PROJETO DE BIOENGENHARIA

PAULO HENRIQUE DOS SANTOS (1828606)*, PEDRO HENRIQUE GARCIA MACEDO (1829696)*

**UTFPR - Universidade Tecnológica Federal do Paraná, Campus Cornélio Procópio*
Avenida Alberto Carazzai, nº 1640, 86300-000, Cornélio Procópio, PR, Brasil

Abstract— This report aims at a real detection of epilepsy by means of an EEG. The signals from this EEG are treated and classified by a PMC network with time domain signals.

Keywords— Signals, Time domain, Artificial Neural Network.

Resumo— O presente relatório tem por finalidade realiar a detecção de epilepsia por meio de um EEG. Os sinais desta EEG serão analisados e classificados por uma rede PMC com sinais no domínio do tempo.

Palavras-chave— Sinais, Domínio do tempo, Rede Neural Artificial.

1 Introdução

A epilepsia é um distúrbio da função cerebral normal, sendo que suas crises podem ser observadas por meio de um eletroencefalograma (EEG). Porém, a detecção visual desses eventos não tem se mostrado muito eficiente, de modo que métodos de detecção automatizados vendo sendo estudados (Tzallas et al, 2009), dentre eles, o uso de redes neurais artificiais.

Srinivasan et al. (2005) propôs o uso de uma rede recorrente e de análises no domínio na frequência para a classificação de sinais de EEG. Porém, devido ao tempo disponível para o trabalho, será utilizada uma rede perceptron multicamadas com sinais no domínio do tempo para classificar tais sinais.

A rede PMC é composta pela camada de entrada, camada de saída e por pelo menos uma camada intermediária. Seu treinamento usa o algoritmo Backpropagation, que é baseado na diminuição do erro quadrático médio.

Os sinais utilizados neste trabalho fazem parte do banco de dados chbmit, do MIT, cujos sinais foram aquisitados de acordo com os canais mostrados na Figura 1.

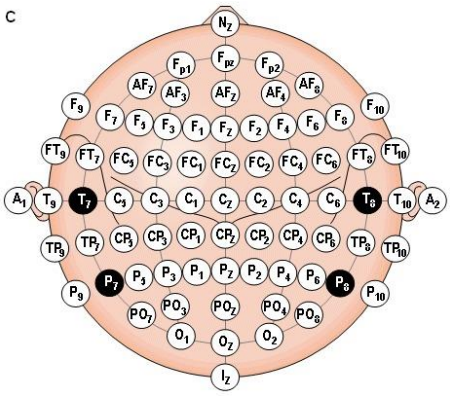

Figura 1 – Disposição dos canais do banco de dados

2 Metodologia

As arquiteturas de RNA testadas são *feedforward* do tipo PMC com funções de ativação do tipo tangente hiperbólica e os sinais são sempre analisados no domínio do tempo. Conforme Srinivasan et al. (2005) existem outras topologias mais adequadas para o problema, no entanto, por restrições de tempo para a execução deste trabalho tais arquiteturas e análise no domínio da frequência estão fora do escopo deste trabalho.

O treino da RNA é feito utilizando a Toolbox do da ferramenta computacional Matlab®. E para o treino e validação são utilizadas apenas as amostras do primeiro canal, que é o diferencial f7-t7. A escolha dele não é devido a performance nem outra característica, mas sim pelo tempo de treinamento e pela complexidade do problema.

Para melhorar o desempenho da RNA primeiramente as amostras de cada tipo são normalizadas entre si no intervalo -1 e 1, e organizadas em janelas de 512 amostras. Sendo que esse número de amostras foi determinado pelo fato das janelas serem de 2 segundos e possuírem frequência de amostragem de 256 Hz.

O treino é feito para diversas arquiteturas para posterior comparação considerando o score de cada topologia, sendo o score definido como o maior número de acertos que uma dada topologia alcança considerando seus resultados na classificação das amostras de validação.

As seguintes arquiteturas foram testadas:

1. Arquitetura com uma camada escondida e número de neurônios variado de 2 a 15 onde para cada uma das arquiteturas a rede foi treinada para 10 conjuntos diferentes de condições iniciais;
2. Arquiteturas com o número de neurônios na primeira e segunda camada iguais a: 2-1, 4-2, 6-3, 8-4, 10-5.

Os algoritmos de treino do Gradiente Descendente e de Lavenberg Marquardt foram utilizados, sendo o critério de parada, o número de épocas consecutivas sem melhora no conjunto de validação da Toolbox igual a 6.

3 Resultados

A Tabela 1 mostra os percentuais de acerto para as arquiteturas com apenas uma camada escondida com o número de neurônios variando de 2 a 15 com número total de janelas de 2766.

Tabela 1- Scores para arquiteturas com uma camada escondida

Nº de **Neurônios**	**%**	Nº de **Neurônios**	**%**	Nº de **Neurônios**	**%**
1	-	6	85,8	11	85,7
2	85,0	7	85,9	12	85,6
3	85,0	8	86,0	13	84,9
4	85,8	9	86,4	14	85,0
5	86,3	10	86,2	15	85,0

A partir da tabela vê-se que os melhores resultados foram ocorreram para 5 e 9 neurônios. Os gráficos da saída da RNA para estes casos são mostrados nas Figuras 1 e 2.

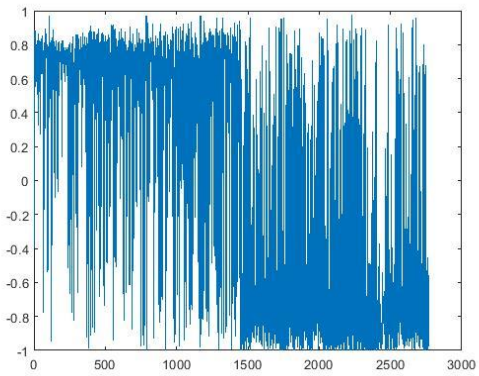

Figura 1 - Rede PMC com 5 neurônios na camada escondida.

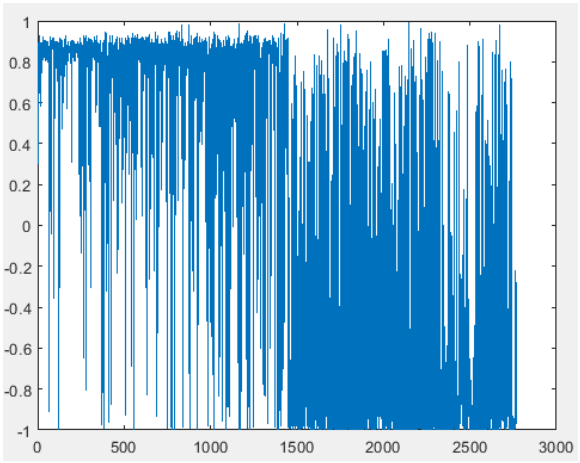

Figura 2 - Rede PMC com 9 neurônios na camada escondida.

Nas Figuras 3, 4, 5, 6 e 7 são mostradas a saída desejada e a saída da rede, obtida na etapa de validação, para cada topologia testada.

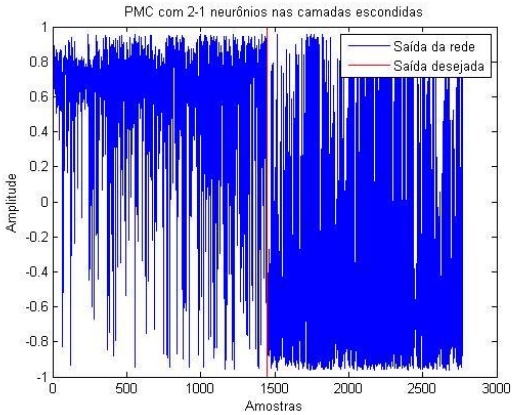

Figura 3 – Rede PMC com 2-1 neurônios nas camadas de saída

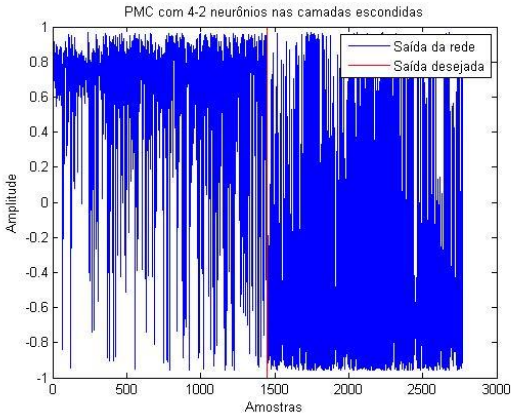

Figura 4 – Rede PMC com 4-2 neurônios nas camadas de saída

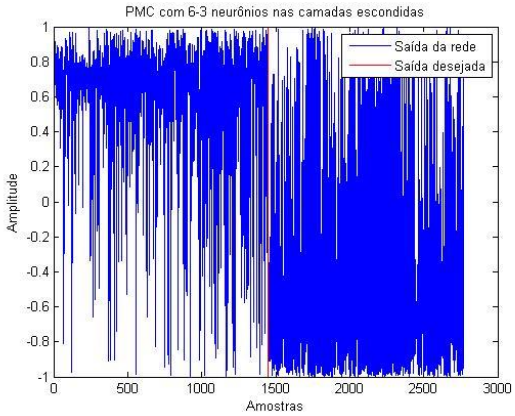

Figura 5 – Rede PMC com 6-3 neurônios nas camadas de saída

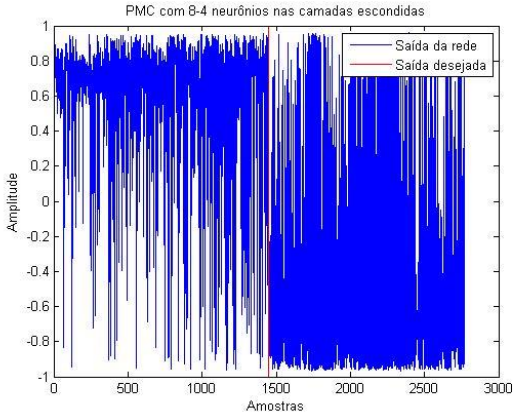

Figura 6 – Rede PMC com 8-4 neurônios nas camadas de saída

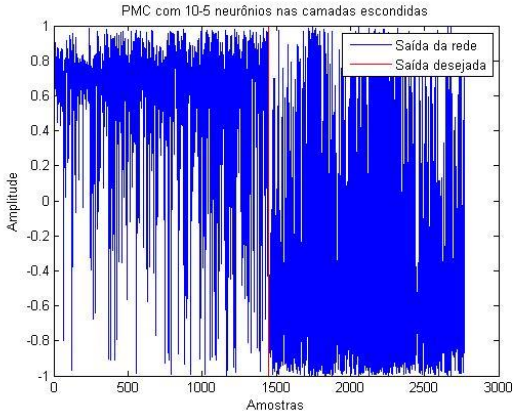

Figura 7 – Rede PMC com 10-5 neurônios nas camadas de saída

A comparação entre o número de épocas de treinamento e porcentagem de amostras classificadas corretamente na fase de treino e na fase de validação para cada uma das configurações é mostrada na Tabela 2.

Tabela 2- Porcentagem de amostras classificadas corretamente em cada topologia

Topologia	% Treino	%Validação
2-1	86,5	83,4
4-2	87,2	83,9
6-3	87,0	83,8
8-4	86,9	84,4
10-5	87,2	83,7

Observando-se a Tabela 2, nota-se que não houve muita diferença entre a porcentagem de amostras classificadas corretamente em cada uma das topologias, sendo que a máxima diferença foi de 0,7%.

4 Conclusão

Ao fim do projeto, os resultados obtidos para as RNA's chegaram a respostas satisfatórias, de acordo com o proposto no trabalho. Nos testes realizados com arquiteturas de uma camada escondida, as arquiteturas com 5 e 9 neurônios nas camadas, respectivamente, apresentaram os melhores scores. Na arquitetura seguinte, com o dobro de neurônios na primeira camada, as topologias apresentaram alto índice de amostras classificadas corretamente, tanto para a fase de treino, quanto para a fase de validação. A maior porcentagem do treino foi a topologia de 4-2 (4 neurônios na primeira camada e 2 na segunda) e 10-5 (87,2%), e porcentagem de validação foi a de 8-4 (84,4%).

Apesar dos resultados obtidos estarem próximos, ainda talvez pudesse ter alguma mudança, haja vista que o número de simulações não foi alto, e não foram alterados os dados que seriam para validação e treino.

Segundo os artigos utilizados como referência, o resultado apresentado poderia ser melhor caso fossem usadas redes mais complexas e no domínio da frequência, na qual não foram utilizadas devido ao tempo disponível para a execução deste trabalho.

Referências Bibliográficas

SRINIVASAN, et al. Artificial Neural Network Based Epileptic Detection Using Time-Domain and Frequency-Domain Feature. **Journal of Medical Systems.** 2005. v. 29. n. 6.

TZALLAS, et al. Epileptic Seizure Detection in EEGs Using Time–Frequency Analysis. **IEEE Transactions on Information Technology in Biomedicine.** 2009. v. 13. n. 5.

ANEXOS

1. Treinamento da PMC

```
%%Rede PMC classificadora

%%
%Normalização

close all
clear all
clc

load('data_norm.mat')

c1=reshape(c,[1,17768448]); %Tranformar para uma matriz linha, já que o
%comando mapminmax utiliza linha

meuc=mapminmax(c1);%Normalização

ccerto=reshape(meuc,[2961408,6]);%Voltar a dimensão anterior

% figure(1)
% subplot(2,1,1)
% plot(c)
% axis([0 3000000 -2000 2000])
% subplot(2,1,2)
% plot(ccerto)

s1=reshape(s,[1,16229376]); %Tranformar para uma matriz linha, já que o
%comando mapminmax utiliza linha

meus=mapminmax(s1);%Normalização

scerto=reshape(meus,[2704896,6]);%Voltar a dimensão anterior

% figure(2)
% subplot(2,1,1)
% plot(s)
% %axis([0 3000000 -2000 2000])
% subplot(2,1,2)
% plot(scerto)

%%
%Separação das amostras

%%dados clear
n=length(ccerto);
N=n/512; %número de janelas
nn=N*0.75; %número de janelas vezes 0.75 - 75% de amostras de treinamento
nt=nn*512; %voltar ao número de linhas

cc_trein=ccerto(1:nt,:);%amostras de treinamento para cc

%N=n/512; %número de janelas
```

```
%mm=N*0.25; %número de janelas vezes 0.25 - 25% de amostras de validação
%nv=mm*512; %voltar ao número de linhas

cc_valid=ccerto(nt+1:n,:);

%%dados s
k=length(scerto);
K=k/512; %número de janelas
kk=K*0.75; %número de janelas vezes 0.75 - 75% de amostras de treinamento
nv=kk*512; %voltar ao número de linhas

ss_trein=scerto(1:nv,:);%amostras de treinamento para ss
ss_valid=scerto(nv+1:k,:);



%%
%Treinamento da rede

cc_trein = prep_pmc(cc_trein, 512);
ss_trein = prep_pmc(ss_trein, 512);
cc_valid = prep_pmc(cc_valid, 512);
ss_valid = prep_pmc(ss_valid, 512);

% Clear = 1; Seizure = -1;
cc_trein_super = ones(length(cc_trein(1,:)),1);
ss_trein_super = -1*ones(length(ss_trein(1,:)),1);
cc_valid_super = ones(length(cc_valid(1,:)),1);
ss_valid_super = -1*ones(length(ss_valid(1,:)),1);

amostras_treino = [cc_trein ss_trein];
amostras_validacao = [cc_valid ss_valid];
d_treino = [cc_trein_super; ss_trein_super];
d_validacao = [cc_valid_super; ss_valid_super];
amostras_treino = amostras_treino((1:512),:);
amostras_validacao = amostras_validacao((1:512),:);
%varredura para o cross validation
for neuro = 1:5

    %configuração da rede neural para classificador de função
    rna= feedforwardnet([2*neuro neuro],'traingd');
    rna.layers{1}.transferFcn = 'tansig';
    rna.layers{2}.transferFcn = 'tansig';
    rna.layers{3}.transferFcn = 'tansig';
    %rna.trainParam.epochs = 5000; %Altera o número limite de épocas

    %treino da rede neural
    [rna,trpar] = train(rna,amostras_treino,d_treino');

    %teste de validação
    resultados_val = rna(amostras_validacao);
    resultados_trei = rna(amostras_treino);

    figure(neuro)
    plot(resultados_val)
    hold on
    plot(d_validacao,'r');
    titulo= cat(2,'PMC com ',num2str(2*neuro),'-', num2str(neuro),' neurô-
nios nas camadas escondidas');
    title(titulo);
```

```
    xlabel('Amostras');
    ylabel('Amplitude');
    legend('Saída da rede','Saída desejada')

    res_val=resultados_val';
    av=sign(res_val);
    hv=(av==d_validacao);
    sum(hv)

    res_trei=resultados_trei';
    at=sign(res_trei);
    ht=(at==d_treino);
    sum(ht)

end

%%
%Cross validation para uma camada escondida
Ntent = 10;
Ncamada1 = 15;
Aval = zeros(4,Ncamada1);
for camada1 = 2:Ncamada1
    arq = camada1

    for i = 1:Ntent
        %configuração da rede neural para classificador de função
        rna = feedforwardnet(arq,'traingd');
        rna.layers{1}.transferFcn = 'tansig';
        rna.layers{2}.transferFcn = 'tansig';
        %rna.layers{3}.transferFcn = 'tansig';
        rna.trainParam.epochs = 20; %Altera o número limite de épocas


        %treino da rede neural
        [rna,trpar] = train(rna,amostras_treino,d_treino');

        %teste de validação
        resultados = rna(amostras_validacao);

        AvalTent = zeros(4,1);
        for I=1:size(cc_valid,2)
            if(resultados(I)>0)
                AvalTent(1) = AvalTent(1)+1;
            else
                AvalTent(2) = AvalTent(2)+1;
            end
        end


        for I = size(cc_valid,2)+1 : size(cc_valid,2)+size(ss_valid,2)
            if(resultados(I)<0)
                AvalTent(3) = AvalTent(3)+1;
            else
                AvalTent(4) = AvalTent(4)+1;
            end
        end

        if( sum(AvalTent([1 3])) > sum(Aval([1 3],camada1)) )
            Aval(:,camada1) = AvalTent;
            RNA{camada1} = rna;
        end
```

```
        disp(sprintf('Na tentativa %d o score é %d',i,sum(AvalTent([1
3])))))
    end

    disp(sprintf('Com %d neuronios o score máximo é %d',camada1,sum(Aval([1
3],camada1))))
    [sum(Aval(:,camada1)) Aval(:,camada1)']
    disp('----------------------')

end
```

2. Função prep_pmc

```
function y = prep_pmc(a, n)
    la= length(a)/n;
    y = [];
    for ii=1:la
        y = [y reshape(a((ii-1)*n+1:ii*n, 1:6), 6*512, 1);];
    end
end
```