Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Bacharelado em Ciência da Computação 2017.2
Disciplina: Redes Neurais Artificiais
Professor: Ajalmar Rocha
Aluno: José Igor de Carvalho

Relatório 2: Adaline

1 Introdução

O modelo *Adaline* (Adptive Linear Element) é um tipo de rede neural artificial de camada única baseado no neurônio de McCulloch-Pitts. Este modelo foi desenvolvido em 1960 por Bernard Widrow e Marcian Hoff e, diferentemente do Perceptron, seu principal objetivo é a regressão linear, isto é, o modelo gera os coeficientes de uma função que aproxima de um resultado esperado y a saída para determinada entrada x.

O objetivo deste documento é analisar a eficiência deste modelo quando aplicado à dois conjuntos de dados artificiais distintos. Um destes conjuntos é baseado em uma função de variável única, enquanto que o outro é baseado em uma função multivariável. Como medida de eficiência usaremos o Erro Quadrático Médio (MSE) e o Desvio Padrão para determinada quantidade de erros coletados por realização.

2 Metodologia

Nesta seção faremos uma pequena explanação de como foram gerados os conjuntos de dados que serão usados para mensurar a eficiência do Adaline. Além disso, iremos explicar como ocorreu a normalização dos dados gerados.

2.1 Geração do conjunto de dados 2D

Para geração dos conjunto de dados 2D, usamos uma função base. Esta função é mostrada abaixo

$$y(x) = 2x + 3. \tag{1}$$

Após gerarmos um conjunto com 1000 entradas e armazenarmos estes dados em um vetor $\mathbf{x}$, aplicamos estes dados em y para formamos um vetor $\mathbf{y}$, com iguais 1000 entradas. No entanto, para tornarmos estes dados gerados um pouco mais realistas, adicionamos um ruido à cada entrada de $\mathbf{y}$. Este ruido é um valor que varia entre -0.5 e 0.5. A Figura 1 mostra como a geração dedados aleatória foi codificada.

Por fim, basta de se concatene os vetores $\mathbf{x}$ e $\mathbf{y}$ para formarmos um novo conjunto de dados com dimensão 1000×2.

```
%% GERANDO UM CONJUNTO DE DADOS ALEATÓRIO

f = @(x) 2*x + 3; % Função base

initial = -5; % Faixa inicial para geração dos dados
final = 5; % Faixa final para geração dos dados
step = 0.01; % Passo entre a faixa inicial e a faixa final

% Inicializando vetores para armazenar os dados
x = zeros();
y = zeros();

% Faixas de ruido
a = -0.5;
b = 0.5;

k = 1;
for i=initial:step:final
    x(k) = i;
    y(k) = f(i)+(b-a).*rand(1) + a;
    k = k + 1;
end
```

Figura 1: Código para geração de dados 2D

2.2 Geração do conjunto de dados 3D

De forma semelhante ao conjunto de dados 2D, também foi usada uma função base para a geração do conjunto de dados 3D. Esta função é mostrada abaixo.

$$y(x_1, x_2) = 2x_1 + x_2 \tag{2}$$

Neste caso, foram criados 3 vetores com 1000 estradas cada, **x1**, **x2** e **y**. Os vetores **x1** e **x2** recebem valores gerados aleatoriamente enquanto que **y** recebe os valores de **x1** e **x2** aplicados em y somados com um ruido. Por fim, concatenamos esses vetores para formarmos um novo conjunto de dados com dimensão 1000×3. A Figura 2 mostra o código para geração deste conjunto de dados.

```
initial = -5; % Faixa inicial para geração dos dados
final = 5; % Faixa final para geração dos dados
step = 0.01; % Passo entre a faixa inicial e a faixa final

% Inicializando os vetores x1 e x2
x1 = zeros();
x2 = zeros();
y = zeros();

k = 1;
for i=initial:step:final
   x1(k) = i;
   k=k+1;
end

x1 = x1(randperm(length(x1)))';
x2 = x1(randperm(length(x1)));

% Faixas de ruido
a = -0.7;
b = 0.7;

for i=1:size(x1, 1)
    % Depois adicionar o ruido
    y(i) = f(x1(i), x2(i))+(b-a).*rand(1) + a; % Adicionando ruido
end
```

Figura 2: Código para geração de dados 3D

2.3 Normalização dos dados

Depois da geração dos conjuntos de dados como mostrado acima, foi necessário normalizá-los. Este normalização se dá de acordo com a Equação 3, que é mostrada abaixo.

$$x_{i,j}^{\text{new}} = \frac{x_{i,j}^{\text{old}} - \min(\mathbf{x}_j)}{\max(\mathbf{x}_j) - \min(\mathbf{x}_j)} \tag{3}$$

Após a normalização de ambos os conjuntos de dados, estes então preparados para treinamento.

3 Resultados

Nesta seção mostraremos os resultados obtidos após o treinamento do modelo em questão, para as bases de dados apresentadas acima.

3.1 Resultados para o conjunto de dados 2D

Para o treinamento do modelo para o conjunto de dados 2D, foram separados de, forma aleatória, 700 amostras formando assim um conjunto de testes. As outras 300 amostras compuseram o conjunto de testes. Os dados e a reta gerada a partir do treinamento são mostrados abaixo.

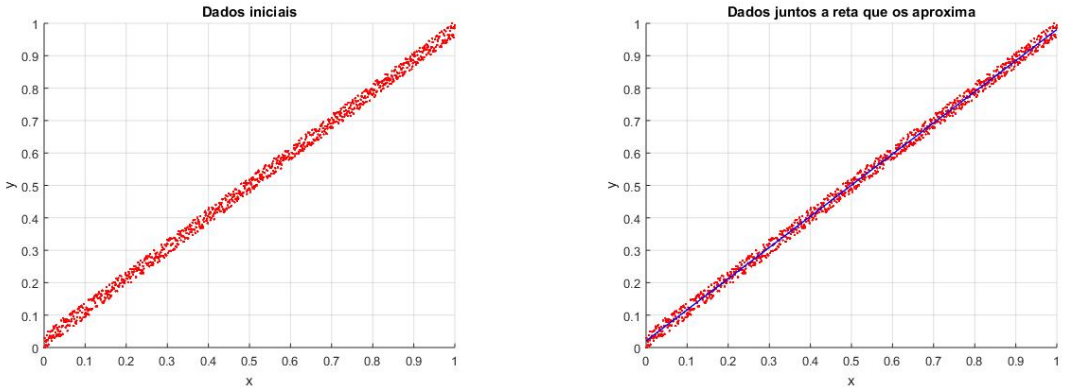

Figura 3: Dados gerados juntamente com a reta que os aproxima

3.2 Resultados para o conjunto de dados 3D

De forma semelhante a mostrada acima, foram usadas 700 amostras para treinar o modelo e o restante, isto é, 300 amostras, foram usadas para testá-lo. Os dados e o hiperplano que aproxima esses dados são mostrados abaixo, na Figura 4.

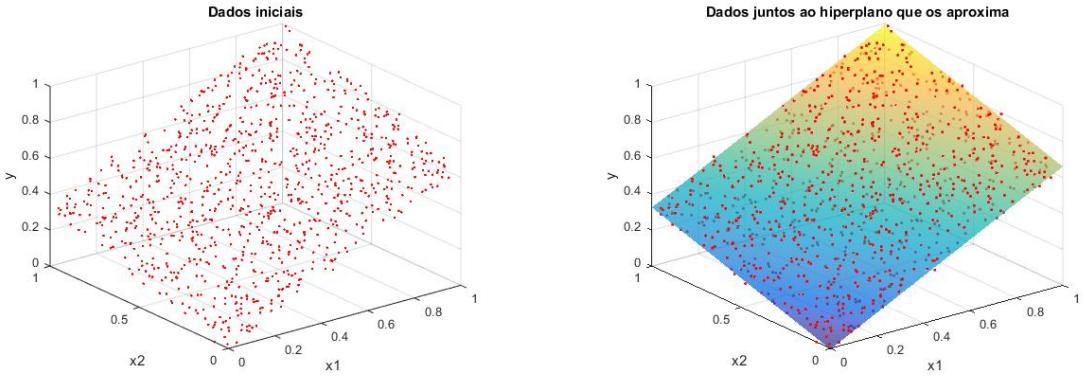

Figura 4: Dados gerados juntamente com a reta que os aproxima

3.3 Eficiência do modelo para os conjuntos de dados

Para medirmos a eficiência, foram usados o MSE (Mean Squared Error) e o RMSE (Root Mean Squared Error) que são mostrados abaixo.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} e^2 = \frac{1}{n}\sum_{i=1}^{n}(y_d - y_i)^2 \tag{4}$$

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_d - y_i)^2} \tag{5}$$

Para avaliarmos uma média desses erros foram feitas um total de 20 execuções seguidas com os dados de treino, isto é, 20 realizações. Os resultados são mostrados na tabela abaixo

Conjunto de dados	**RMSE médio**	**Desvio padrão**
2D	0.0118	0.0059
3D	0.0148	0.0077

4 Conclusão

Como foi possível verificar, o Adaline é um modelo simples e extremamente eficaz para conjuntos de dados lineares. Isto é provado com os resultados obtidos na seção anterior. Obviamente, na maioria das vezes, os problemas reais são bem mais complexos do que os mostrados nesse documento porém, a implementação deste modelo é, sem dúvida, de extrema importância para qualquer pessoa que queira se aprofundar no estudos das Redes Neurais Artificiais.