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 1: Perceptron Simples

1 Introdução

O documento em questão tem por objetivo registrar os resultados obtidos após a execução do algoritmo Perceptron Simples para duas base de dados distintas. Um dos conjuntos de dados usados foi gerado artificialmente enquanto o outro, conjunto de dados da Iris, é bastante conhecido e está disponível na internet. Mais especificamente, nos preocupamos com o registro da taxa de acerto, acurácia e desvio padrão para a classificação que o algoritmo nos fornece para as base de dados citadas acima.

Inicialmente iremos apresentar as bases de dados utilizadas para os experimentos, isto é, ao que se referem os dados contidos no conjunto de dados da Iris e como foi gerada uma base de dados artificial. Feita esta pequena explanação sobre os dados utilizados, trataremos de alguns detalhes de implementação. Por fim, apresentaremos os resultados obtidos.

2 Bases de dados utilizadas

Nesta seção faremos breves esclarecimentos sobre as bases de dados que foram utilizadas para dos testes.

2.1 Base de dados da Iris

A base de dados da Iris é, provavelmente, o conjunto de dados mais conhecido quando falamos em reconhecimento de padrões. Este conjunto de dados se refere a classificação de 3 especies de flores para plantas do gênero Iris: Iris-setosa, Iris-versicolor e Iris-virginica. Abaixo temos uma amostra dos dados deste conjunto.

x_1	x_2	x_3	x_4	Classe
5.1	3.5	1.4	0.2	Iris-setosa
5.9	3.0	4.2	1.5	Iris-versicolor
7.7	2.6	6.9	2.3	Iris-virginica

Neste conjunto, x_1, x_2, x_3 e x_4 referem-se á largura da sépala, comprimento da sépala largura da pétala e comprimento da pétala das flores respectivamente. Já a classe refere-se à espécie da flor.

O conjunto mostrado acima possui apenas 3 amostras, no entanto, a base de dados original possui 150 amostras com 50 amostras para cada espécie.

2.2 Base de dados artificial

Como dito anteriormente, também foram gerados alguns dados aleatórios para testes. É importante ressaltar que o Perceptron Simples é um classificador linear, isto é, ele é capaz de classificar dados em apenas duas classes. Assim, os dados gerados tiveram que formar um conjunto linearmente independente. Abaixo mostraremos o código utilizado para gerar este conjunto de dados.

```
N = 500; %Quantidade de pontos à serem gerados

offset = 5; % Distancia media entre cada ponto a ser gerado

x = [randn(2,N) randn(2,N)+offset]; % atributos
y = [zeros(1,N) ones(1,N)];         % tags

% Criando e normalizando um conjunto da dados artificial linearmente
% separável
dataset = horzcat(x', y');
```

Figura 1: Gerando dados aleatórios linearmente separáveis

Inicialmente definimos uma variável especificando a quantidade de amostras para cada classe. Nesse caso, temos uma variável `N` com valor 500 especificando que queremos gerar 500 amostras de cada classe. Feito isso, criamos uma variável chamada `offset` definindo uma distância média para cada amostra á ser gerada. Feito isto, foram geradas aleatoriamente duas matrizes `X` e `Y` com dimensões 1000×2 e 1000×1 respectivamente. A primeira matriz refere-se aos atributos de cada amostra gerada. A segunda refere-se as classes dos atributos que foram gerados. Por fim, concatenamos as duas matrizes horizontalmente.

3 Metodologia

Nesta seção trataremos de alguns detalhes de implementação do algoritmo. Os códigos em questão foram escritos em Matlab.

3.1 Regra de aprendizagem

Como já sabemos, matematicamente, a regra de aprendizagem é dada pela formula equação abaixo

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta e \mathbf{x}(n) \tag{1}$$

onde o vetor $\mathbf{w}$ é um vetor de pesos sinápticos, η é uma taxa de aprendizagem, e é o erro entre os dados na base de dados e o valor que temos após a aplicação da função de ativação e $\mathbf{x}$ é o vetor de atributos.

Esta regra de aprendizagem é aplicada em uma fração do conjunto de dados chamada conjunto de treinamento. Além disso, esta operação é realizada uma determinada quantidade de vezes que são conhecidas como épocas.

Para implementar o que foi dito acima, criamos uma função chamada `learning_rule` a qual tem como parâmetros um conjunto de treinamento, uma quantidade de épocas e uma taxa de aprendizagem e nos retorna um vetor $\mathbf{w}$ da pesos sinápticos da forma

$$\mathbf{w} = [\theta, w_1, w_1, \ldots, w_n] \tag{2}$$

onde θ é o chamado thrashold ou bias.

3.2 Função de ativação

A função de ativação é responsável por classificar uma determinada amostra. Para o Perceptron Simples, esta função dada da seguinte forma:

$$f(u) = \begin{cases} 1, & \text{se } u \geq \theta \\ 0, & \text{se } u < \theta \end{cases} \tag{3}$$

onde

$$u = \sum w_i x_i. \tag{4}$$

Implementamos uma função chamada `activation` que tem como parâmetro um valor u e nos retorna um outro valor que pode ser igual a 1 para u para valores positivos e 0 caso contrário.

Uma realização consiste no treinamento de um neurônio para um um determinado conjunto de treinamento por determinadas épocas, junto com o cálculo da taxa de acerto para cada teste realizado após o treinamento.

3.3 Normalização de dados

Antes de submeter ambos os conjuntos de dados para treinamento e posteriormente à testes, foi necessário normalizar os dados. Esta normalização foi executada com base na equação abaixo

$$x_{i,j}^{\text{new}} = \frac{x_{i,j}^{\text{old}} - \min(\mathbf{x}_j)}{\max(\mathbf{x}_j) - \min(\mathbf{x}_j)} \tag{5}$$

Uma pequena função chamada `normalize` foi implementada. Esta função recebe como parâmetro uma base de dados e devolve outra base de dados normalizada segundo a equação acima.

3.4 Verificação dos dados

A verificação dos dados é, relativamente, bem simples. Esta verificação é realizada da seguinte forma: se $u - \mathbf{y}(i) = 0$ então houve um acerto na verificação, caso contrário, temos uma falha e então o algoritmo errou a classificação. Note que $u - \mathbf{y}(i) = e$ onde e é o mesmo da equação (1).

Para realizar efetuar esta verificação, foi implementada uma pequena função chamada `verify` que recebe como parâmetro um conjunto de testes e nos retorna a quantidade de acertos e erros de classificação.

Este função é extremamente importante para o cálculo da taxa de acerto para um teste. Este cálculo é feito da seguinte forma: para cada teste realizamos uma verificação no conjunto de testes e então teremos a quantidade de acertos e a quantidade de erros de classificação. Feito isto, podemos calcular a taxa de acertos pela equação

$$\textbf{Taxa de acerto} = \frac{\text{qtd\_acertos}}{\text{qtd\_acertos} + \text{qtd\_erros}}. \tag{6}$$

4 Resultados

Nesta seção iremos avaliar a eficácia do algoritmo implementado para as bases de dados citadas na seção 2.

4.1 Base de dados artificial

Já explicamos como a base de dados artificial foi gerada. Aqui iremos mostrar os resultados para execução do algoritmo implementado nesta base de dados. Mas antes, devemos explicar o conceito de realização.

Cada realização é composta por o treinamento de um neurônio por determinada quantidade de épocas, um teste e o registro da taxa de acerto. Dessa forma, se tivermos 10 realizações iremos armazenar 10 taxas de acerto. Calculando a média das taxas de acerto, obteremos a acurácia do método para determinada base de dados.

Como já foi dito, a base de dados artificial possui 1000 amostras com 500 amostras de cada classe. Definimos as classes como 0 e 1. Selecionamos aleatoriamente 700 amostras para treinamento e 300 amostras para teste. Os dados gerados são mostrados na Figura 2

Além disso, como temos apenas dois atributos para cada amostra, é possível desenharmos uma reta de decisão. Esta reta pode ser obtida pela equação (7). A figura 3 mostra essa reta de decisão para os dados gerados.

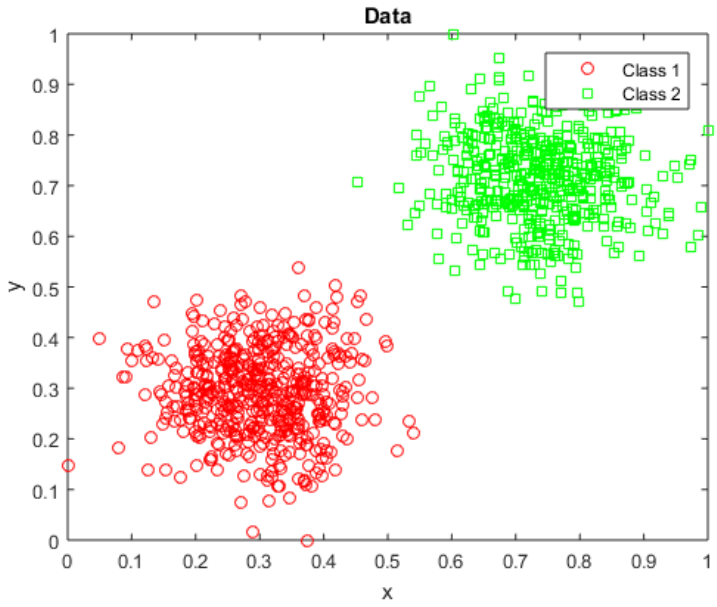

Figura 2: Dados gerados aleatoriamente

$$x_2 = -\left(\frac{w_1}{w_2}\right) x_1 + \left(\frac{\theta}{w_2}\right) \tag{7}$$

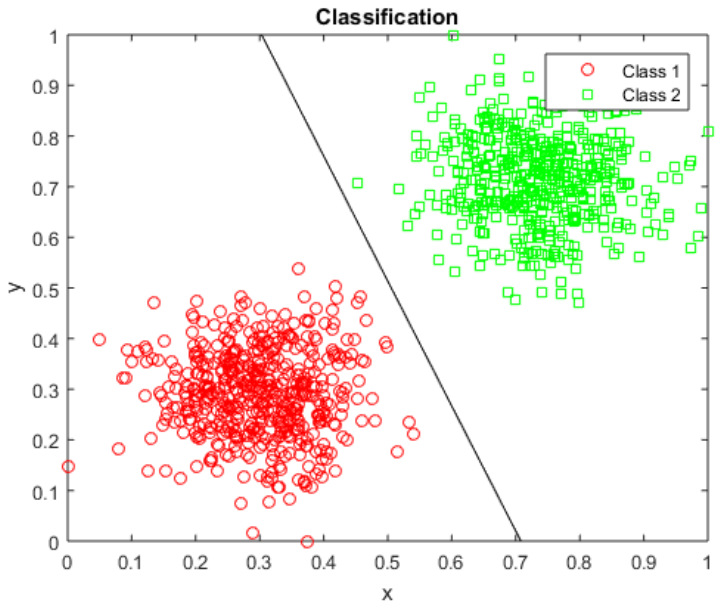

Figura 3: Dados gerados aleatoriamente juntamente com a reta de decisão

Com os dados gerados, podemos gerar um mapa de cores simples, que é mostrado

na Figura 4.

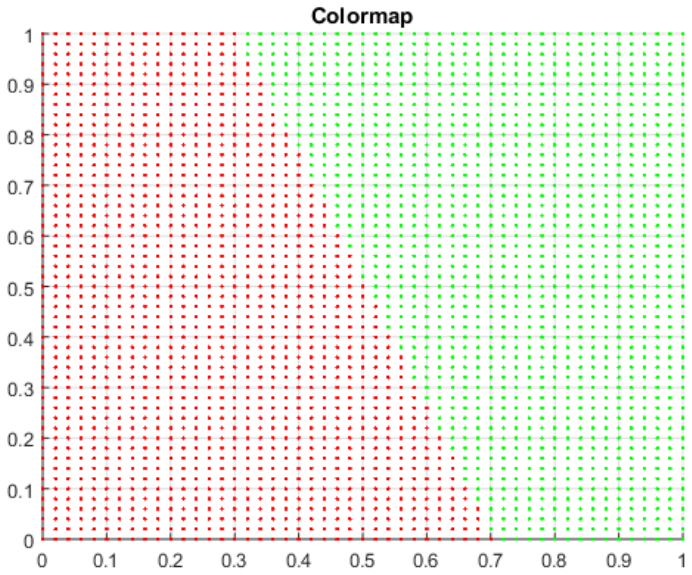

Figura 4: Mapa de cores

Para esta base de dados, a acurácia foi 0.9870 e o desvio padrão foi 0.0200. Também foi calculada uma matriz de confusão para a ultima realização. Esta matriz é mostrada mais abaixo.

	0	1
0	154	2
1	0	144

5 Base de dados da Iris

Como dito anteriormente, a base de dados da Iris possui 150 amostras e 3 classes. Desta forma, o Perceptron Simples não é capaz de classificar corretamente. Assim, foi necessário uma pequena alteração para tornar os dados linearmente independentes. Basicamente, substituímos a classe Iris-setosa por 1 e as classes Iris-versicolor e Iris-virginica foram substituídas por 0. Dessa forma, o conjunto de dados se torna linearmente independente.

Além disso, pelo fato de possuirmos 4 atributos para cada amostra, não é possível gerar gráficos mostrando os dados presentes no conjunto. Assim, podemos apenas mostrar a eficácia do algoritmo por meio do acurácia, desvio padrão e matriz de confusão. Assim, acurácia registrada foi 0.9850 já o desvio padrão foi 0.0253. Já a matriz de confusão é mostrada mais abaixo.

	0	1
0	15	1
1	0	14

6 Considerações finais

Como podemos verificar, o Perceptron Simples é um algoritmo extremamente eficaz quando necessitamos classificar conjuntos de dados linearmente separáveis. Isto é provado pelos resultados obtidos na seção anterior. É claro que, na maioria das vezes, os problemas do mundo real são bem mais complexos do que os apresentados neste documento mas a implementação deste algoritmo foi, sem dúvida, um ponta pé inicial no estudo das Redes Neurais Artificiais.