UNIVERSIDADE
FEDERAL DO CEARÁ

CURSO DE ENGENHARIA DE COMPUTAÇÃO

INTELIGÊNCIA COMPUTACIONAL APLICADA

PROFESSOR MÁRCIO AMORA

REDE PERCEPTRON PARA APROXIMAÇÃO DE FUNÇÃO LINEAR

ARTHUR SOUSA DE SENA MATRÍCULA: 345750

VICTÓRIA TOMÉ OLIVEIRA MATRÍCULA: 366333

SOBRAL – CE

10/05/2016

SUMÁRIO

1. INTRODUÇÃO

Uma rede neural artificial é uma tentativa de modelar as habilidades que um cérebro possui de processar informações. Basicamente, trata-se de uma rede formada por um conjunto de unidades de processamento simples que se comunicam entre si, enviando sinais através de conexões ponderadas. As unidades de processamento da rede são chamadas de neurônios. O papel de cada neurônio é bem simples, eles são responsáveis por receber as entradas de neurônios vizinhos, ou de fontes externas, e usa-los para gerar um sinal de saída.

A figura 1 mostra um modelo geral de um neurônio com n entradas, cada entrada x_i é multiplicado por um peso correspondente w_i. A entrada total para o neurônio é simplesmente a soma de todas as entradas. A função de ativação recebe como argumento o somatório das entradas. A amplitude de saída da função geralmente varia de 0 a 1, ou de -1 a 1. Três funções básicas utilizadas são: degrau unitário, semi-linear e sigmoide, como mostrado na figura 2.

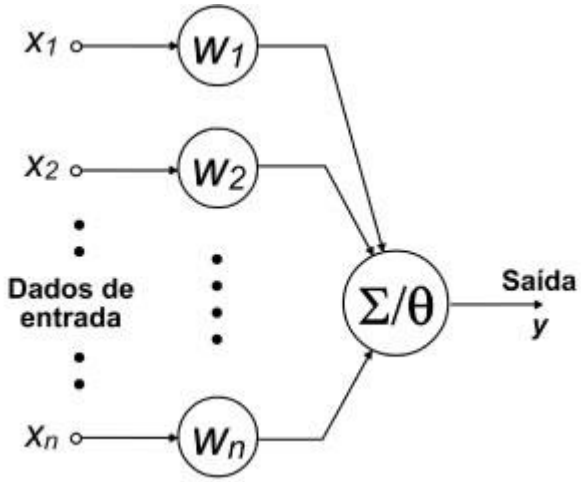

Figura 1. Modelo genérico de um neurônio artificial

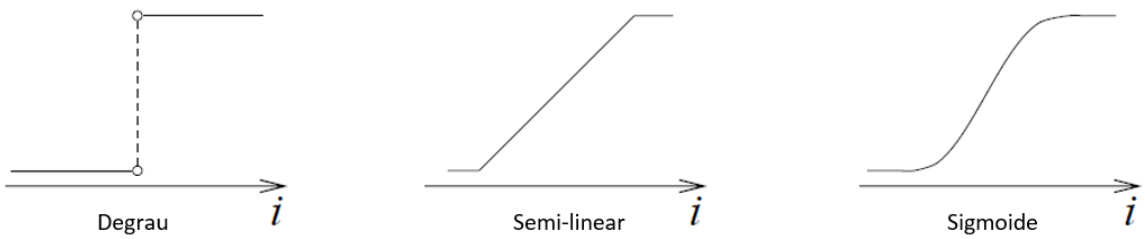

Figura 2. Funções de ativação de um neurônio

Uma das características importantes das redes neurais é a capacidade de aprendizagem e generalização que elas possuem. Elas conseguem se adaptar e melhorar seu desempenho ao longo do tempo. Essa adaptação acontece através do processo de treinamento da rede neural, por meio do fornecimento de um conjunto de padrões conhecidos. A medida que os dados de treinamento são introduzidos na rede, os pesos são modificados, de acordo com alguma regra de aprendizagem.

Uma rede perceptron consiste de uma rede neural composta por apenas uma camada de neurônios. A configuração mais simples possível para uma perceptron é aquela formada por um único neurôinio e duas entradas. Com esse modelo simplista, é possível resolver uma série de problemas linearmente separáveis. Uma possível aplicação é a aproximação de funções lineares, que é o objetivo deste trabalho. O resultado esperado desse trabalho é obter uma aproximação da função linear $y = x_1 + \frac{x_2}{2}$. Para se chegar ao objetivo desejado, será utilizado o software MATLAB para implementar tal modelo.

2. REDE PERCEPTRON

A rede Perceptron é uma rede de múltiplos neurônios e é a mais antiga de todas as redes neurais, foi à primeira máquina criada para o processamento de informações sobre redes neurais. Ela foi criada com o intuito de reconhecimentos de letras maiúsculas do alfabeto, no ano de 1957 por Franck Rosenblatt nos laboratórios das forças militares americana. Atualmente, o Perceptron é a forma de processamento de informações baseado em redes neurais, é construído com neurônios artificiais, formando um processamento de rede.

Utiliza o modelo de neurônio não-linear de McCulloch-Pitts, pois possuem a função de ativação do tipo degrau, porém contém pesos associados e bias, o modelo pode ser visto na figura 3. O algoritmo de treinamento Perceptron foi o pioneiro em modelos de treinamento supervisionados. O Perceptron é a rede neural direta mais simplificada. O modelo original fornece valores de saída que só podem assumir dois valores (verdadeiro ou falso; 1 ou 0).

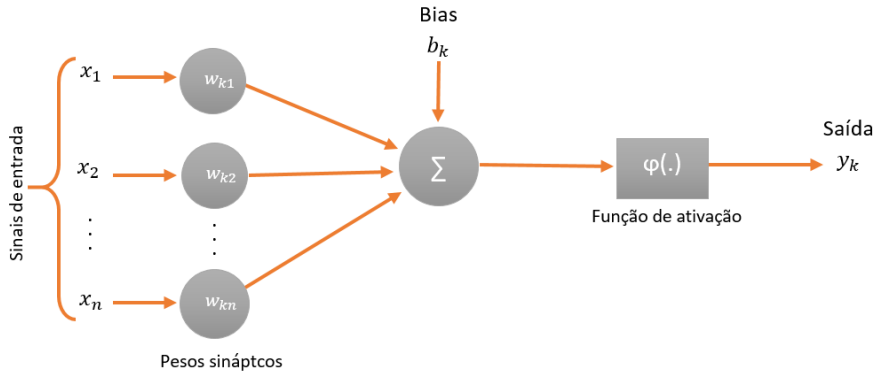

Figura 3. Modelo de uma rede perceptron

A propriedade mais importante de uma Perceptron é sua habilidade de aprender a partir de seu ambiente de funcionamento e a partir disso eleva o índice do seu desempenho. Esta rede funciona a partir da classificação linear, ou seja, todos os tipos de problemas solucionados por esta rede são linearmente separáveis.

O aprendizado de uma rede Percepetron se dá pelo seu treinamento, elas aprendem padrões e as classificações desses padrões com aprendizado supervisionado. O Percepton é formado por uma única camada de neurônios com pesos sinápticos e bias ajustáveis. Quando a entrada da rede for linearmente separável o algoritmo converge, pois ele é capaz de identificar os pesos classificando os dados corretamente.

Segundo Minsky e Papert:

> "O perceptron mostrou-se merecedor de estudo apesar de (e mesmo por causa de) suas severas limitações. Ele tem muitas características que atraem a atenção: sua linearidade; seu teorema de aprendizagem intrigante; sua clara simplicidade paradigmática como uma forma de computação paralela. Não há razão para se supor que qualquer uma dessas virtudes persista na versão de múltiplas camadas. Apesar disso, consideramos que é um importante problema a ser pesquisado para elucidar (ou rejeitar) nosso julgamento intuitivo de que a sua extensão para sistemas de múltiplas camadas é estéril." (Perceptrons de Misky e Papert (1969) – Seção 13.2)

Contudo, a observação de Misky e Papert se mostrou infundada, pois na atualidade as redes neurais avançaram de uma forma que conseguem resolver problemas não lineares por perceptrons de múltiplas camadas, treinados com o algoritmo de retropropagação.

2.1 Treinamento

O treinamento da rede Percepton funciona da seguinte forma, valores de treinamento são expostos a rede ($w_1x_1 + w_2x_2 + b$), um após o outro. Para cada padrão de treinamento x_i é dada uma saída y_i, logo em seguida é determinado o erro $e_i = d_i - y_i$, onde d_i é a saída desejada. Se a saída da rede for verdadeira, nenhuma mudança é feita, porém, se a saída for falsa, os pesos ($w_i(t+1) = w_i(t) + a * e_i * x_i$) e o bias ($b(t+1) = b(t) + a * e_i$) são atualizados pela regra de aprendizado, os valores são passados novamente e testados até que a saída seja verdadeira.

2.2 Teorema de Convergência

O teorema de convergência garante que a rede será treinada e que seus números de interações sejam finitos, garante também que o problema deve ser linearmente separável a fim de fazê-lo convergir. Os padrões a serem classificados devem ser separados entre si, pois assim garante que a superfície de decisões esteja no hiperplano.

2.3 Perceptron de Camada Simples

O perceptron de camada simples é uma rede que pode ser usada com entradas binárias e bipolares. Um dos problemas da perceptron de camada simples é que ela não é capaz de resolver problemas que não sejam linearmente separáveis.

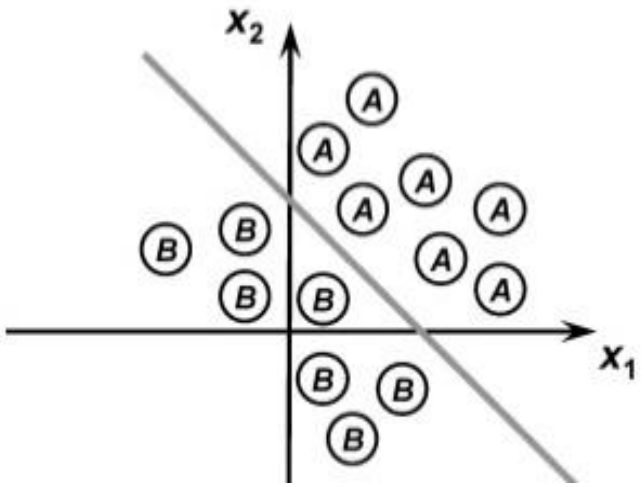

Figura 4. Perceptron Simples

2.4 Perceptron de Várias Camadas

O perceptron de várias camadas é uma extensão do perceptron abordado por Rosenblatt, é uma rede feed-foward, ou seja, cada camada se conecta à próxima camada, porém não há caminho de volta. Os problemas que não são linearmente separáveis podem ser revolvidos através dessa rede. Essa conexão ocorre com um ou mais camadas entre os nós de entrada e de saída. Essas camadas intermediárias são chamadas de camadas ocultas, abrangem nós que estão diretamente conectados aos nós de entrada e saída. É a arquitetura de rede neural mais utilizada hoje em dia.

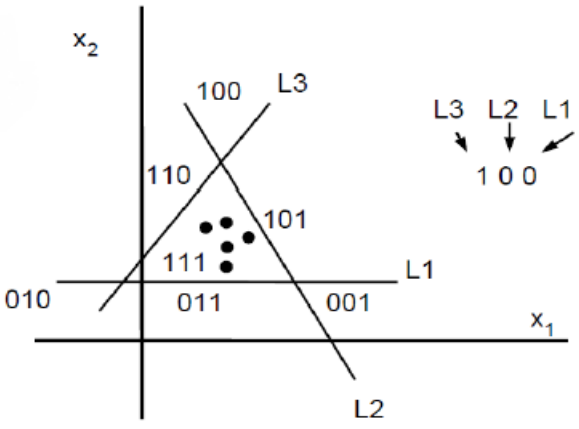

Figura 5. Perceptron de várias camadas

3. DESCRIÇÃO DO PROBLEMA

A rede Perceptron tem como uma das suas principais aplicações a aproximação de funções lineares. Redes perceptrons utilizam um algoritmo de treinamento simples, chamado de regra-delta. O princípio de funcionamento do algorítimo consiste em alterar os pesos sinápticos por meio do cálculo de erros entre a saída da perceptron e a saída desejada. O objetivo do trabalho é implementar esse algorítimo de treinamento para aproximar a função real linearmente independente, dada por: $y = x_1 + \frac{x_2}{2}$. Sua representação gráfica pode ser vista na figura 6.

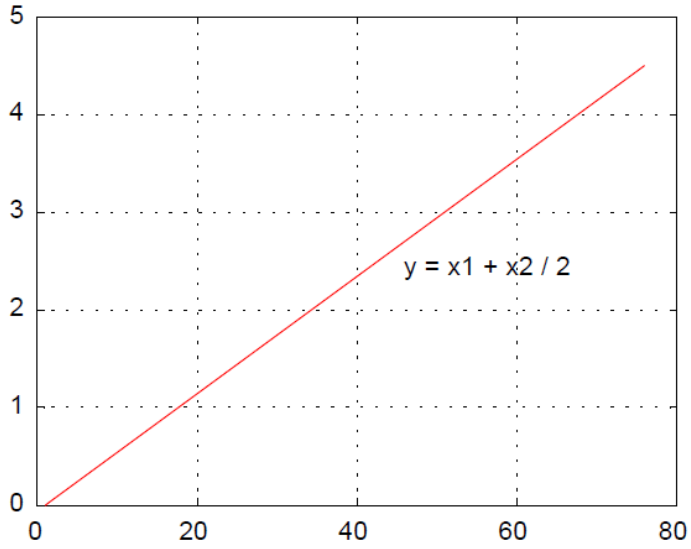

Figura 6. Função real a ser aproximada.

Para se chegar ao objetivo almejado, simplesmente será utilizada uma rede com um único neurônio. A função de ativação da rede será também uma função linear. A topologia da rede utilizada para resolver tal problema é mostrado na figura 7.

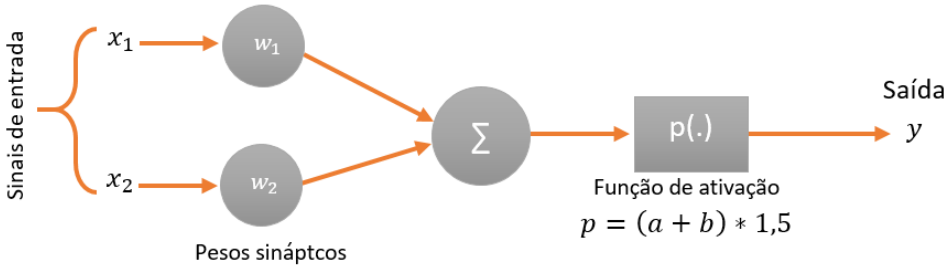

Figura 7. Perceptron utilizada para a resolução do problema.

Os seguintes parâmetros serão utilizados no algorítimo de treinamento:

- Função de ativação: $p = (x_1 w_1 + x_2 w_2) \cdot 1{,}5$;
- x_1 e x_2 com valores no intervalo [0 , 3];
- Steps de 0,04;
- Taxa de aprendizagem a = 0,1;
- Pesos inicializados com 0,1;
- Não foi utilizado bias.

4. IMPLEMENTAÇÃO EM MATLAB

De maneira resumida, o algoritmo de aprendizagem da rede perceptron para a realização da aproximação da função fornecida pode ser explicado em dez passos, como é mostrado abaixo. Os quatro primeiros passos consistem da inicialização dos parâmetros necessários para o processo de treinamento. Os passos 5 ao 9 é o trecho de repetição onde realmente acontece o treinamento da rede. O último passo consiste em mostrar os resultados obtidos após o treinamento.

Algoritmo de aprendizagem:

Passo 1. Declara função real.

Passo 2. Declara função de ativação.

Passo 3. Inicializa variáveis de peso, taxa de aprendizagem e erro.

Passo 4. Define vetores com os valores de treinamento.

Passo 5. Calcula o valor real esperado para a entrada atual.

Passo 6. Calcula o valor da função de ativação para a entrada atual.

Passo 7. Calcula o erro obtido.

Passo 8. Atualiza o valor dos pesos.

Passo 9. Testa se o erro minimo foi atingido.

Se sim, vai para passo 10.

Se não, vai para passo 5.

Passo 10. Finaliza algoritmo e exibe resultados.

Baseando-se no algoritmo acima, é possível facilmente implementar esse processo de treinamento em qualquer linguagem de programação. A implementação realizada em MATLAB possui a mesma estrutura. O código implementado segue abaixo.

Código MATLAB:

```
%-------------------------------------------------------------------------------%
% PROBLEMA:                                                                     %
%   CRIAR UMA REDE PERCEPTRON COM UM ÚNICO NEURONIO                             %
%   PARA APROXIMAR A COMBINAÇÃO LINEAR y=x1+x2/2.                               %
%   OS SEGUINTES VALORES INICIAIS SÃO UTILIZADOS:                               %
%   w1=0.1 w2=0.1   a=0.1                                                       %
%-------------------------------------------------------------------------------%

clear, clc

fprintf('Rede perceptron para aproximação da função y=x1+x2/2\n');
fprintf('-----------------------------------------------------\n\n');

func = @(x1,x2) (x1+x2/2); %Funcão real.
u = @(a,b) (a + b)*1.5; %Função de ativação da rede perceptron.

w=[0.1 0.1]; %Valores iniciais dos pesos.
a=0.1; % Taxa de aprendizagem.

x1=0:0.04:3; % Vetores utilizados para treinamento da rede perceptron.
x2=0:0.04:3; %
N=length(x1); % Armazena o tamanho do vetor x1.

e=ones(1,N); % Vetor de erros.
erro=true; % Indicador de erro.
it=1; % Contador de iterações.

E=0; % Variável utilizada para armazenar a soma dos erros.

% Vetores do histórico dos pesos são utilizados para plotar o gráfico.
histw1=ones(1,N); %Vetor para armazenar histórico dos pesos 1.
histw2=ones(1,N); %Vetor para armazenar histórico dos pesos 2.
histw1(1)=w(1); %Valor inicial de w(1) é armazenado em histw1.
histw2(1)=w(2); %Valor inicial de w(2) é armazenado em histw2.

c = colormap(jet(2000)); % Utilizado para plotar gráfico em degradê.
subplot(2,2,3);

% Loop responsável por atualizar o valor dos pesos.
% O loop é executado até que um valor aceitável de erro seja atingido.
while(erro)

        % Loops responsáveis por percorrer os vetores de treinamento.
        for i=1:N % Primeiro loop percorre os elementos de x1.
    for j=1:N % Segundo loop percorre os elementos de x2.

      % O valor real da função é calculado para as entradas
      % x1(i) e x2(j).
      d = func(x1(i),x2(j));

      % Para os valores x1(i) e x2(j),os pesos w(1) e w(2) são
```

```
        % testados na rede perceptron.
        y = u( x1(i)*w(1) , x2(j)*w(2) );

        % O erro é calculado.
        % d é o valor esperado e y é o resultado obtido na perceptron.
        e(it)=d-y;

        % A variável E receberá a soma dos valores absolutos dos erros de N iterações.
        % Onde N é o tamanho do vetor de treinamento.
        E=E+abs(e(it));

        % Os pesos são então atualizados, dado o erro obtido.
        w(1)=w(1)+e(it)*a*x1(i);
        w(2)=w(2)+e(it)*a*x2(j);

           % Os novos valores de pesos são armazenados nos vetores de
        % historico.
        histw1(it+1)=w(1);
        histw2(it+1)=w(2);


           % Plota as retas de cada iteração, mostrando a perceptron
        % convergir para a solução desejada.
        plot(u(x1*w(1) , x2*w(2)),'Color',c(it,:));
           hold on


        it = it + 1; % O número de iterações é atualizado.

    end

    % A soma de erros E é utilizada como critério de parada.
    % Se o E for suficientemente pequeno,
    % a variável 'erro' é setado para false,
    % fazendo com que seja encerrada as itereções.
    if E<0.001
          erro=false;
          break;
          end
          E=0; % A cada N iterações, a soma dos erros é zerada.

          end

end

% Trecho responsável por mostrar os resultados ----------------------------
fprintf('Função de ativação utilizada:\nu=(w1*x1 + w2*x2)*1.5\n\n');
fprintf('Pesos calculados:\nw1 = %f   w2 = %f\n\n',w(1),w(2));
fprintf('Número de iterações:\n%d\n\n',it);

fprintf('\n\nTeste da rede para valores não treinados:\n');
```

```
% Dois vetores de valores aleatórios são criados para testar
% a rede perceptron depois de treinada.
k= randi(100,10,1);
l=randi(100,10,1);
table(10,5)=ones();

%Imprime os resultados em formato de tabela
for i=1:10
            table(i,1)=k(i);
            table(i,2)=l(i);
   table(i,3)=func(k(i),l(i));
            table(i,4)=u(k(i)*w(1) , l(i)*w(2));
   table(i,5)=abs(table(i,3)-table(i,4))*100;
end

fprintf('\n x1    x2   F(x1,x2)  P(x1,x2)  Erro');
fprintf('\n --    --   --------  --------  -----');

for i=1:10
            fprintf('\n');
            for j=1:5
            if j>2&&j<5
            fprintf('%7.3f   ',table(i,j));
            else if j==5
            fprintf('%3.2f%%   ',table(i,j));
            else
            fprintf('%3.0d   ',table(i,j));
            end

            end

            end
end
fprintf('\n --    --   -------   -------   -----\n');

% Trecho responsável por plotar os gráficos ----------------------------
title('Histórico de aprendizagem');
str = strcat(int2str(it),' iterações');
text(70,4.7,str);
text(70,0.5, '0 iterações');

%Gráfico de comparação Perceptron/Função real é plotado
subplot(2,2,1);
plot(func(x1,x2),'r');
title('Comparação Perceptron e Função real');
hold on
plot(u(x1*w(1) , x2*w(2)), 'g*');
legend('Função real','Perceptron','Location','West');

%Gráfico de pesos é plotado
subplot(2,2,2);
plot(histw1,'r');
title('Histórico dos pesos')
```

```
hold on
plot(histw2);
legend(' Peso w1',' Peso w2','Location','northwest');
str1 = strcat('w1=',num2str(w(1)));
text(1800,0.7,str1);
str2 = strcat('w2=',num2str(w(2)));
text(1800,0.3,str2);

%Gráfico de erro é plotado
subplot(2,2,4);
plot(e);
title('Erro')
```

5. RESULTADOS

Como desejado, a rede perceptron se adaptou, melhorando seu desempenho ao longo das iterações. Com as configurações utilizadas, foram necessárias um total de 1977 iterações para se chegar a soma de erro mínima especificada, no caso, um valor menor que 0,001. No final do processo de aprendizagem, o código implementado mostra um relátorio com os valores finais dos pesos, o número de iterações, e exibe uma comparação entre o resultado da função real e o obtido pela perceptron para valores diferentes daqueles utilizados no treinamento. Na figura 8 abaixo, pode-se verificar esse relatório que é mostrado no prompt do MATLAB.

```
Rede perceptron para aproximação da função y=x1+x2/2
-----------------------------------------------------

Função de ativação utilizada:
u=(w1*x1 + w2*x2)*1.5

Pesos calculados:
w1 = 0.666645   w2 = 0.333341

Número de iterações:
1977



Teste da rede para valores não treinados:

 x1    x2   F(x1,x2)   P(x1,x2)  Erro
 --    --   --------   --------  -----
 17    46    40.000     40.000   0.01%
 80     9    84.500     84.497   0.26%
 32    23    43.500     43.499   0.08%
 53    92    99.000     98.999   0.07%
 17    16    25.000     25.000   0.04%
 61    83   102.500    102.499   0.11%
 27    54    54.000     54.000   0.03%
 66   100   116.000    115.999   0.11%
 69     8    73.000     72.998   0.22%
 75    45    97.500     97.498   0.20%
 --    --   -------    -------   -----
```

Figura 8. Relatório final exibido pelo código do MATLAB

Pode-se notar na figura 8 que os resultados obtidos pela rede perceptron treinada é bastante próximo aos resultados da função real, não chegando a 0,3% de erro para cada par de entradas. Verifica-se o poder de generalização, já que são testados valores que não foram treinados na rede. Além do relatório mostrado, é plotado uma série de gráficos para se ter uma melhor vizualização da eficácia do processo de treinamento da rede.

No gráfico da figura 9 pode-se ver a sobreposição das retas da função real e da rede perceptron. Através da análise do gráfico, percebe-se com clareza que o processo de aprendizagem foi eficaz e gerou uma aproximação bastante fiel à função original.

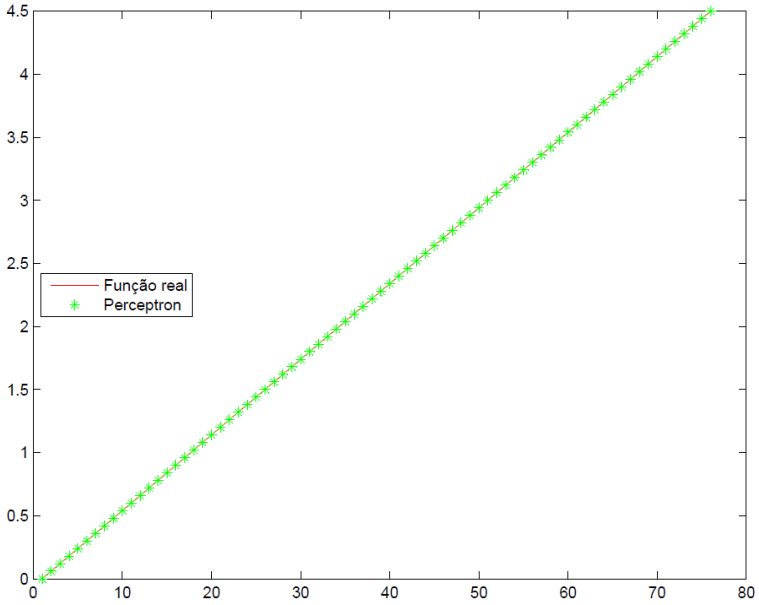

Figura 9. Comparação da rede perceptron e da função real

O gráfico da figura 10 mostra a adaptação sofrida pela rede perceptron durante o processo de aprendizagem. Verifica-se que quanto maior o número de iterações, mais próxima a rede perceptron fica da função real.

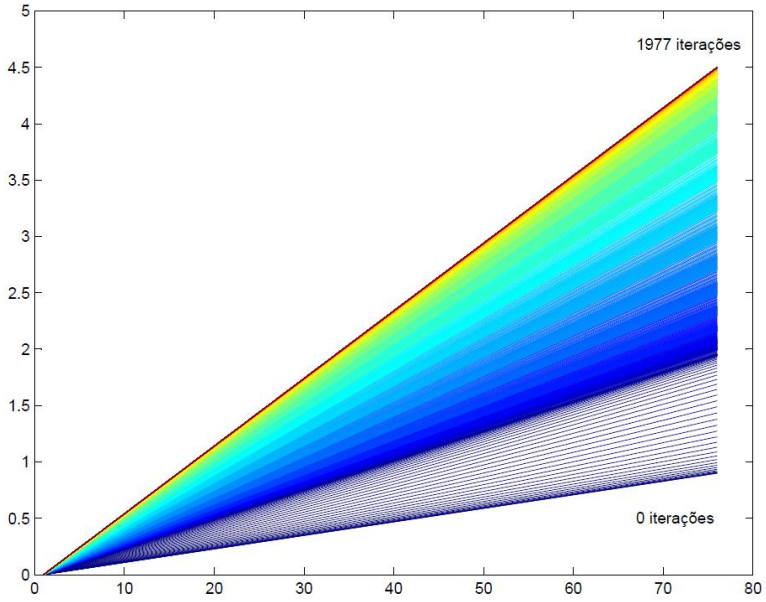

Figura 10. Hitórico de adaptação da rede perceptron durante o processo de treinamento

O gráfico da figura 11 também é bastante informativo, ele mostra a evolução dos pesos ao longo das iterações. Po de-se observar que o primeiro peso possui um comportamento predominantemente crescente, e o segundo peso fica oscilando próximo ao valor para o qual ele converge.

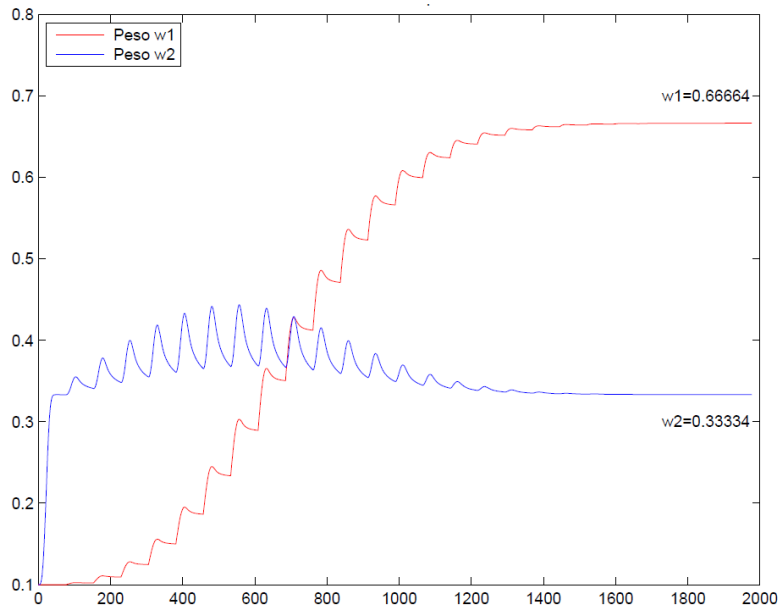

Figura 11. Evolução dos pesos da rede perceptron

E por último, na figura 12, encontra-se o gráfico de erro. Como percebe-se, o erro possui um comportamento bastante oscilatório ao longo das iterações, mas depois de um certo ponto os picos das oscilações começam a diminuir, até atingir um valor de erro bem próximo a zero, momento em que o processo de treinamento é concluido.

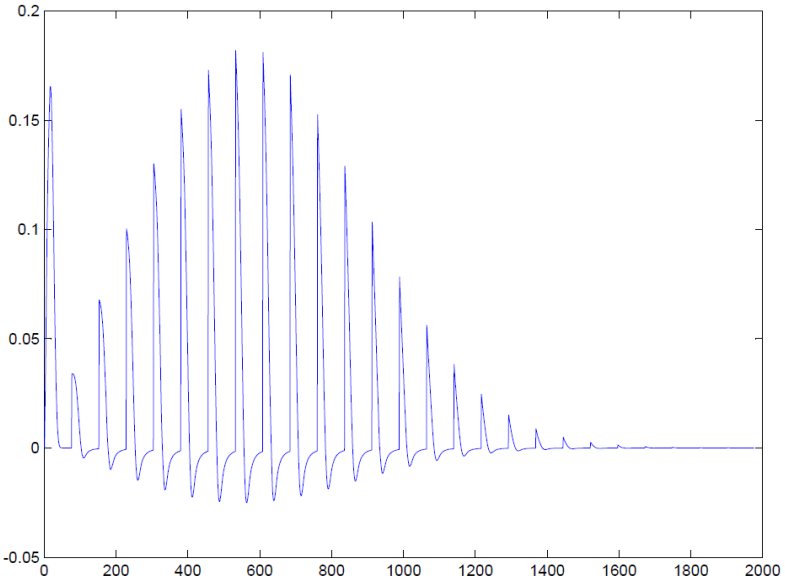

Figura 12. Gráfico de erro

6. CONCLUSÃO

Ao termino deste trabalho, podemos concluir que, as redes Perceptrons nos mostram que podemos produzir máquinas inteligentes, aptas a aprender, errar, melhorar a partir dos erros e principalmente que não são presas a instruções e programações que podem levar a falha. A única coisa que limita uma rede Perceptron é a sua própria experiência.

É necessário frisar que as redes neurais têm grandes vantagens como sua simplicidade na implementação e sua boa capacidade de generalização, porém tem como desvantagens a dificuldade de justificar suas respostas e o seu custo computacional significativo, característica que pode ser comprovada através da implentação desse trabalho, pois foram necessárias quase 2000 iterações para se chegar a uma aproximação aceitável de uma simples função linear.

A rede perceptron implementada forneceu resultados bastante satisfatórios. Ao final do treinamento, a rede se comportou de maneira muito similar a função real, com erros menores que 0,3%. Além disso, através dos gráficos plotados pudemos extrair várias informações interessantes da implementação, como a relação entre as alterações dos pesos e o aumento da eficiência da rede ao longo do treinamento. Verificou-se também o processo de convergência, através do gráfico dos erros. Os resultados obtidos satisfizeram as expectativas teóricas, portando podemos considerar que todos os objetivos do trabalho foram alcançados.

REFERÊNCIAS

KROSE, Ben; SMAGT, Patrick. **An Introduction to Neural Networks**. 8. ed. Amsterdam: University of Amsterdam, 1996.

ROJAS, Raul. **Neural Networks***:* A Systematic Introduction. Berlin: Springer-Verlag, 1996.

HAYKIN, Simon. **Neural Networks***:* a comprehensive foundation. 2. ed. Singapore: Pearson Education, 2001.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence***:* a modern approach. 3. ed. New Jersey: Pearson Education, 2010.

Neural Networks. Disponível em <https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html>. Acesso em 05 de Maio de 2016.

Aula 3: Perceptron simples. Disponível em <http://slideplayer.com.br/slide/335839/>. Acesso em 07 de Maio de 2016.

Introduzindo Redes Neurais e Perceptron. Disponível em <http://aimotion.blogspot.com.br/2008/12/artigo-introduzindo-redes-neurais-e_5497.html>. Acesso em 07 de Maio de 2016.

Perceptron. Disponível em <https://pt.wikipedia.org/wiki/Perceptron>. Acesso em 07 de Maio de 2016.