0.8121 -0.8290 (…)
  -0.6348 -0.7158 (…)
  0.5123 -0.0089 (…)];
- vet_teste_desejado = [-0.1413 0.3242 (…)];
- Visualize os vetores para checar se estão ok.

Problema de Aplicação Prática
(Estimação de Parâmetros // Atividades // Parte VIII)

8. Obtenha as saídas estimadas pela rede (já treinada) frente ao conjunto de teste.

Procedimento de Teste do PMC Treinado

- **Função: `sim`**

```
% 4º Passo: Testar a Rede
% ========================================================
vet_saida = sim(net, vet_teste_entrada)
% vet_saida    → Vetor com os resultados de saída do PMC.
% net          → Variável que instanciou o PMC.
% vet_teste_entrada → Vetor com dados de entrada para teste.
```

15

Problema de Aplicação Prática
(Estimação de Parâmetros // Atividades // Parte IX)

9. Imprima lado a lado os valores de saída obtidos pela rede (vet_saida) com aqueles que seriam os valores desejados (vet_teste_desejado), assim como o erro relativo frente à cada amostra (conforme formato da tabela abaixo).

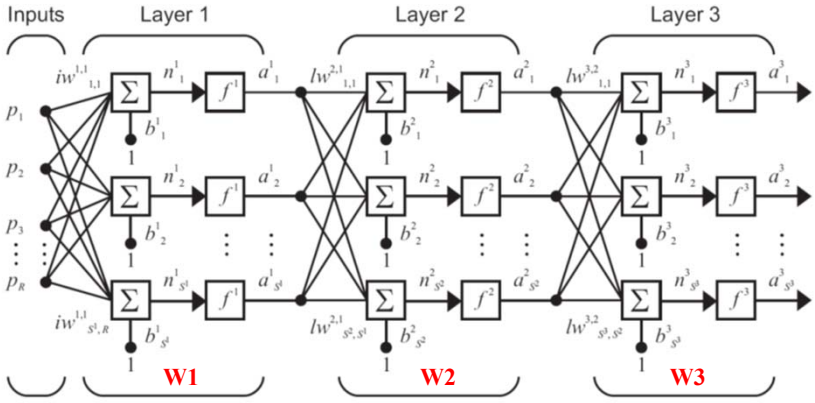
vet_saida	vet_teste_desejado	erro_relativo
0.3143	0.3242	0.0306
0.4499	0.4541	0.0093
-0.0823	-0.0818	0.0059
-0.7392	-0.7340	0.0070
-0.3562	-0.3675	0.0307
0.2269	0.2144	0.0584
-0.1272	-0.1219	0.0434
(...)	(...)	(...)

10. Obtenha o erro relativo médio frente a todas as amostras do conjunto de teste.

16

Problema de Aplicação Prática

(Extraindo os Parâmetros do PMC Treinado / Parte I)



$$a^1 = f^1(IW^{1,1}p + b^1) \qquad a^2 = f^2(LW^{2,1}a^1 + b^2) \qquad a^3 = f^3(LW^{3,2}a^2 + b^3)$$

$$a^3 = f^3(LW^{3,2} \cdot f^2(LW^{2,1} \cdot f^1(IW^{1,1} \cdot p + b^1) + b^2) + b^3)$$

1ª Camada: W1 = $IW^{1,1}$ ← net.IW{1,1} ; b1 = b^1 ← net.b{1}
2ª Camada: W2 = $LW^{2,1}$ ← net.LW{2,1} ; b2 = b^2 ← net.b{2}
3ª Camada: W3 = $LW^{3,2}$ ← net.LW{3,2} ; b3 = b^3 ← net.b{3}

p → vetor com amostra de entrada
f^1 → função ativação da 1ª Camada
f^2 → função ativação da 2ª Camada
f^3 → função ativação da 3ª Camada

17

Problema de Aplicação Prática

(Extraindo os Parâmetros do PMC Treinado // Parte II)

12. Utilizando agora somente os valores contidos nas matrizes **W1**, **W2** e **W3**, assim como nos vetores **b1**, **b2** e **b3**, implemente as instruções que nos permita utilizar a estrutura da rede (já treinada). Para tanto, implemente a seguinte função:
    - Faça uma função que receba como argumento um vetor ***x***, constituído por [x_1, x_2, x_3], retornando como resposta o valor calculado pela rede (ver slide anterior), ou seja:

$$a^3 = f^3(W3 \cdot f^2(W2 \cdot f^1(W1 \cdot x + b1) + b2) + b3)$$

    Obs. Utilize o comando "save" para salvar a workspace ou as suas variáveis de interesse. O comando "load" pode ser utilizado para recuperar os valores dessas variáveis.
13. Pegue a primeira amostra do Item 7 e verifique se a função (rede) está produzindo a mesma resposta que aquela obtida pela instrução "sim".

18