PEL208 — Relatório Atividade 6
Implementação do *Perceptron*

Miller Horvath
Mestrando em Engenharia Elétrica (Processamento de Sinais e Imagens)
Centro Universitário FEI, São Bernardo do Campo, SP, Brasil

24 de novembro de 2018

1 Introdução

Este trabalho apresenta o relatório do desenvolvimento da sexta atividade avaliativa referente à disciplina PEL208, intitulada Tópicos Especiais em Aprendizagem, apresentada pelo Prof. Dr. Reinaldo Augusto da Costa Bianchi.

O objetivo desta atividade é implementar o modelo *Perceptron* de redes neurais artificias, conforme abordado em sala de aula. Para isso, a linguagem de programação *Python* foi adotada.

2 Conceitos Fundamentais

2.1 Perceptron

Os estudos em Redes Neurais Artificiais (RNA) surgiram a partir das modelagens computacionais propostas por Warren McCulloch e Walter Pitts em (MCCULLOCH; PITTS, 1943). Um neurônio biológico é composto por dendritos, que recebem informações de outros neurônios, o corpo, onde encontra-se o seu núcleo, e o axônio, que envia impulsos elétricos a partir do neurônio. O contato entre axônio e um dendrito é chamado de sinapse. Inspirado no neurônio biológico, o modelo de um neurônio artificial é composto por um conjunto X de entrada e um conjunto W que pondera as entradas, simulando a sinapse.

As sinapses ponderadas são somadas no núcleo do neurônio artificial, que possuí uma função de ativação para determinar se o neurônio artificial será excitado, propagando o sinal, ou inibido. No caso do Perceptron, a função de ativação é conhecida como função degrau, definida pela Equação 1.

$$f(x) = \begin{cases} 1, & \text{se } w \cdot x + b >= 0 \\ 0, & \text{caso contrário} \end{cases} \tag{1}$$

Sendo x a entrada, ponderada por w, e b é um valor real que atua como viés.

Donald Olding Hebb propôs o primeiro método de aprendizado para modelos neurais computacionais (HEBB et al., 1949), sendo um método de aprendizado não-supervisionado. O algoritmo *Perceptron* foi proposto por Frank Rosenblatt em (ROSENBLATT, 1957). O *Perceptron* é um modelo computacional para reconhecimento de padrões baseado na estrutura básica do neurônio biológico.

$$\begin{aligned} \Delta w_i &= \eta(y - \hat{y})x_i \\ w_i &= w_i + \Delta w_i \end{aligned} \tag{2}$$

Diferentemente do método de Hebb, o *Perceptron* apresenta um método de aprendizado supervisionado, definido pela Equação 2, onde Δw_i defini uma taxa de atualização do ponderamento das sinapses definido por um fator de aprendizado η, pela entrada x_i e pela diferença entre o valor observado y e a saída do *Perceptron* $\hat{y}$.

Devido à característica binária da função de ativação, o modelo *Perceptron* só é capaz de realizar tarefas de classificação para 2 classes. Para tratar 3 ou mais classes, deve-se aplicar duas adaptações à este modelo: (1) treinar um modelo *Perceptron* para cada classe presente no problema; e (2) utilizar uma função de ativação diferente da função degrau. Lidar com o conflito de classes justifica a necessidade de alteração da função de ativação. Suponha que uma determinada observação foi classificada positivamente por mais de um perceptron. Neste caso, todos estes perceptrons terão o mesmo ”peso”, devido a característica binária dos mesmos. Alterando a função de ativação para uma função sigmoide, o conflito entre classes pode ser tratado assumindo que o perceptron que apresentar a maior saída seja escolhido para classificar esta observação.

Outra limitação deste modelo é que o *Perceptron* só é capaz classificar dados linearmente separáveis. Esta característica, juntamente com a afirmação de Minsky e Papert, em 1969, de que estes modelos de RNA nunca seriam capazes de aprender a função lógica *XOR* (ou exclusivo), estagnaram a os estudos em RNA por alguns anos (MINSKY; PAPERT, 2017).

2.2 Exemplos de Modelos *Perceptron*

Para exemplificar o modelo *Perceptron*, serão apresentados o resultado do treinamento deste modelo para aprender as funções lógicas *AND*, *OR* e *XOR*. Dado um par de sinal binário x_1 e x_2 como entrada do *Perceptron*, as equações resultantes do treinamento do modelo para aprender as funções lógicas definidas são:

$$y = \begin{cases} 1, & \text{se } 0.208x_1 + 0.084x_2 - 0.004 >= 0 \\ 0, & \text{caso contrário} \end{cases} \tag{3}$$

$$y = \begin{cases} 1, & \text{se } 0.013x_1 + 0.135x_2 - 0.139 >= 0 \\ 0, & \text{caso contrário} \end{cases} \tag{4}$$

$$y = \begin{cases} 1, & \text{se } 0.143x_1 + 0.141x_2 - 0.140 >= 0 \\ 0, & \text{caso contrário} \end{cases} \quad (5)$$

Sendo que as Equações 3, 4 e 5 são equivalentes as funções lógicas *OR*, *AND* e *XOR*, respectivamente.

Entrada		Saída Esperada			Saída Perceptron		
x_1	x_2	OR	AND	XOR	OR	AND	XOR
0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1
1	0	1	0	1	1	0	1
1	1	1	1	0	1	1	1

Tabela 1 – Compara a saída esperada das funções lógicas *OR*, *AND* e *XOR* com a resultante do treinamento do modelo *Perceptron* para aprender as mesma funções lógicas.

A Tabela 1 mostra que o modelo *Perceptron* foi capaz de aprender as funções lógicas *OR* e *AND*, já que as saídas do modelo foram equivalentes as saídas esperadas. Em contrapartida, após dez mil iterações de aprendizado, o modelo não conseguiu aprender perfeitamente a função lógica *XOR*.

3 Trabalhos Relacionados

Atualmente, devido as suas limitações, o modelo básico de *Perceptron* caiu em desuso. Entretanto, este o *Perceptron* propiciou o desenvolvimento de modelos de Redes Neurais Artificiais (RNA) mais robustos, que são amplamente utilizados hoje em dia. Um dos modelos mais populares é o *Perceptron* Multicamadas (GARDNER; DORLING, 1998).

Uma vertente bastante explorada em RNA é o desenvolvimento de métodos de treinamento (aprendizado) de *Perceptrons* multicamadas. Em (TANG; DENG; HUANG, 2016), foi proposto um novo método de aprendizado para *Perceptron* Multicamadas, baseado num método chamado *Extreme learning machine* (ELM). Esta trabalho visa melhorar o desempenho do *Perceptron* Multicamadas para tratar sinais naturais, como imagens e vídeos. Em (MIRJALILI; MIRJALILI; LEWIS, 2014), é proposto um método de treinamento baseado em biogeografia para mitigar problemas clássicos de RNA, tais como velocidade de convergência, travamento em mínimos locais e sensibilidade do modelo em relação à inicialização das redes.

Modelos preditivos são frequentemente desenvolvidos através de RNAs. Em (ESFE et al., 2015), um modelo *Perceptron* multicamadas é utilizado para correlacionar condutividade térmica de micropartículas de hidróxido de magnésio com o seu volume e temperatura. A capacidade preditiva de três modelos, sendo *Functional Trees*, *Naive Bayes* e *Perceptron* multicamadas, são comparadas em (PHAM et al., 2017) para avaliar suscetibilidade de deslizamento de terra em regiões da Índia. Em (ZHANG et al., 2016), é proposto um modelo

de detecção de patologias cerebrais através da análise de imagens cerebrais, resultantes de ressonâncias magnéticas, utilizando *Perceptron* multicamadas.

4 Metodologia

A implementação da atividade foi desenvolvida na linguagem *Python*. Para apoiar o desenvolvimento da atividade, foram utilizadas as bibliotecas *pandas*, para manipulação dos dados através da estrutura de dados chamada *DataFrame*, e *numpy*, para resolução de operações matriciais.

Foi desenvolvida uma classe chamada *Perceptron*, que recebe uma matriz X e um vetor y como parâmetros de seu método construtor. O parâmetro X armazena o conjunto de observações, definidos por uma série de atributos, utilizados como entrada do modelo perceptron. O parâmetro y possui a classe respectiva a cada uma das observações em X. Devido as limitações do perceptron com função degrau de ativação, conforme discutido na Seção 2.1, esta implementação só pode ser aplicada num problema com 2 classe, pois a classificação das observações deve ser binária (assumindo valor 1 ou 0).

A classe *Perceptron* armazena os seguintes atributos:

- A matriz de pesos W;
- O fator de aprendizado m;
- O número max_it que determina o limite de iterações para treinamento do modelo

Além disso, a classe também oferece os seguintes métodos:

- *train* — Possibilita re-treinar o modelo com um novo conjunto de dados. Recebe os mesmos parâmetros do método construtor;
- *predict* — Recebe uma matriz X, que contém um conjunto de dados a serem classificados, e seus respectivos atributos, e retorna um vetor que contém a classe respectiva a cada um dos dados em X.

5 Experimentos

Para testar e analisar a implementação desenvolvida, foi utilizada uma base de dados de classificação, pertencente ao repositório de aprendizado de máquina da University of California, Irvine (UCI) (DHEERU; TANISKIDOU, 2017), que será descrita na sequência. Foi utilizado o classificador *Perceptron*, descrito na Seção 2.1, para classificar as observações da base de dados adotada. Pelo fato do *Perceptron* com função degrau de ativação ser limitado a lidar apenas com 2 classes, vide Seção 2, este trabalho adota apenas as observações

das classes *Iris-versicolor* e *Iris-virginica*, pois as observações da classe *Iris-setosa* são bastante discriminadas em relação as demais.

Para avaliar o desempenho do *Perceptron*, foi calculada a matriz de confusão para avaliar a taxa de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos para cada uma das classes.

5.1 Bases de Dados

5.1.1 Iris Data Set

Está é uma das bases de dados mais tradicionais em classificação. Esta base possuí informações sobre três diferentes classes de flores, tendo um total de 150 observações, sendo 50 de cada classe, e 5 atributos, descritos a seguir:

- Comprimento da sépala em centímetros;
- Largura da sépala em centímetros;
- Comprimento da pétala em centímetros;
- Largura da pétala em centímetros;
- Classes: *Iris Setosa*, *Iris Versicolour* e *Iris Virginica*.

Mais informações sobre esta base de dados podem ser encontradas em (FISHER, 1936).

6 Resultados

Nesta seção, são apresentados os resultados obtidos através do desenvolvimento experimental descrito na Seção 5.

6.1 Iris Data Set

A Tabela 2 apresenta a matriz de confusão resultante da aplicação do algoritmo *Perceptron*, descrito na Seção 2.1, para classificar os dados da base Iris Data Set, apresentada na Seção 5.1.1, seguindo a metodologia descrita na Seção 4.

		Prevista	
		Iris-versicolor	Iris-virginica
Observada	Iris-versicolor	49	1
	Iris-virginica	0	50

Tabela 2 – Matriz de confusão resultante do classificador *Perceptron* aplicado a base de dados Iris Data Set, descrita na Seção 5.1.1.

De um total de cem observações, igualmente distribuídas entre as classes, 99% delas foram classificadas corretamente pelo modelo *Perceptron*. Apenas uma observação da classe *Iris-versicolor* foi confundida com a classe *Iris-virginica*.

7 Conclusão

Este trabalho visa relatar a implementação do algoritmo *Perceptron*, utilizando *Python* como linguagem de programação, como tarefa do curso PEL208 do programa de pós-graduação em engenharia elétrica do Centro Universitário FEI. O *Perceptron* foi um algoritmo muito importante para o desenvolvimento das redes neurais artificiais como são conhecidas hoje.

O fato dos resultados, apresentados na Seção 6, serem condizentes com os apresentados em aula sugere que a implementação foi adequada. Ademais, o classificador *Perceptron* apresentou ótimos resultados com a base de dados adotada, pois 99% de assertividade foi obtida.

Entretanto, este modelo de classificação apresenta limitações, pois não é capaz de classificar corretamente dados que não são linearmente separáveis e apresenta a limitação de classes, quando utilizada a função degrau de ativação.

Referências

DHEERU, D.; TANISKIDOU, E. K. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>. Citado na página 4.

ESFE, M. H. et al. Applications of feedforward multilayer perceptron artificial neural networks and empirical correlation for prediction of thermal conductivity of mg (oh) 2–eg using experimental data. *International Communications in Heat and Mass Transfer*, Elsevier, v. 67, p. 46–50, 2015. Citado na página 3.

FISHER, R. *Iris Data Set*. 1936. Acessado em: 2018-11-17. Disponível em: <https://archive.ics.uci.edu/ml/datasets/Iris>. Citado na página 5.

GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998. Citado na página 3.

HEBB, D. O. et al. *The organization of behavior*. [S.l.]: New York: Wiley, 1949. Citado na página 2.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 1.

MINSKY, M.; PAPERT, S. A. *Perceptrons: An introduction to computational geometry*. [S.l.]: MIT press, 2017. Citado na página 2.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Let a biogeography-based optimizer train your multi-layer perceptron. *Information Sciences*, Elsevier, v. 269, p. 188–209, 2014. Citado na página 3.

PHAM, B. T. et al. Landslide susceptibility assesssment in the uttarakhand area (india) using gis: a comparison study of prediction capability of naïve bayes, multilayer perceptron neural networks, and functional trees methods. *Theoretical and Applied Climatology*, Springer, v. 128, n. 1-2, p. 255–273, 2017. Citado na página 3.

ROSENBLATT, F. *The perceptron, a perceiving and recognizing automaton Project Para.* [S.l.]: Cornell Aeronautical Laboratory, 1957. Citado na página 2.

TANG, J.; DENG, C.; HUANG, G.-B. Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems*, IEEE, v. 27, n. 4, p. 809–821, 2016. Citado na página 3.

ZHANG, Y. et al. A multilayer perceptron based smart pathological brain detection system by fractional fourier entropy. *Journal of medical systems*, Springer, v. 40, n. 7, p. 173, 2016. Citado na página 3.