Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica

Alunos: Luisa Helena Bartocci Liboni
Rodrigo de Toledo Caropreso

Data de Entrega: 25/06/2012

Redes Neurais Artificiais

(Prof. Ivan Nunes da Silva)

EPC-12

O comportamento de um processo industrial pode ser analisado levando-se em consideração várias variáveis de status relativas às fases do processo. Na tabela abaixo, fornecem-se 10 situações do comportamento do processo a partir dos valores de 16 variáveis de status.

	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}	x_{14}	x_{15}	x_{16}
Situação 1	0	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1
Situação 2	1	0	1	0	1	1	1	1	1	1	1	0	1	0	0	0
Situação 3	1	0	1	1	1	1	1	0	1	1	0	1	1	0	1	1
Situação 4	1	1	1	0	1	0	1	0	1	1	1	1	0	1	0	0
Situação 5	0	0	1	1	1	1	1	1	0	1	1	0	0	0	0	1
Situação 6	1	1	0	1	0	0	1	0	1	1	0	1	1	1	1	1
Situação 7	1	0	1	0	1	1	0	1	1	1	1	0	1	1	1	0
Situação 8	1	0	1	1	1	1	1	0	1	1	0	1	1	0	1	1
Situação 9	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	1
Situação 10	0	0	1	1	1	1	1	1	0	1	1	0	0	0	0	1

Implementar e treinar uma rede ART-1 que classifique e agrupe em classes as situações que são “parecidas”, de modo que se tenha um provável diagnóstico para uma eventual manutenção.

Classificar as entradas considerando os seguintes graus de vigilância: $\rho = 0.5$, $\rho = 0.8$, $\rho = 0.9$ e $\rho = 0.99$. Após cada simulação, indicar quantas classes estarão ativas e que situações também estarão inseridas nos respectivos agrupamentos.

Após a implementação do código no MatLab, os resultados produzidos estão dispostos na seguinte tabela:

Graus de vigilância (ρ)	Número de classes ativas	Situações por agrupamentos
0.5	3	Cluster 1 (1, 3, 5, 10) Cluster 2 (2, 4, 7) Cluster 3 (6, 8, 9)
0.8	5	Cluster 1 (1, 6) Cluster 2 (2, 7) Cluster 3 (3, 8) Cluster 4 (4, 9) Cluster 5 (5, 10)
0.9	8	Cluster 1 (1, 6) Cluster 2 (2) Cluster 3 (3, 8) Cluster 4 (4) Cluster 5 (5, 10) Cluster 6 (7) Cluster 7 (9)
0.99	8	Cluster 1 (1) Cluster 2 (2) Cluster 3 (3, 8) Cluster 4 (4) Cluster 5 (5, 10) Cluster 6 (6) Cluster 7 (7) Cluster 8 (9)

CÓDIGO FONTE

- Arquivo "amostras.txt" inclui os dados da tabela dada no anexo EPC.

- Arquivo lerArquivo.m

```
function dados =
lerArquivo( nomeArquivo, linhas,
colunas)
% Importando informações das amostras
pFile = fopen(nomeArquivo,'r');
AmostraDoArquivo = fscanf(pFile,'%f',
[linhas*colunas,inf]);
fclose(pFile);
% Organizando dados do arquivo txt
numa matriz Amostras
for i=1:linhas
    for j=1:colunas
        Amostras(i,j) =
AmostraDoArquivo((i-1)*colunas+j);
    end
end
dados = Amostras;
```

- Função principal

```
%inicialização das variáveis.
numeroDeAmostras = 10;
numeroDeVariaveisDeStatus = 16;
grauDeVigilancia = 0.5;

M = 1;
%numero maximo de clusters (neuronios)
é definido pelo número total de
%amostras. Se grau de vigilancia é 1 e
nao existem amostras repetidas,
%M será igual ao numero de amostras.
Mmaximo = numeroDeAmostras;

%passo1: inicialização das matrizes Wf
e Wb
Wf(1:Mmaximo,1:numeroDeVariaveisDeStat
us) = 1/(1+numeroDeVariaveisDeStatus);
```

```
Wb(1:numeroDeVariaveisDeStatus,1:Mmaximo) = 1;

conjPadroes = zeros(10,10); %conjunto Teta.

%passo 2: apresentar um novo padrao de entrada "X"
x = lerArquivo( 'amostras.txt', numeroDeAmostras, numeroDeVariaveisDeStatus);

for i=1:numeroDeAmostras
    %passo 3: computar as ativação usando as conexões da matriz Wf.
    u = Wf(1:M,:)*x(i,:)';

    flag = 0; %flag para sair do loop de criação/atribuição de clusters
    flagCriou = 0; %flag para criar novo cluster apenas uma vez no loop
    while (flag == 0)
        %passo 4: seleciona neuronio vencedor
        [vencedorLinha,vencedorColuna] = max(u);

        neuronioVencedor = Wb(:,vencedorColuna) & x(i,:)';

        %passo 5: executa teste de vigilancia.

[WX,v1]=size(find(neuronioVencedor));
        [XX,v2]=size(find(x(i,:)'));
        if ((WX/XX) > grauDeVigilancia)

            %passo 7: adaptação dos pessoas
            for L=1:numeroDeVariaveisDeStatus
                Wf(vencedorColuna,L) = (Wb(L,vencedorColuna)*x(i,L))/(0.5 + sum(Wb(L,vencedorColuna)*x(i,L)));
                Wb(L,vencedorColuna) = Wb(L,vencedorColuna)*x(i,L);
            end

            aux = 1;
            while (conjPadroes(vencedorColuna,aux) ~= 0)
                aux = aux+1;
            end

conjPadroes(vencedorColuna,aux) = i;
            flag = -1;

            %passo 8: habilitar todos os neurônios em F2 e voltar ao passo 2
            [lin,col] = size(u);
            for j=1:lin
                if u(j) == -1;
                    u(j) = Wf(j,:)*x(i,:)';
                end
            end

        else
            %passo 6: desabilita o neuronioVencedor temporariamente
            [lin,col] = size(u);
            for j=1:lin
                if u(j) == vencedorLinha
                    u(j) = 0;
                end
            end

            %cria um novo neuronio
            if(flagCriou == 0)
                M = M+1;
                u(lin+1) = Wf(lin+1,:)*x(i,:)';
                flagCriou = 1;
            end
        end

    end
end


%imprime resultado final.
for neuronioVencedor=1:M
    columConjPadroes = 1;
    disp('C: ');
    disp (neuronioVencedor);
    while (conjPadroes(neuronioVencedor,columConjPadroes) ~= 0)
        disp (conjPadroes(neuronioVencedor,columConjPadroes));
        columConjPadroes = columConjPadroes+1;
    end
end
```