Relatório de Atividade - Redes Neurais Artificiais

David Clifte, *Mestrando em Ciências da Computação, IFCE,*

I. Introdução

NESTE trabalho é apresentado o resultado da implementação do Perceptron e do Adaline no Matlab. Inicialmente é apresentado o Classificador Linear, base do funcionamento para o Perceptron e o Adaline e então é apresentado os métodos e resultados do trabalho.

Setembro 21, 2014

II. Classificador Linear

Um classificador linear é capaz de dividir duas classes através de uma, reta, plano ou hiperplano, dependendo do número de dimensões das características. Isso é feito através do valor obtido da combinação linear das características. Dependendo do valor escalar obtido dessa combinação é feita a decisão de qual classe pertence o vetor de características ou vetor de entradas.

Uma combinação linear é o resultado do produto de cada termo de um vetor por uma constante e os resultados são então somados. Como exemplo temos a combinação linear de $a \times \overrightarrow{x} + b \times \overrightarrow{y}$ onde a e b são constantes e $\overrightarrow{x}$ e $\overrightarrow{y}$ vetores pertencentes ao mesmo espaço vetorial.

Podemos ainda generalizar combinação linear como um produto de matrizes. Dada uma matriz **A** onde cada coluna é um vetor que será combinado. Seja **C** uma matriz com os coeficientes. A combinação linear pode ser representada como:

$$A \times B = \begin{bmatrix} a_{11} & a_{1..} & a_{1n} \\ a_{..1} & a_{....} & a_{....} \\ a_{m1} & a_{m..} & a_{mn} \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_{..} \\ c_n \end{bmatrix}$$

$$= c_1 \times \begin{bmatrix} a_{11} \\ a_{..1} \\ a_{m1} \end{bmatrix} + c_{..} \times \begin{bmatrix} a_{1..} \\ a_{....} \\ a_{m..} \end{bmatrix} + ... + c_n \times \begin{bmatrix} a_{1n} \\ a_{....} \\ a_{mn} \end{bmatrix}$$

A saída obtida por um classificador linear pode ser obtida através da seguinte equação. $y = f(\overrightarrow{w} \cdot \overrightarrow{x}) = f\left(\sum_j w_j x_j\right)$, onde $\overrightarrow{w}$ é um vetor com valores reais que representam os coeficientes da combinação linear, geralmente chamado de pesos do classificador, $\overrightarrow{x}$ o vetor de características que será avaliado e y o valor escalar resultante da combinação linear.

III. Perceptron

Um perceptron é um classificador linear. Se dois conjuntos de pontos podem ser separados linearmente pode-se utilizar o perceptron para fazer a classificação, para tanto é necessário ajustar os pesos para as sinapses.

O perceptron tem duas limitações. Primeiro, os valores de saída do perceptron podem assumir somente dois valores (Verdadeiro ou Falso). Segundo, perceptrons somente podem classificar grupos de vetores linearmente separados, como dito anteriormente. Para realizar a classificação de mais de uma classe é necessário um número maior de perceptrons, entretanto ainda sim as classes devem ser linearmente separáveis.Na figura 1 temos a representação de um perceptron. De forma geral, nos neurônios artificiais os seguintes elementos estão envolvidos:

Conjunto de sinapses (W): Ligações entre neurônios. Cada ligação possui um valor (peso), que representa a sua força: os estímulos de entrada são multiplicados pelos repectivos pesos de cada ligação, podendo gerar um sinal tanto positivo (exitatório) quanto negativo (inibitório).

Combinador Linear(): Executa o somatório dos sinais produzidos pelo produto entre os pesos sinápticos e as entradas fornecidas ao neurônio. Em outras palavras, é o integrador dos sinais que chegam ao neurônio.

Como pode ser observado o vetor de entrada *in(t)* possui os seguintes valores (x1, x2, ..., xn). (w1, w2, ..., wn) representam o respectivo valor do peso do vetor de entrada.

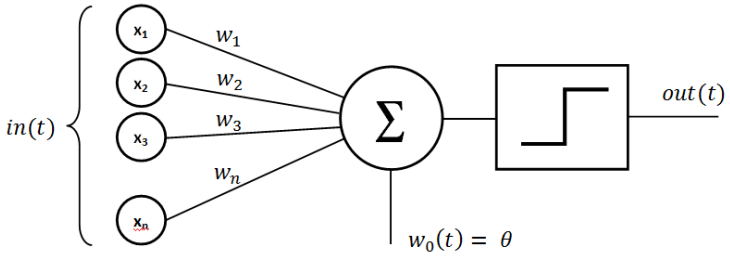

Figura 1: Modelo de um perceptron.

A. *Treinamento*

O algoritmo de aprendizagem é um método adaptativo em que o perceptron faz os ajustes necessários nos pesos para se adequar a saída desejada. Isso é feito através da apresentação repetida das n amostras de treinamento e da correção dos pesos para cada amostra. A correção é feita até que o erro das amostras de treinamentos seja minimizado. O vetor de entrada é combinado com os pesos e o resultado, um escalar, é avaliado por uma função degral.

Durante a fase de treinamento o perceptron é acionado por um vetor de entradas $\overrightarrow{x}(n)$ onde n representa a amostra apresentada ao perceptron. O sinal de saída $y(n)$ é calculado pelo perceptron e então comparado com a saída desejada $d(n)$. Com isso o erro $e(n)$ é para o perceptron é produzido.

$$e(n) = d(n) - y(n) \tag{1}$$

O $e(n)$ é utilizado para fazer a correção dos pesos utilizados pelo perceptron. Isso é feito atravéz da minimização da função de custo, definida em função do erro como:

$$\epsilon(n) = \frac{1}{2}e^2(n) \tag{2}$$

O ajuste no peso é feito atravéz da regra delta, definida por Widrow-Hoff. Na regra é realizado o ajuste dos pesos sinápticos $w(i)$, onde i representa o peso do neurônio assiciado a entrada $x(i)$.

$$\Delta W = \eta e(n)x(n) \tag{3}$$

O ajuste feito em um peso sináptico de um perceptron é proporcional ao produto do sinal de erro pelo sinal de entrada da sinapse em questão. De acordo com o valor do peso da conexão podemos verificar se ela é do tipo inibitória ou excitatória. Caso o valor do peso seja maior que zero a conexão é dita excitatória, caso contrário, inibitória.

Afim de simplificar os cálculos, é comum fazer a adição do bias ao vetor de pesos e ao vetor de entrada. Essa versão adicionada de bias é comumente chamada de versão extendida da entrada e dos pesos.
Tendo como base a figura 1 temos: x é o vetor de entrada, w é o vetor de pesos, θ é o bias e y é a saída calculada em função dos pesos e da entrada. A saída é calculada usando a seguinte equação:

$$y = f(\sum_{j=1}^{n} x_j w_j + \theta) \tag{4}$$

Onde n é o número de entradas, $f(\overrightarrow{w}^T \cdot \overrightarrow{x})$ é a função de ativação aplicada na saída calculada, no perceptron é uma função degral. Como dito anteriormente é comum adicionar o bias ao vetor de entrada e ao vetor de pesos. O valor adicionado ao bias de entrada neste trabalho foi 1. Com a versão extendida da entrada e dos pesos o calculo realizado é apenas um produto de matrizes $y = \overrightarrow{w}^T \cdot \overrightarrow{x}$.

B. *Operador AND, OR e NOT*

A base de dados foi criada para os operadores AND, OR e NOT de forma programática. Dado um conjunto de dados gerados aleatóriamente, os valores das saídas foram calculados de acordo com o operador desejado. Os detalhes para a contrução da base podem ser verificados na sessão V-A localizada no Apêndice.

1) Operador AND: Nesta sessão são exibido os resultados obtidos para o operador lógico AND. A tabela verdade do operador AND pode ser visualizada na tabela I; Os dados utilizados no treinamento estão expressos na figura 2.

Tabela I: Tabela verdade do operador AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

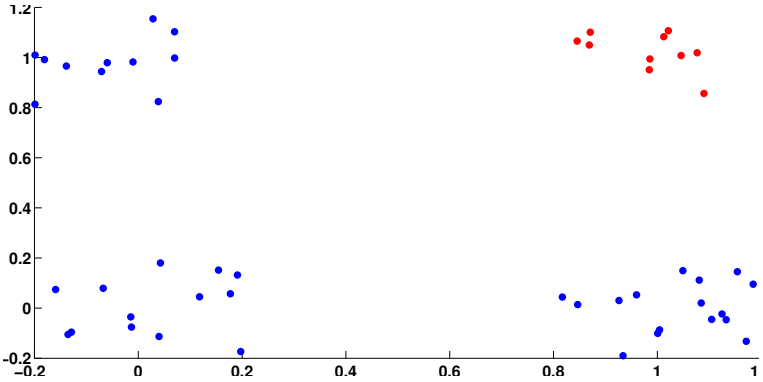

Figura 2: Dados utilizados no treinamento. Em vermelho os valores onde o operador AND tem como resultado o valor 1 em função dos valores de X1 e X2

A seguir, figura 3, é exibido o erro quadrático médio obtido durante o treinamento. É exibido também o erro global, percentual de acerto da época durante o treinamento usando as amostras de treinamento.

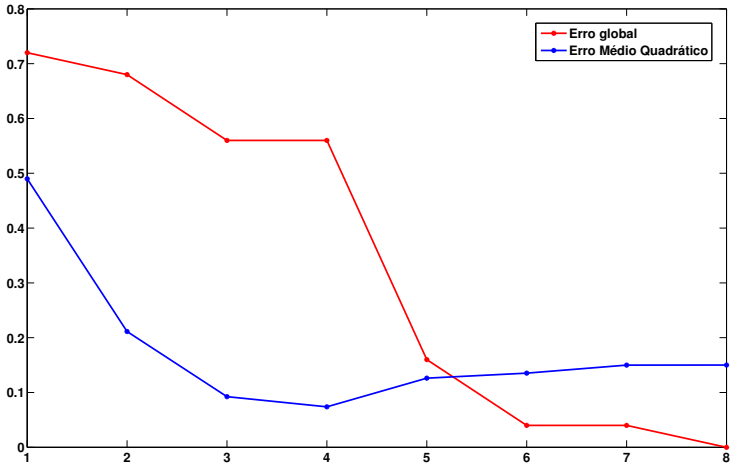

Figura 3: Erro obtido durante cada época do treinamento do operador AND.

Após o treinamento é possível calcular a região de decisão traçada pelo perceptron, figura 4.

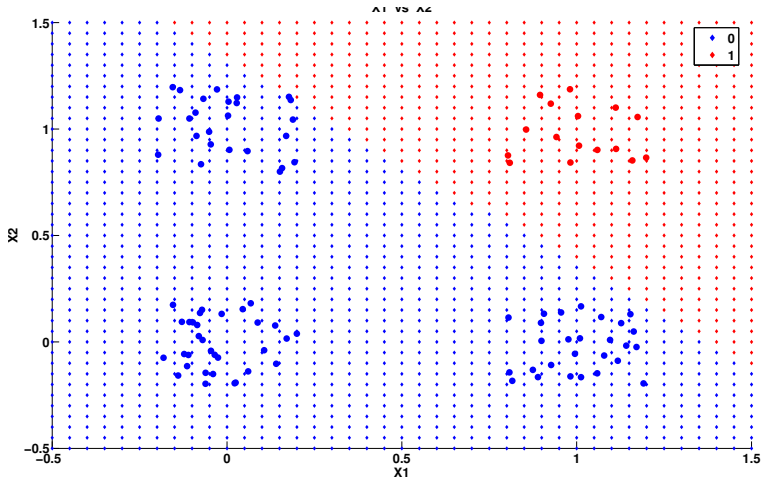

Figura 4: Região de decisão calculada para a porta AND após o treinamento. O fundo pontilhado indica a região de decisão. Em vermelho a região em que as amostras são classificadas como 1. Os pontos de raio maior são os dados de treinamento.

2) Operador OR: Nesta sessão são exibido os resultados obtidos para o operador lógico OR. A tabela verdade do operador OR pode ser visualizada na tabela II; Os dados utilizados no treinamento estão expressos na figura 5.

Tabela II: Tabela verdade do operador OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

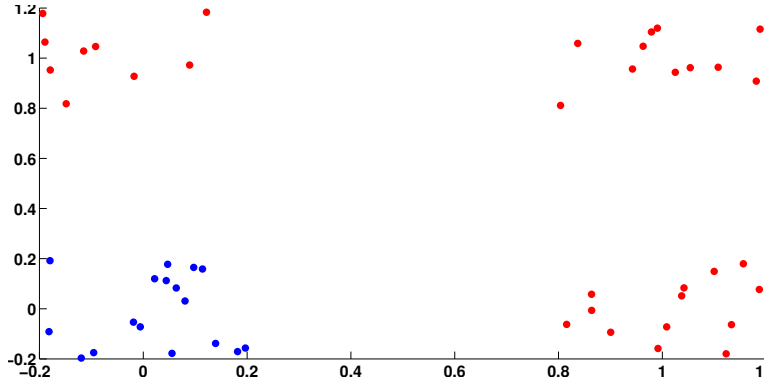

Figura 5: Dados utilizados no treinamento. Em vermelho os valores onde o operador OR tem como resultado o valor 1 em função dos valores de X1 e X2

A seguir, figura 6 é exibido o erro quadrático médio obtido durante o treinamento. É exibido também o erro global, percentual de acerto da época durante o treinamento usando as amostras de treinamento.

Após o treinamento é possível calcular a região de decisão traçada pelo perceptron, figura 7.

3) Operador NOT: Nesta sessão são exibido os resultados obtidos para o operador lógico NOT. A tabela verdade do operador NOT pode ser visualizada na tabela III; Os dados utilizados no treinamento estão expressos na figura 8.

A seguir, figura 9 é exibido o erro quadrático médio obtido durante o treinamento. É exibido também o erro global, percentual de acerto da época durante o treinamento usando as amostras de treinamento.

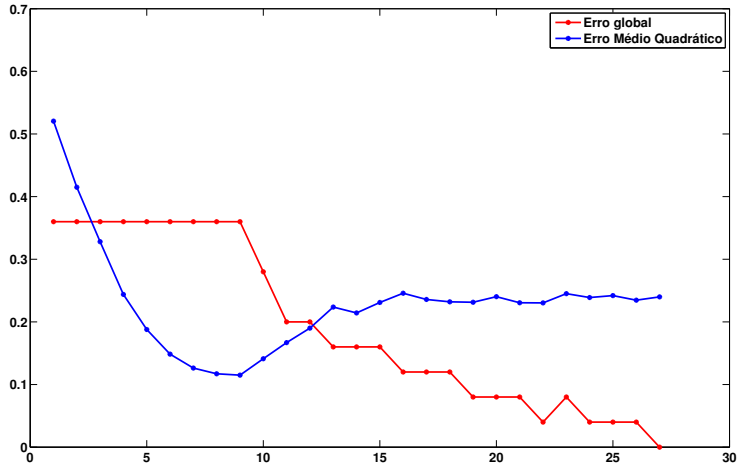

Figura 6: Erro obtido durante cada época do treinamento

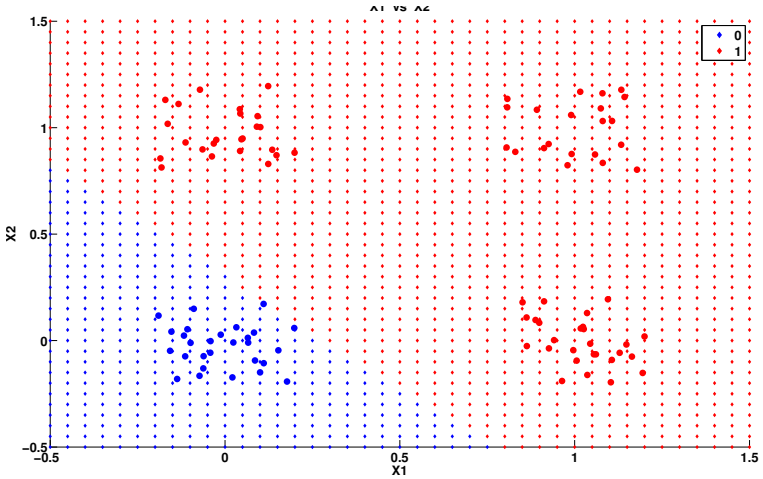

Figura 7: Região de decisão calculada para a porta OR após o treinamento. O fundo pontilhado indica a região de decisão. Em vermelho a região em que as amostras são classificadas como 1. Os pontos de raio maior são os dados de treinamento.

Tabela III: Tabela verdade do operador NOT

x	y
0	1
1	0

Após o treinamento é possível calcular a região de decisão traçada pelo perceptron, figura 10.

Diferentemente dos outros operadores, o operador NOT possui apenas um operando. Isso faz com que a região de decisão seja apenas uma linha. Na 10 pode-se perceber que para entradas acima de 0.8 os valores são mapeados para 0 ao contrário dos valores menores que 0.8 que são mapeados para 1.

C. Setosa vs Outras

Este problema consiste em classificar a espécie de Íris Setosa das outras(Íris Virgínica e Íris Versicolor). O dataset utilizado para o treinamento consiste de 50 amostras de cada uma das três espécies e a classificação será feita com base nas 4 características disponíveis. Largura e comprimento da pétala e sépala.

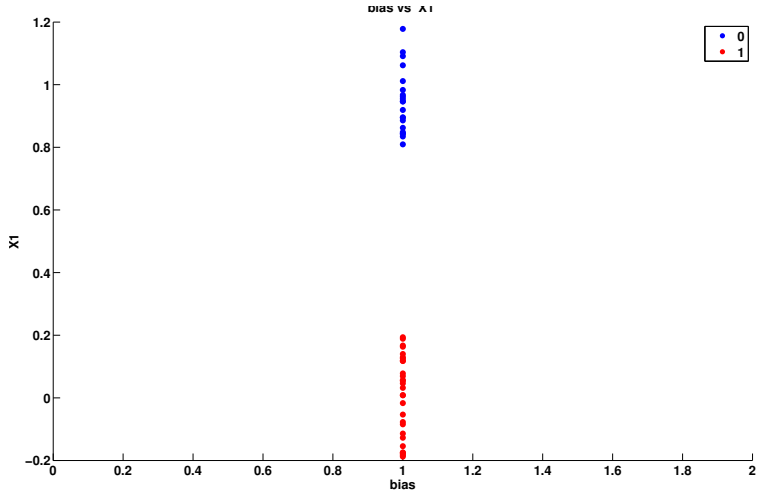

Figura 8: Dados utilizados no treinamento. Em vermelho os valores onde o operador NOT tem como resultado o valor 1 em função do valor X

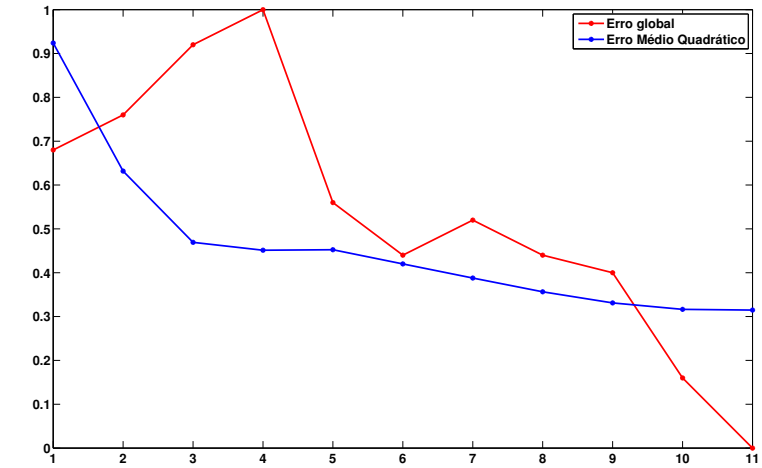

Figura 9: Erro obtido durante cada época do treinamento

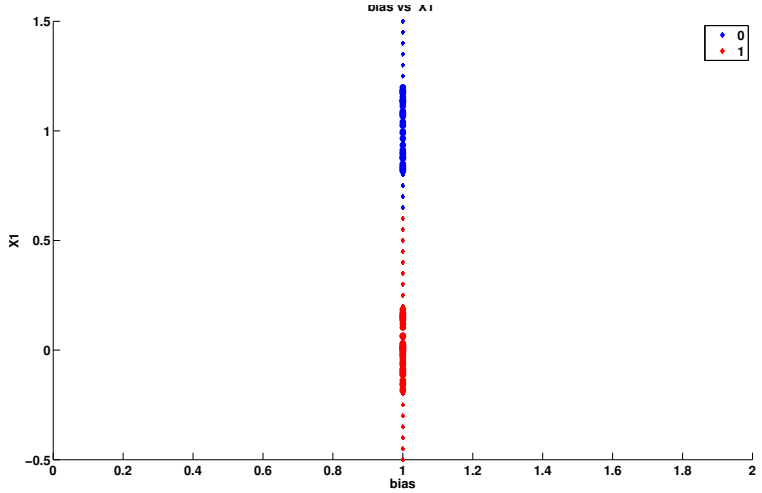

Figura 10: Região de decisão calculada para a porta NOT após o treinamento. O fundo pontilhado indica a região de decisão. Em vermelho a região em que as amostras são classificadas como 1. Os pontos de raio maior são os dados de treinamento.

Abaixo, na figura 11 temos a disposição dos dados tendo como característica as larguras da sépala e pétala. A distribuição de outras características pode ser visualizada na subsessão V-C localizada no Apêndice.

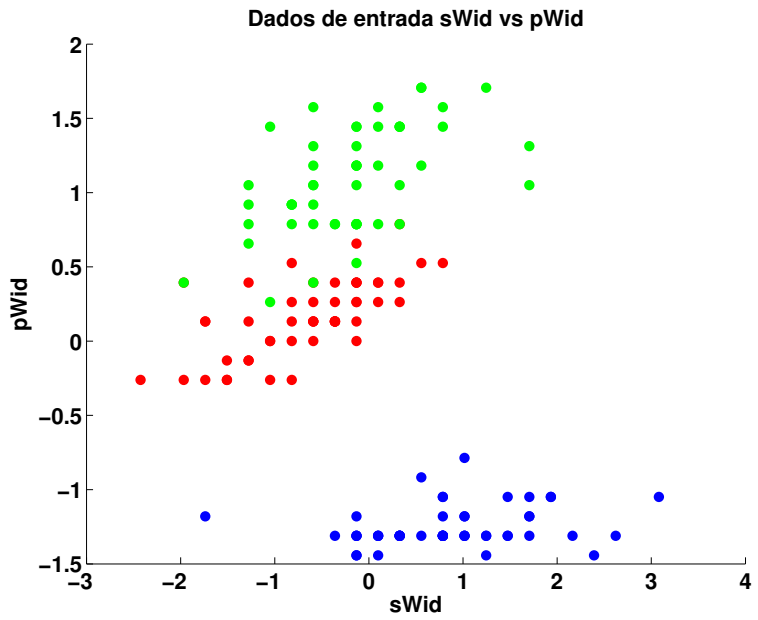

Figura 11: Largura da pétala e sépala da íris. Cada espécie é representada por uma cor. Em azul temos a Íris Setosa, em vermelho Íris Versicolor e verde a Íris Virgínica.

Foi necessário realizar alguns ajustes nos dado de entrada para que ele possa ser processado pelo perceptron. Para este problema a classe setosa recebeu o valor 1 como identificador e as outras duas classes receberam o valor 0. Veja a subsessão V-B localizada no Apêndice.

Os dados coletados foram separados em duas categorias: dados de treinamento, utilizados para o treinamento do perceptron e dados de teste, utilizados para verificar sua performance.

Após o treinamento é possível calcular a região de decisão traçada pelo perceptron, figura 12. Os pontos existentes nesta região são os pontos utilizados como teste.

A acurácia média e o desvio padrão obtido para 100, 50 e 25 repetições são exibidas na tabela IV e V Percebe-se que com apenas 50% dos dados para treinamento é possível obter-se uma grande taxa de acerto, cerca de 98%.

Tabela IV: Resultado da acurácia utilizando 20% dos dados para treinamento

Repetições	Média	Desvio
25	0.9938	0.0128
50	0.99	0.0194
100	0.9909	0.0179

Tabela V: Resultado da acurácia utilizando 50% dos dados para treinamento

Repetições	Média	Desvio
25	0.9875	0.0161
50	0.9875	0.0160
100	0.9864	0.0182

D. Adaline

O Adaline difencia-se do perceptron basicamente pelo não uso da função de ativação. Durante a fase de aprendizagem o adaline ajusta o pesos de acordo com o erro e o valor da entrada. O Adaline assim

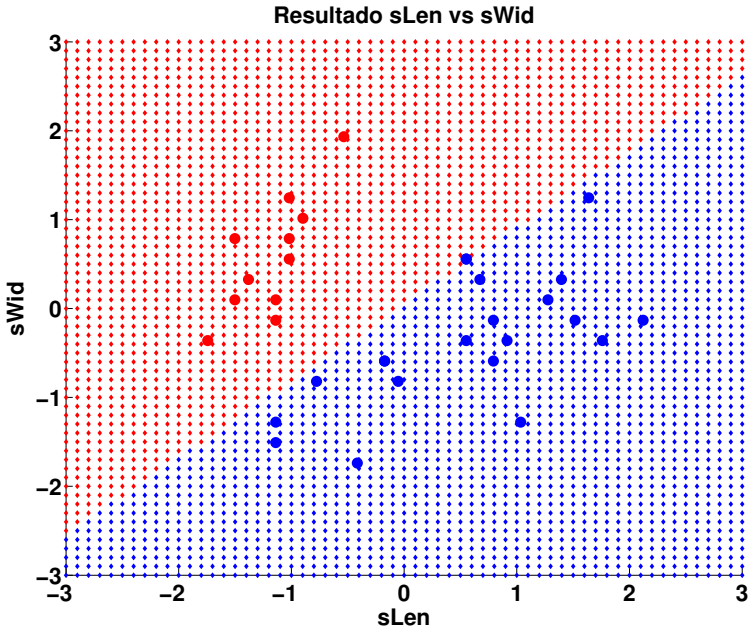

Figura 12: Região de decisão calculada para a classe setosa. No eixo vertical temos o comprimento da sépala e no eixo horizontal a largura da sépala.

como o perceptron aceita várias entradas e apenas uma saída. Se a diferença entre a saída desejada e o valor calculado for maior que um dado limiar os pesos das entradas são atualizados. O Adaline tem como principal utilização a aproximação de funções lineares. Na figura 13 é exibido um comportamento aproximado da seguinte função $f(x) = 3 \times x + 8$. Os dados exibidos na figura foram utilizados no treinamento, o qual gerou o gráfico plotado na figura 14. Na figura 15 é exibido o resultado após o treinamento do Adaline. Vale ressaltar que após o treinamento os valores dos pesos do Adaline são os coeficientes da função linear aproximada por ela.

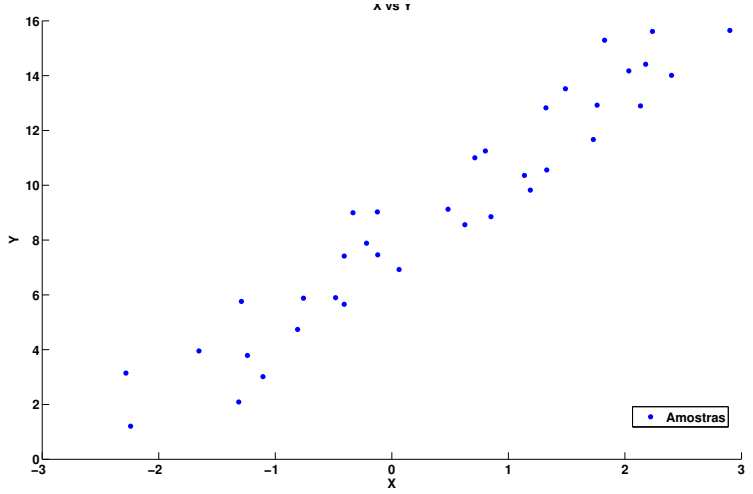

Figura 13: Dados utilizados para o treinamento do Adaline.

E. Setosa vs Virgínica vs Versicolor

Este problema consiste em classificar as espécie de Íris individualmente, Íris Setosa Íris Virgínica e Íris Versicolor. O conjunto de dados utilizados é o mesmo utilizado na subsessão III-C.

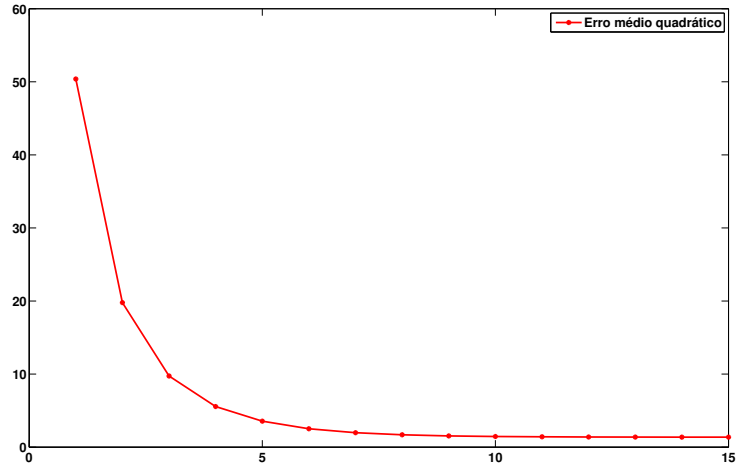

Figura 14: Erro médio quadrático gerado durante o treinamento do Adaline.

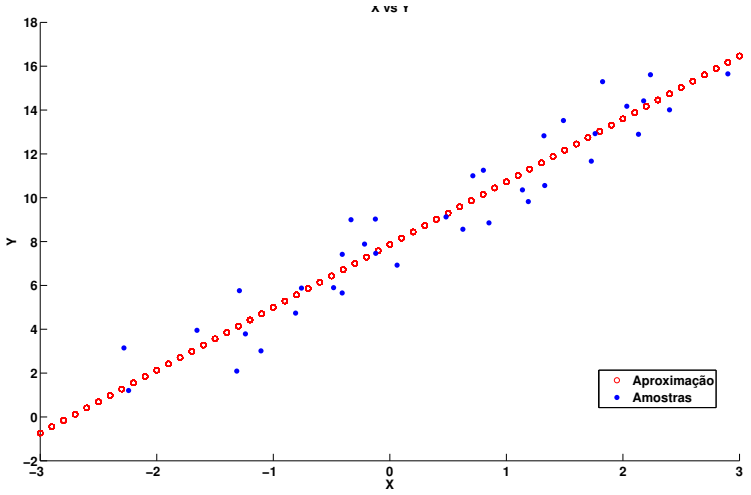

Figura 15: Resultado após o treinamento. Em vermelho temos a reta aproximada aos dados.

Na figura 11 temos a disposição dos dados tendo como característica as larguras da sépala e pétala. A distribuição de outras características pode ser visualizada na subsessão V-C localizada no Apêndice.

Os ajustes necessários para o treinamento foram feitos no momento que o dataset é carregado pela aplicação. Desta vez foram utilizados 3 perceptrons fazendo com que a saída desejada seja mapeada da forma exposta na tabela VI. Veja o Apêndice V-B para mais detalhes de como é feito o tratamento do dataset.

Para a solução deste problema serão utilizados 3 perceptrons dispostos de acordo com a figura 16.

Tabela VI: Mapeamento da saída de acordo com a classe da Íris.

Tipo	$y1$	$y2$	$y3$
Setosa	1	0	0
Versicolor	0	1	0
Virgínica	0	0	1

Neste modelo pode-se perceber que os perceptrons são independentes entre si. Cada um, assim como o problema relatado na subssessão III-C, é responsável por verificar se aquele dado de entrada pertence a classe na qual o perceptron foi treinado. Após o treinamento obtemos o seguinte gráfico do erro quadrático médio.

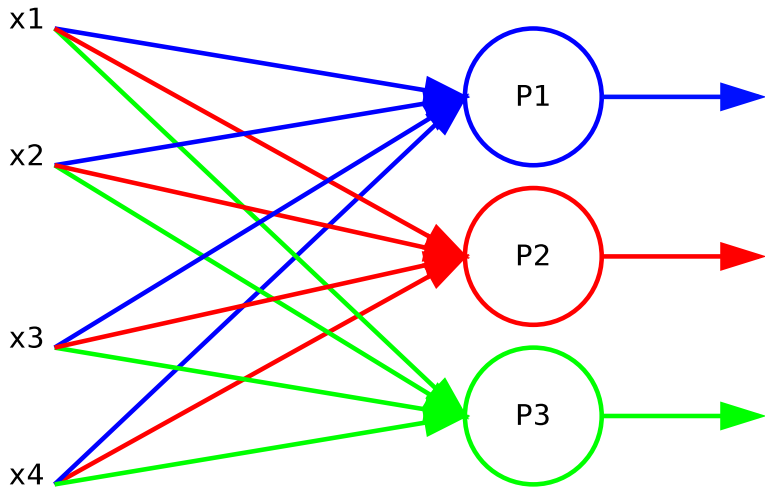

Figura 16: Perceptrons utilizados para realizar a classificação das espécies de Íris. O perceptron P1, P2 e P3 classificam a espécie setosa, virgínica e versicolor respectivamente.

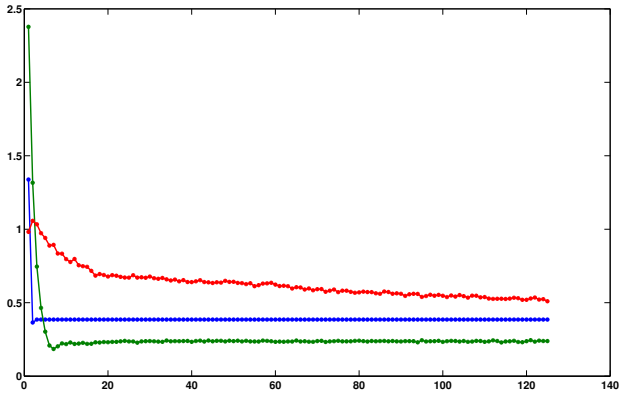

Figura 17: Em vermelho o erro médio quadrático obtido durante o treinamento para o perceptron da classe Setosa, em azul e verde as classes versicolor e vigínica respectivamente.

A acurácia média e o desvio padrão obtido para 100,50 e 25 repetições são exibidas na tabela VII e VIII Percebe-se que não foi possível obter um resultado melhor que 65 %. Este resultado é esperado tendo em vista que as classes virgínica e versicolor não podem ser separadas linearmente.

Tabela VII: Resultado da acurácia utilizando 20% dos dados para treinamento das 3 classes de Íris

Repetições	Média	Desvio
25	0.6212	0.0767
50	0.6331	0.0808
100	0.6431	0.0795

Tabela VIII: Resultado da acurácia utilizando 50% dos dados para treinamento das 3 classes de Íris

Repetições	Média	Desvio
25	0.6270	0.0877
50	0.6247	0.0778
100	0.6227	0.0807

IV. Conclusão

Este trabalho permitiu fazermos uma análise do perceptron e do adaline. Foi percebido ao longo do trabalho a grande capacidade do perceptron de realizar a separação linear de duas classes. Por parte do adaline foi dado um exemplo de como se fazer a aproximação de funções lineares dado uma distribuição de dados. Foi apresentado também o resultado da classificação de 3 classes sendo 2 delas não separável linearmente, utilizando 3 perceptrons. Foi verificado um resultado não satisfatório para esta classificação.

V. Apêndice

A. Gera Base de Treinamento

Abaixo é possível visualizar o código implementado em Matlab para gerar a base de dados para os operadores AND, OR e NOT. Através do parâmetro tipo é possível selecionar o tipo de base de dados que será gerado. As opções são AND, OR e NOT. Para a base NOT é importante ressaltar que os valores de X apresentam apenas uma dimensão. A base de dados gera uma nuvem em torno dos valores 0 e 1 com variação de até 0.2, isso é feito nos passos da linha 4 à 7. Após a criação da núvem é feita uma operação utilizando os operadores do matlab.

```
1  function [x y] = geraBD( tipo, n)
2      %%
3      %Gerando dados de treinamento aleatoriamente
4      r1 = (rand(n,2)*2 - 1)/5;                    %Gerando valores na faixa de [-0.2;0.2]
5      r2 = round(rand(n,2));                       %Gerando valores valores 0 ou 1
6      r = r1 + r2;
7      x = r;
8
9      if (strcmp(tipo,'AND'))
10         y = round(x(:,1)) & round(x(:,2));
11
12     end
13
14     if (strcmp(tipo,'OR'))
15         y = round(x(:,1))  | round(x(:,2));
16     end
17
18     if (strcmp(tipo,'NOT'))
19       x = r(:,1);
20       y = ¬round(x(:,1));
21
22     end
23
24 end
```

B. Carrega Dataset

Abaixo é possível visualizar o código implementado em Matlab para carregar os dados da Íris. De acordo com o valor da variável *modo*, a saída y, utilizada como saída deseja pelo classificador, é alterada.

```
1  %% Parametros
2  %1-setosa
3  %2-versicolor
4  %3-virginica
5
6  %modo
7  %.
8  %     Escolha 1 para classificar entre setosa e outras
9  %     Escolha 2 para classificar entre versicolor e outras
10 %     Escolha 3 para classificar entre virginica e outras
```

```
%     Escolha 4 para classificar as 3 entre si

function [ x , y ] = carregaDados(modo,path)

    %Carregando dados
    x = csvread(path);

    [m n] = size(x);

    %Ajustando saida desejada


    if(modo≥1 & modo≤3)
        y = zeros(m,1);
        y(:,1) = x(:,5)==modo;
    else
        y = zeros(m,3);
        y(:,1) = x(:,5)==1;
        y(:,2) = x(:,5)==2;
        y(:,3) = x(:,5)==3;
    end

    %apagando 5 coluna.
    x(:,5) = [];
end
```

C. *Características Combinadas para o DataSet da Íris*

Abaixo temos as caractéristicas do DataSet da Íris combinados. É fácil perceber que a classe que está em azul(Setosa) é linearmente separável em praticamente qualquer combinação de características. Diferentemente das outras duas classes, Versicolor e Vigínica que têm uma região em comum em várias características principalmente quando relacionamos o comprimento com a largura da sépala.

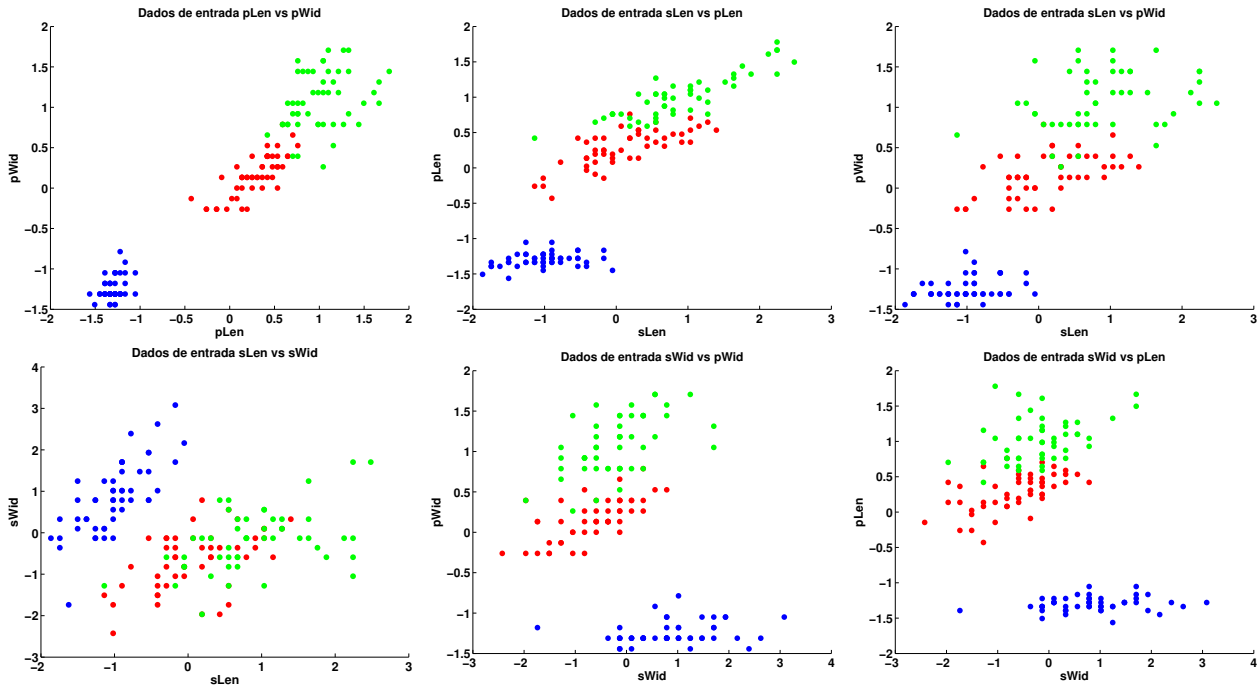

Figura 18: Caractéristicas do DataSet da Íris combinados

D. Implementação do Perceptron

Abaixo é possível visualizar o código implementado em Matlab para executar o treinamento do perceptron. Está função considera que o bias foi previamente adicionado como se fosse uma característica do dado de entrada.

```
1
2  %%Treina uma rede neural MLP com apenas uma camanda
3  %Retorna os pesos e o Erro de cada  poca
4  function  [Ws err] = perceptron(x,d,n,maxEpoca)
5
6      [nAmostras nEnt] = size(x) ;
7
8      Ws = rand( length(d(1,:))  , nEnt);
9
10     err = [];
11
12
13     for epc=1:maxEpoca
14
15         cont = 0;
16         em = [];
17
18         ordem = randperm(nAmostras);
19
20         for j=1:nAmostras
21             i = ordem(j);
22
23             xi =  x(i,:);
24             y = Ws * xi';
25
26             em(j,:) = (d(i,:) - y');
27
28             y = y > 0;
29
30             e = d(i,:) - y';
31
32             if( sum(e) ≠0 )
33                 cont = cont + 1;
34             end
35
36             dW = e' * xi * n;
37             Ws = Ws + dW;
38         end
39
40         em = mean (em.^2);
41
42         cont = (cont/nAmostras);
43         %err = [err cont];
44         err(epc,:) = [cont em ];
45
46     if cont==0
47         break
48     end
49
50     end
51
52 end
53  %%
```

E. Exemplo de utilização do Perceptron

Abaixo é possível visualizar o código implementado em Matlab para executar e avaliar o treinamento do perceptron.

```
clc; clear all; close all;

%% Parmetros
%Nmero de amostras geradas
n = 100;
%Nmero de repetioes
nRepeticoes = 10;
%Opera
op =  'AND' ;

%%

[ x  y ] = geraBD(op, n );

%Adicionando bias
x = [ones(n,1) x];


for i=1:nRepeticoes
    [W err] = perceptron(x,y,0.01,100);
    ce(i) = err(end);
end

sprintf('Mdia e desvio padro do erro quadrtico %0.3f, %0.3f',mean(ce),std(ce))

figure('name','Anlise do erro');
plot(err(:,1),'-*r','linewidth',2);hold;
plot(err(:,2),'-*b','linewidth',2);
legend('Erro global','Erro Mdio Quadrtico');
set(gca,'FontSize',15,'fontWeight','bold');
set(findall(gcf,'type','text'),'FontSize',15,'fontWeight','bold');

figure('name','Dados de treinamento');
mostraResultado(x,y,['bias';' X1 ';' X2 '],50);
set(gca,'FontSize',15,'fontWeight','bold');
set(findall(gcf,'type','text'),'FontSize',15,'fontWeight','bold');

figure('name','Regio de Deciso');
mostraRegiaoDecisao(W,1,15, -0.5:0.05:1.5);
mostraResultado(x,y,['bias';' X1 ';' X2 '],60);
set(gca,'FontSize',15,'fontWeight','bold');
set(findall(gcf,'type','text'),'FontSize',15,'fontWeight','bold');
```

REFERÊNCIAS

[1] CS 4793: Introduction to Artificial Neural Networks. Department of Computer Science, University of Texas at San Antonio.
[2] Braga, Antônio de Pádua; Carvalho, André P. L. Ferreira; Ludermir, Teresa Bernarda, "Redes Neurais Artificiais: Teoria e Aplicações"(2000), Rio de Janeiro: LTC.
[3] HAYKIN, Simon. Redes neurais: princípios e prática. trad. Paulo Martins Engel. -2.ed. - Porto Alegre: Bookman, 2001.
[4] http://users.ics.aalto.fi/ahonkela/dippa/node41.html
[5] http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/2.4.0.html
[6] http://www.gsigma.ufsc.br/ popov/aulas/rna/neuronio_artificial/index.html