UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Redes Neurais *Perceptron*

Computação Inspirada pela Natureza

Aluno: Samuel de Souza Lopes

Professor: Dr. Fabricio Aparecido Breve

2018

1. Objetivos

Rosenblatt introduziu o *perceptron* como a forma mais simples de uma rede neural usada para a classificação de padrões linearmente separáveis. Basicamente, o *perceptron* consiste em uma única camada de neurônios com pesos e *bias* ajustáveis (DE CASTRO, 2006).

O objetivo deste trabalho é mostrar duas aplicações relacionadas a redes neurais do tipo *Perceptron*, a saber: classificação de dados do *Iris Dataset* e do *Wine Dataset*. Na seção 2, os detalhes das implementações, bem como os testes e resultados são mostrados. Por fim, na seção 3 são feitas conclusões acerca do trabalho.

2. Aplicações

Nesta seção são detalhadas as implementações de duas aplicações relacionadas a redes neurais do tipo *Perceptron*. A linguagem de programação e o ambiente de desenvolvimento utilizados foram *Python* e *PyCharm*, respectivamente. Ao final de cada implementação, foram feitos testes e os resultados obtidos foram discutidos. As especificações do computador utilizado para fazer os testes de execução dos algoritmos implementados é:

- **Processador:** Intel® Core™ i3-3217U CPU @ 1.80GHz × 4;
- **Memória RAM:** 4 GB;
- **Placa de Vídeo:** Intel® Ivybridge Mobile (integrada);
- **Sistema Operacional:** Ubuntu 16.04 LTS.

2.1. *Iris Dataset*

O objetivo desta aplicação é implementar uma rede neural artificial do tipo *Perceptron* que seja capaz de classificar três classes de espécies de íris, a saber: setosa, virginica e versicolor. Os atributos utilizados para determinar qual classe a íris analisada pertence são:

- comprimento da sépala;
- comprimento da pétala;
- largura da sépala;
- largura da pétala.

Detalhes da implementação do algoritmo

O algoritmo recebe quatro parâmetros, a saber: max_it, alpha, x e d. O parâmetro max_it armazena o número de vezes que o conjunto de treinamento x é inserido na rede. Por sua vez, o alpha é a taxa de aprendizado, representado por um número real entre 0 e 1. Já o x é uma matriz que armazena os padrões de entrada, em que cada linha representa 1 padrão de entrada e as 4 colunas representam os atributos do padrão de entrada. Por fim, o parâmetro d é um vetor que representa as saídas desejadas da rede.

No início da execução da rede, as variáveis w e b são inicializadas. A variável w é uma matriz de pesos com 3 linhas e 4 colunas, onde as linhas representam os pesos das três saídas da rede. Já b é um vetor que armazena os *bias* das saídas da rede. Tanto w quanto b são inicializados com valores reais aleatórios entre 0 e 1.

A cada ciclo, um padrão de treinamento x é utilizado para atualizar a matriz de pesos w e o vetor de *bias* b. Desta maneira, para cada entrada x_i, a saída da rede y_i é calculada somando o produto escalar de w com x_i^t com o vetor b, e aplicando a função *softmax* para normalizar os elementos de y_i. Após isso, é calculado o erro e_i para a entrada x_i, em que e_i é a diferença entre os vetores d_i e x_i. O vetor d_i representa a saída ideal da rede, em que o neurônio de saída a ser ativado em uma classificação correta vale 1 em d_i e os outros valem 0. O erro e_i é utilizado para atualizar w, de forma que w seja incrementada pelo produto de alpha, e_i e x_i. O vetor de *bias* também é atualizado, sendo incrementado pelo produto de alpha e e_i. Ao final, o erro total do ciclo, denominado E, é calculado somando-se os erros quadráticos das saídas y_i.

Caso o valor de E seja menor ou igual a 0 ou a execução atinja o número máximo de ciclos, a execução do algoritmo é encerrada. Caso contrário, um novo ciclo é executado.

Experimentos e Testes

Ao final da implementação do algoritmo, testes foram feitos, sendo divididos em 3 etapas, a saber:

1. Comportamento do algoritmo com diferentes pesos de inicialização e taxas de aprendizado;
2. Gráfico do erro médio quadrático, mostrando a convergência do algoritmo;

3. Matrizes de confusão para cada subconjunto de separadamente: treinamento, validação e teste.

Na etapa 1, os exemplos foram divididos aleatoriamente em subgrupos, de maneira que a intersecção entre ambos seja vazia. As divisões formadas foram, a saber:

- **Subconjunto de treinamento:** 35 instâncias de cada classe;
- **Subconjunto de teste:** 8 instâncias de cada classe;
- **Subconjunto de validação:** 7 instâncias de cada classe.

Executando testes, observou-se que o classificador possui melhores resultados quando a taxa de aprendizado vale 0,1. A Tabela 1 mostra a taxa de erro e acerto média dos conjuntos de subconjuntos de treinamento, teste e validação. Para cada conjunto, a rede foi treinada durante 1000 ciclos. A Tabela 1 mostra as taxas de erro e acertos médias obtidas durante a execução de 10 conjuntos distintos.

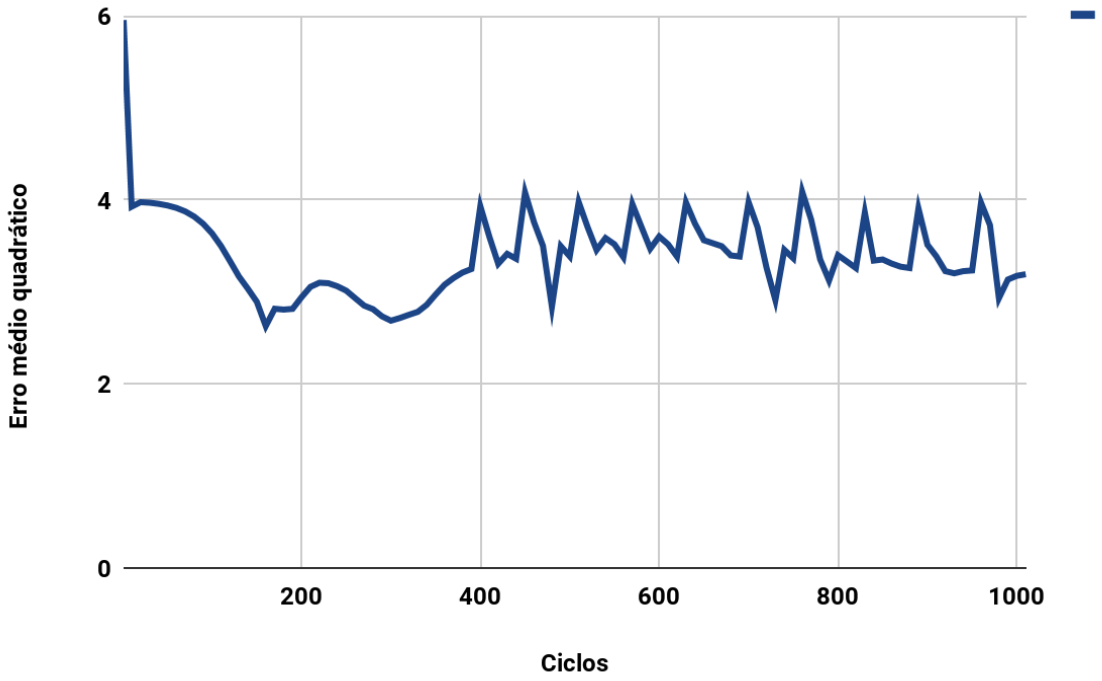
Tabela 1: Taxa de erro e acerto média durante os testes em cada subconjunto.

Subconjunto	**Treinamento**	**Teste**	**Validação**
Taxa de acerto média	96,2%	95%	100%
Taxa de erro média	3,6%	5%	0%

Fonte: Elaborado pelo próprio autor.

Na etapa 2, observou-se a convergência do erro médio quadrático durante a etapa de treinamento da rede, sendo que a taxa de aprendizado utilizada foi 0,1. A Figura 1 mostra os resultados obtidos, sendo que o eixo x representa os ciclos de execução da rede e o eixo y representa os erros médios quadráticos.

Figura 1: Erro médio quadrático do algoritmo.



Fonte: Elaborado pelo próprio autor.

Por fim, na última etapa verificou-se as amostras classificadas erroneamente em cada subconjunto durante uma execução do algoritmo, de forma que os resultados estão expostos na Tabela 2.

Tabela 2: Amostras erroneamente classificadas em cada subconjunto de teste.

	Iris-Setosa	**Iris-Versicolor**	**Iris-Virginica**
Treinamento	0	5	0
Teste	0	1	0
Validação	0	0	0

Fonte: Elaborado pelo próprio autor.

O que se pode concluir com esta aplicação é que o classificador gera resultados aceitáveis pois, mesmo com apenas 35 exemplos de cada classe no subconjunto de treinamento, a taxa de erro do subconjunto de teste foi 3,8%. Os dados expostos na Tabela 1 mostram que, ao contrário das outras classes, a Iris-Versicolor não é facilmente separável já

que ela foi confundida pelo classificador. Por fim, a Figura 1 mostra que o erro médio quadrático converge para valores próximos a 3.

2.2. *Wine Dataset*

O objetivo desta aplicação é implementar uma rede neural artificial do tipo *Perceptron* que seja capaz de classificar três classes de regiões nas quais os vinhos foram produzidos, a saber: região 1, região 2 e região 3. Cada exemplo possui 13 atributos.

Detalhes da implementação do algoritmo

O algoritmo é análogo ao algoritmo da aplicação anterior, sendo que as únicas diferenças são que o número de atributos de cada exemplo é alterado de 4 para 13, de maneira que a matriz w tenha 3 linhas e 13 colunas, e que a matriz de atributos é normalizada por meio da técnica denominada *zscore*.

Experimentos e Testes

Ao final da implementação do algoritmo, testes foram feitos, sendo divididos em 3 etapas, a saber:

1. Comportamento do algoritmo com diferentes pesos de inicialização e taxas de aprendizado;
2. Gráfico do erro médio quadrático, mostrando a convergência do algoritmo;
3. Matrizes de confusão para cada subconjunto de separadamente: treinamento, validação e teste.

Na etapa 1, os exemplos foram divididos em aleatoriamente subgrupos, de maneira que ambos sejam disjuntos. As divisões formadas foram, a saber:

- **Subconjunto de treinamento:** 40 exemplos da classe 1, 49 exemplos da classe 2 e 33 exemplos da classe 3;
- **Subconjunto de teste:** 10 exemplos da classe 1, 11 exemplos da classe 2 e 8 exemplos da classe 3;
- **Subconjunto de validação:** 9 exemplos da classe 1, 11 exemplos da classe 2 e 7 exemplos da classe 3.

Executando testes, observou-se que o classificador possui melhores resultados quando a taxa de aprendizado vale 0,1. Para verificar a taxa de erro e acerto do classificador, foram criados cinco conjuntos distintos de subconjuntos de treinamento, teste e validação. Foram utilizados 100.000 ciclos durante a etapa de aprendizado da rede. Os valores obtidos estão expostos na Tabela 3, em que foram calculadas as taxas de erro e acerto médias dos resultados obtidos.

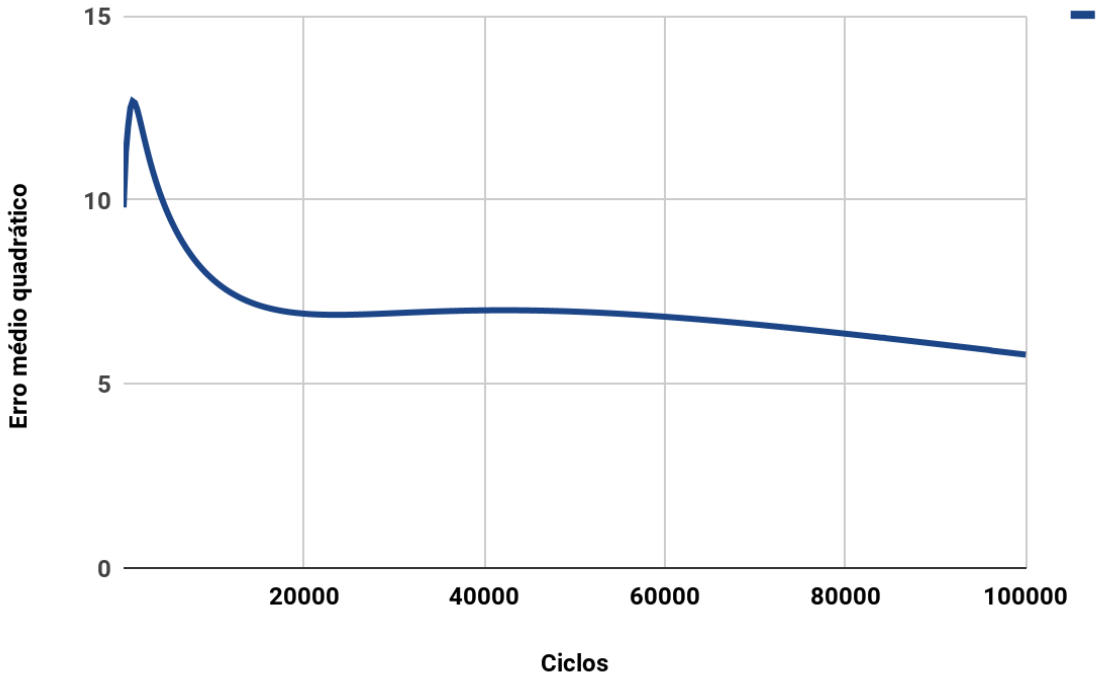
Tabela 3: Taxa de erro e acerto média durante os testes.

Subconjunto	**Treinamento**	**Teste**	**Validação**
Taxa de acerto	95,52%	89,14%	93,82%
Taxa de erro	4,48%	10,86%	6,18%

Fonte: Elaborado pelo próprio autor.

Já na etapa 2, observou-se a convergência do erro médio quadrático durante a etapa de treinamento da rede, sendo que a taxa de aprendizado utilizada foi 0,1. A Figura 2 mostra os resultados obtidos, sendo que o eixo x representa os ciclos de execução da rede e o eixo y representa os erros médios quadráticos.

Figura 2: Erro médio quadrático do algoritmo.



Fonte: Elaborado pelo próprio autor.

Por fim, na última etapa verificou-se as amostras classificadas erroneamente em cada subconjunto durante uma execução do algoritmo, de forma que os resultados estão expostos na Tabela 2.

Tabela 3: Amostras classificadas erroneamente.

	Região 1	**Região 2**	**Região 3**
Treinamento	4	3	0
Teste	4	3	0
Validação	0	1	0

Fonte: Elaborado pelo próprio autor.

O que se pode concluir com esta aplicação é que o classificador gera resultados aceitáveis pois, assim como o *Iris Dataset* que possui subconjunto de treinamento pequeno, o erro do conjunto de teste foi 4,48%. Os dados expostos na Tabela 1 mostram que tanto a região 1 quanto a região 2 não são facilmente separáveis já que ambas foram confundidas pelo

classificador. Por fim, a Figura 1 mostra que o erro médio quadrático converge para valores próximos a 5.

3. Conclusões

A rede neural artificial do tipo *Perceptron* se mostrou eficiente quando utilizada para classificar dados do *Iris Dataset* e do *Wine Dataset* já que os erros verificados foram relativamente baixos. No *Iris Dataset*, a Iris-Versicolor não é facilmente separável visto que o classificador falhou em algumas classificações desta classe. O mesmo vale para as regiões 1 e 2 do *Wine Dataset*. Em geral, redes neurais do tipo *Perceptron* são bastante úteis em problemas de classificação de classes linearmente separáveis.

4. Referências Bibliográficas

DE CASTRO, Leandro Nunes. **Fundamentals of natural computing: basic concepts, algorithms, and applications**. CRC Press, 2006.