SISTEMAS INTELIGENTES

Prática 5 – Implementação de Redes PMC
(Especificação Estrutural)

(**Aplicação em Classificação de Padrões**)

Ivan Nunes da Silva

Objetivos da Aula

- Fixar a teoria sobre redes PMC estudadas nas aulas anteriores, visando sua aplicação em classificação de padrões.
- Implementar exemplos aplicativos para realizar tarefas de classificação de padrões.

Problema de Classificação de Padrões

- Problema de Classificação de Padrões consiste de associar um padrão de entrada (amostra) para uma daquelas classes que já foram previamente definidas.
- Diferentemente dos problemas envolvendo Aproximação Funcional (saídas reais/analógicas), as respostas associadas aos problemas de Classificação de Padrões estão sempre relacionadas com grandezas discretas (enumeráveis).

3

Problema de Aplicação Prática

(Classificação de Padrões // Introdução ao Problema)

- Uma determinada indústria pretende lançar diferentes tipos de vinhos para atender perfis de clientes diferenciados. Os vinhos serão classificados em três classes {*A*, *B* e *C*}.
- Para tanto, uma equipe de engenheiros e cientistas pretende aplicar um PMC como classificador de padrões, utilizando-se para tanto uma base de dados que contém 13 atributos relacionados às diversas características físico-químicas do vinho, tais como acidez, concentração de álcool, cor, teor de água, etc.
- Os 10 primeiros elementos da tabela de dados contidos no arquivo "*treinamento.txt*" são representados na tabela seguinte.

4

Problema de Aplicação Prática

(Classificação de Padrões // Tabela de Treinamento PMC)

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

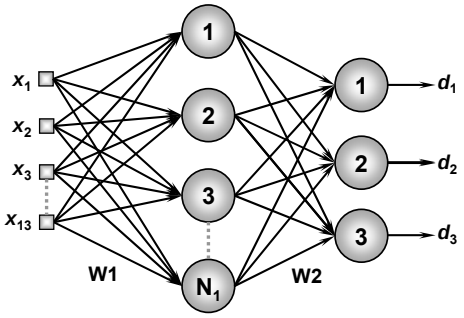
Amostra	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	d_1	d_2	d_3
1	0.7079	0.1364	0.6096	0.3144	0.4130	0.8345	0.7025	0.1132	0.5142	0.4710	0.3333	0.5861	0.7183	0	0	1
2	0.3658	0.1719	0.4439	0.6134	0.4130	0.3517	0.3692	0.3962	0.3786	0.0666	0.4715	0.6191	0.0478	0	1	0
3	0.8421	0.1917	0.5722	0.2577	0.6196	0.6276	0.5738	0.2830	0.5931	0.3720	0.4553	0.9707	0.5613	0	0	1
4	0.3526	0.0395	0.0000	0.0000	0.1957	0.3448	0.0485	0.2830	0.0032	0.0572	0.4634	0.2015	0.1726	0	1	0
5	0.4816	0.1206	0.5134	0.3814	0.5652	0.1828	0.1920	0.1509	0.1672	0.2406	0.2276	0.0073	0.2511	1	0	0
6	0.3526	0.1759	0.5027	0.7165	0.1957	0.4276	0.4452	0.5094	0.4700	0.0717	0.3333	0.5531	0.0456	0	1	0
7	0.2658	0.7036	0.5455	0.5876	0.1087	0.3862	0.2975	0.5472	0.2965	0.1126	0.2520	0.4762	0.2154	0	1	0
8	0.5711	0.2055	0.4171	0.0309	0.3261	0.5759	0.5106	0.2453	0.2745	0.2645	0.4634	0.7802	0.5506	0	0	1
9	0.3421	0.0711	0.4920	0.2784	0.3370	0.3690	0.1582	0.9434	0.0000	0.1698	0.6260	0.1465	0.2867	0	1	0
10	0.4868	0.4447	0.5562	0.4845	0.3696	0.1103	0.1857	0.2076	0.1325	0.3515	0.2114	0.0549	0.1797	1	0	0

5

Problema de Aplicação Prática

(Classificação de Padrões // Topologia do PMC)

- Como existe 13 grandezas físicas que estão sendo medidas, o PMC terá então 13 entradas { x_1, x_2, ..., x_{13} }.
- Consequentemente, a saída { d_1, d_2, d_3 } do PMC estará então identificando, baseada em suas 13 entradas, qual a classe que determinada amostra de vinho estará pertencendo.
- O diagrama representativo do modelo da rede PMC e a definição de suas classes são ilustrados abaixo.



Classe	d_1	d_2	d_3
A	1	0	0
B	0	1	0
C	0	0	1

6

Problema de Aplicação Prática

(Classificação de Padrões // Atividades // Parte I)

1. Carregue a matriz de treinamento referente ao arquivo "treinamento.txt".

Amostra	x_1	x_2	x_3	(...)	d_1	d_2	d_3
1	0.7079	0.1364	0.6096	(...)	0	0	1
2	0.3658	0.1719	0.4439	(...)	0	1	0
3	0.8421	0.1917	0.5722	(...)	0	0	1
(...)	(...)	(...)	(...)	(...)	(...)	(...)	(...)

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 = [0.7079 0.3658 0.8421 (…)
     0.1364 0.1719 0.1917 (…)
     0.6096 0.4439 0.5722 (…)
     (…) (…) (…) (…)];
   - vet_desejado = [0 0 0 (…)
     0 1 0 (…)
     1 0 1 (…)];
   - 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".

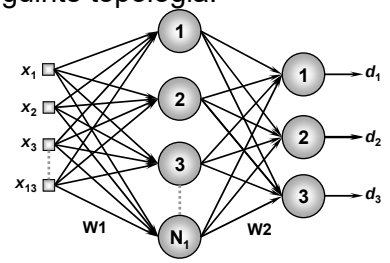
Problema de Aplicação Prática

(Classificação de Padrões // Atividades // Parte II)

4. Crie o PMC com duas camadas neurais, treinada com o algoritmo de "Levenberg-Marquardt", tendo a seguinte topologia:
   - 1ª Camada Neural → 05 neurônios (Logística).
   - 2ª Camada Neural → 03 neurônios (Linear).

Topologia: Perceptron Multicamadas (Feed-Forward).

- **Função: newff**




```
% 1º Passo: Inicializar a Rede
% ================================================================
net = newff( [x1min x1max ; x2min x2max ; (...); x13min x13max],...% Função que cria o PMC
             [N1 N2],...
             {'ativação1' 'ativação2'},...
              'algoritmo_treinamento' );
% [x1min x1max ; x2min x2max ; (...); x3min x3max] → Limites amostras/treinamento
% [N1 N2] → Número de neurônios de cada camada
% {'ativação1' 'ativação2'} → Funções de ativação de cada camada
```

Problema de Aplicação Prática

(Classificação de Padrões // 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   → Resilient Backpropagation (Rprop)
% ------------------------------------------------------------------
```

9

Problema de Aplicação Prática

(Classificação de Padrões // Atividades // Parte IV)

5. Especifique os parâmetros internos da rede considerando os seguintes valores:
   - 500 épocas de treinamento (*trainParam.epochs*).
   - Precisão de 10^{-6} (*trainParam.goal*).
   - Taxa de aprendizado de 0.5 (*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-6;   % Erro final desejado
net.trainParam.lr = 0.5;      % Taxa de aprendizado
net.trainParam.show = 5;      % Refresh da tela (épocas)
```

10

Problema de Aplicação Prática
(Classificação de Padrões // Atividades // Parte V)

6. Efetue o treinamento da rede.

Procedimento de Treinamento do PMC

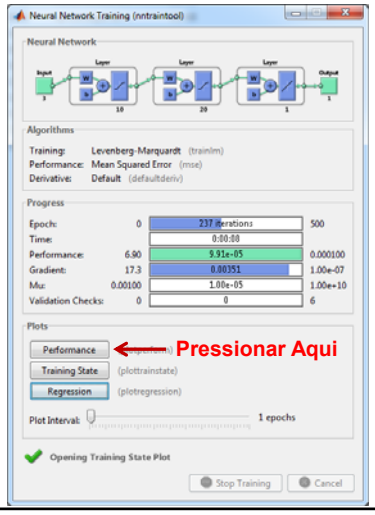
- **Função: `train`**

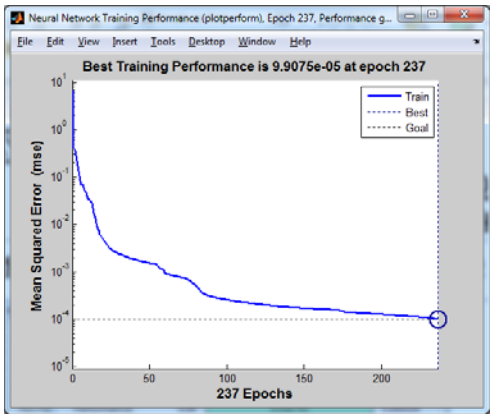
```
% 3º Passo: Treinar a Rede
% ======================================================
net = train(net, vet_entrada, vet_desejado);

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


Problema de Aplicação Prática
(Classificação de Padrões // Atividades // Parte VI)

Visualização de Curva de Treinamento





Problema de Aplicação Prática
(Classificação de Padrões // 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	d_2	d_3
1	0.1526	0.1206	0.7166	(...)	0	1	0
2	0.7368	0.1798	0.6631	(...)	0	0	1
3	0.1131	0.5929	0.2459	(...)	0	1	0
(...)	(...)	(...)	(...)	(...)	(...)	(...)	(...)

- vet_teste_entrada = [0.1526 0.7368 0.1131 (…)
  0.1206 0.1798 0.5929 (…)
  0.7166 0.6631 0.2459 (…)
  (…) (…) (…) (…)];
- vet_teste_desejado = [0 0 0 (…)
  1 0 1 (…)
  0 1 0 (…)];
- Visualize os vetores para checar se estão ok.

13

Problema de Aplicação Prática
(Classificação de Padrões // Atividades // Parte VIII)

8. Obtenha as saídas computadas 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.
```

14

Problema de Aplicação Prática

(Classificação de Padrões // Atividades // Parte IX)

9. Imprima lado a lado (por meio de uma matriz) os valores de saída obtidos pela rede (vet_saida) com aqueles que seriam os valores desejados (vet_teste_desejado), conforme formato da tabela abaixo. Efetue o pós-processamento a fim de gerar somente valores inteiros.

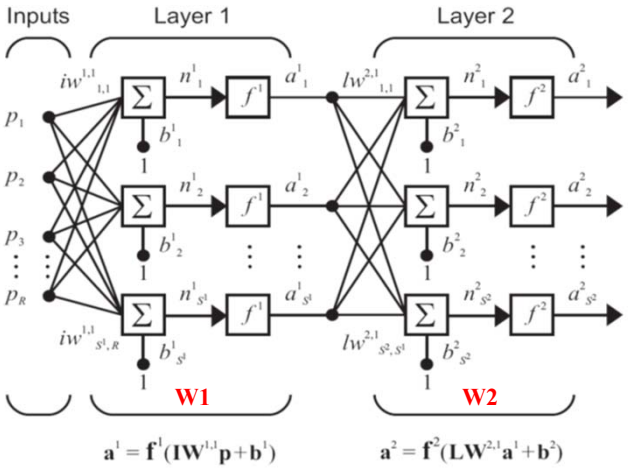
vet_saida	vet_teste_desejado
0 1 0	0 1 0
0 0 1	0 0 1
0 1 0	0 1 0
(...)	(...)

10. Faça um procedimento que dado “vet_saída” e “vet_teste_desejado”, retorne então o percentual de acertos entre os dois vetores.
11. Para propósitos de comparação de velocidade e desempenho, execute agora o treinamento usando o algoritmo “Gradiente Descendente”. Compare a velocidade com o “Levenberg-Marquardt”, assim como o percentual de acertos frente ao conjunto de teste.

15

Problema de Aplicação Prática

(Extraindo os Parâmetros do PMC Treinado)



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

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

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

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

16

Problema de Aplicação Prática
(Extraindo os Parâmetros do PMC Treinado)

12. Utilizando agora somente os valores contidos nas matrizes **W1** e **W2**, assim como nos vetores **b1** e **b2**, 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_{13}], retornando como resposta o valor calculado pela rede (ver slide anterior), ou seja:

$$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}2 \cdot \mathbf{f}^1(\mathbf{W}1 \cdot \mathbf{x} + \mathbf{b}1) + \mathbf{b}2)$$

    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".

17