PEL208 — Relatório Atividade 7
Implementação do *Perceptron* Multicamadas

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

19 de dezembro de 2018

1 Introdução

Este trabalho apresenta o relatório do desenvolvimento da sétima 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* Multicamadas de redes neurais artificias, conforme abordado em sala de aula. Para isso, a linguagem de programação *Python* foi adotada.

2 Conceitos Fundamentais

2.1 Multi Layer 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).

Com o surgimento do algoritmo *Back-Propagation* (RUMELHART; HINTON; WILLIAMS, 1986), surgiram também as redes *Perceptron* Multicamadas, que retomaram o interesse da comunidade acadêmica em RNAs. O *Back-Propagation* faz a atualização dos pesos da rede neural artificial através do método do gradiente descendente de modo a minimizar a função de erro definida na Equação 3. Onde t_j^p é o valor desejado de saída do padrão p para o processador j da camada de saída e s_j^p é o estado de ativação do processador j da camada de saída para o padrão p.

$$E = \frac{1}{2}\sum_p \sum_j (t_j^p - s_j^p)^2 \tag{3}$$

Na camada de saída, o erro é computado diretamente pela diferença entre o valor observado e a saída do modelo. Nas camadas subsequentes, o erro da camada de saída é

retro-propagado ponderado pelos pesos das conexões entre as camadas. Entretanto, devido ao método de descida de gradiente, a função de ativação dos neurônios artificiais deve ser derivável, impossibilitando a utilização da função degrau adotada pelo *Perceptron* tradicional, que foi substituída pela função sigmoide, definida pela equação 4.

$$f(net) = \frac{1}{(1 + e^{-net})} \tag{4}$$

A RNA *Perceptron* Multicamadas supera as limitações do modelo *Perceptron* tradicional, sendo capaz de detectar padrões não lineares e resolver, por exemplo, o problema do operador lógico XOR (ou exclusivo).

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*, *Perceptron Multicamadas* 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 *MLP*, que recebe uma matriz X, um vetor y, um vetor *hidden*, um valor real n e um valor inteiro max_it 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* multicamadas. O parâmetro y possui a classe respectiva a cada uma das observações em X. O parâmetro *hidden* defini a topologia da rede neural, sendo que a quantidade de elementos no vetor determina o número de camadas escondidas e os valores em si determinam o quantidade de neurônios artificiais em cada camada. A taxa de aprendizado do modelo é definido pela variável n. Por fim, o parâmetro max_it determina o limite de iterações de aprendizado do modelo.

A classe *Perceptron* armazena os seguintes atributos:

- A matriz de pesos w;
- O fator de aprendizado n;
- O número max_it que determina o limite de iterações para treinamento do modelo
- Um dicionário $class\_to\_idx$ que relaciona os valores de classe da variável y para valores numéricos de 0 a $(N-1)$, sendo N a quantidade de classes do problema.
- Um dicionário idx_to_class que relaciona a classe numérica, definida em $class\_to\_idx$ novamente para a classe original.
- O vetor $class\_vector$ é uma matriz identidade $N \times N$ que define o vetor de saída do modelo para cada classe.

Além disso, a classe também oferece os seguintes métodos:

- *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, foram utilizadas três bases de dados de classificação, pertencentes ao repositório de aprendizado de máquina da *University of California, Irvine* (UCI) (DHEERU; TANISKIDOU, 2017), que serão descritas na sequência. Foi utilizado o classificador *Perceptron* multicamadas, descrito na Seção 2.1, para classificar as observações das base de dados adotada.

Primeiramente, os atributos das bases de dados foram normalizados entre 0 e 1. Em seguida, dois novos atributos foram criados a partir da potencialização ao quadrado e ao cubo de cada atributo original.

Em seguida, é realizada a amostragem dos dados, separando os dados de forma aleatório em três amostras: amostra de treinamento, amostra de teste e amostra de validação; sendo que todas as amostras mantém a proporção de classes da base de dados original. A amostra de treinamento, composta por 60% das observações, é utilizado para treinar os modelos. A amostra de teste, composta por 20% das observações, é utilizado para seleção do melhor modelo, neste caso, a topologia da RNA que melhor atende a classificação dos dados. A amostra de validação, composta por 20% das observações, é utilizada para validar a qualidade do modelo selecionado.

Para selecionar o modelo mais adequado para cada base de dados, várias RNAs foram treinadas e avaliadas, variando a topologia e o fator de aprendizado. Sendo C a quantidade de camadas escondidas, c a quantidade de neurônios artificiais em cada camada e n o fator de aprendizado, foram adotados todas as possibilidades dentro do seguinte intervalo de parâmetros: $1 \leq C \leq 3$, $3 \leq c \leq 6$ e $n \in \{0.3, 0.1, 0.03, 0.01, 0.003\}$. Para cada parametrização, foram treinadas cinco redes com inicialização aleatória de pesos, sendo que cada peso inicial w respeita o seguinte intervalo: $0.0 \leq w \leq 1.0$.

Para avaliar o desempenho do *perceptron* multicamadas, 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).

5.1.2 UCI Wine Data Set

Esta base possuí informações sobre análises químicas de vinhos de três cultivares diferentes, todos de uma mesma região da Itália, possuindo um total de 178 observações e 13 atributos. Mais informações sobre esta base de dados podem ser encontradas em (**??**).

5.1.3 UCI Seeds Data Set

Esta base possuí informações sobre três tipos diferentes de trigo, possuindo um total de 210 observações, sendo 70 de cada classe, e 7 atributos. Mais informações sobre esta base de dados podem ser encontradas em (**??**).

6 Resultados

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

6.1 Iris Data Set

As Tabelas 1 e 2 apresentam as matrizes de confusão resultantes da aplicação do modelo *Perceptron* Multicamadas, descrito na Seção 2.1, na base de dados Iris Data Set, apresentada na Seção 5.1.1, seguindo a metodologia descrita na Seção 4. Sendo que apenas os resultados do modelo selecionado (que obteve maior acurácia na amostra de teste) são apresentados.

		Prevista		
		Iris-setosa	Iris-versicolor	Iris-virginica
Observada	Iris-setosa	10	0	0
	Iris-versicolor	0	10	0
	Iris-virginica	0	0	10

Tabela 1 – Matriz de confusão resultante da classificação da amostra de teste através do melhor modelo *Perceptron* Multicamadas selecionado aplicado a base de dados Iris Data Set, descrita na Seção 5.1.1.

		Prevista		
		Iris-setosa	Iris-versicolor	Iris-virginica
Observada	Iris-setosa	10	0	0
	Iris-versicolor	0	10	0
	Iris-virginica	0	2	8

Tabela 2 – Matriz de confusão resultante da classificação da amostra de validação através do melhor modelo *Perceptron* Multicamadas selecionado aplicado a base de dados Iris Data Set, descrita na Seção 5.1.1.

Nesta base de dados, a topologia definida por 1 camada oculta com 3 neurônios, treinada com uma taxa de aprendizado $n = 0.3$, atingiu 100% de acurácia na classificação da base de dados de teste. Apesar de outras parametrizações também terem alcançado 100% de acurácia, esta topologia foi selecionada por ser a mais enxuta dentre as que atingiram o desempenho máximo. Com a amostra de validação, a topologia selecionada

errou na classificação de apenas 2 observações de um total de 30, atingindo uma acurácia de $93,3\%$.

6.2 Wine Data Set

As Tabelas 3 e 4 apresentam as matrizes de confusão resultantes da aplicação do modelo *Perceptron* Multicamadas, descrito na Seção 2.1, na base de dados UCI Wine Data Set, apresentada na Seção 5.1.2, seguindo a metodologia descrita na Seção 4. Sendo que apenas os resultados do modelo selecionado (que obteve maior acurácia na amostra de teste) são apresentados.

		Prevista		
		Classe 1	Classe 2	Classe 3
Observada	Classe 1	12	0	0
	Classe 2	0	14	0
	Classe 3	0	0	10

Tabela 3 – Matriz de confusão resultante da classificação da amostra de teste através do melhor modelo *Perceptron* Multicamadas selecionado aplicado a base de dados UCI Wine Data Set, descrita na Seção 5.1.2.

		Prevista		
		Classe 1	Classe 2	Classe 3
Observada	Classe 1	12	0	0
	Classe 2	0	13	1
	Classe 3	0	1	8

Tabela 4 – Matriz de confusão resultante da classificação da amostra de validação através do melhor modelo *Perceptron* Multicamadas selecionado aplicado a base de dados UCI Wine Data Set, descrita na Seção 5.1.2.

Nesta base de dados, similarmente ao ocorrido na base de dados *Iris*, a topologia definida por 1 camada oculta com 3 neurônios, treinada com uma taxa de aprendizado $n = 0.3$, atingiu 100% de acurácia na classificação da amostra de teste. Outras parametrizações também alcançaram 100% de acurácia, porém a topologia mais enxuta dentre as que atingiram o desempenho máximo foi selecionada. Com a amostra de validação, a topologia selecionada errou na classificação de apenas 2 observações de um total de 35, atingindo uma acurácia de $94,3\%$.

6.3 Seeds Data Set

As Tabelas 5 e 6 apresentam as matrizes de confusão resultantes da aplicação do modelo *Perceptron* Multicamadas, descrito na Seção 2.1, na base de dados UCI Seeds Data Set, apresentada na Seção 5.1.3, seguindo a metodologia descrita na Seção 4. Sendo que

apenas os resultados do modelo selecionado (que obteve maior acurácia na amostra de teste) são apresentados.

		Prevista		
		Classe 1	Classe 2	Classe 3
Observada	Classe 1	14	0	0
	Classe 2	0	14	0
	Classe 3	1	0	13

Tabela 5 – Matriz de confusão resultante da classificação da amostra de teste através do melhor modelo *Perceptron* Multicamadas selecionado aplicado a base de dados UCI Seeds Data Set, descrita na Seção 5.1.3.

		Prevista		
		Classe 1	Classe 2	Classe 3
Observada	Classe 1	12	0	2
	Classe 2	0	14	0
	Classe 3	0	0	14

Tabela 6 – Matriz de confusão resultante da classificação da amostra de validação através do melhor modelo *Perceptron* Multicamadas selecionado aplicado a base de dados UCI Seeds Data Set, descrita na Seção 5.1.3.

Nesta base de dados, a topologia definida por 1 camada oculta com 3 neurônios, treinada com uma taxa de aprendizado $n = 0.3$, atingiu $97,6\%$ de acurácia na classificação da amostra de teste, sendo a topologia mais enxuta a ter atingido a melhor acurácia dentre todas as parametrizações, errando na classificação de 1 observação de um total de 42. Com a amostra de validação, a topologia selecionada errou na classificação de apenas 2 observações de um total de 42, atingindo uma acurácia de $95,2\%$.

7 Conclusão

Este trabalho visa relatar a implementação do algoritmo *Perceptron* Multicamadas, 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* Multicamadas foi um dos algoritmos mais importantes para o impulsionamento do desenvolvimento de pesquisas em RNAs, área que estava estagnada devido as limitações do *Perceptron* tradicional.

O fato dos resultados, apresentados na Seção 6, serem condizentes com os apresentados em aula sugere que a implementação foi adequada. Ademais, algumas conclusões podem ser tiradas dos experimentos.

O modelo *Perceptron* Multicamadas se mostrou sensível a parametrização. Por exemplo, com 2 camadas ocultas, o modelo não obtinha bons resultados com valores de

taxa de aprendizado muito baixos. Isso se dá pelo fato da taxa de aprendizado mitigar o erro nas camadas finais durante o *Backpropagation*, impossibilitando o aprendizado das camadas iniciais. Este modelo também é sensível a inicialização dos pesos. Como os pesos iniciais foram definidos com valores aleatórios próximos de 1, os resultados dos modelos variaram, algumas vezes atingindo uma acurácia muito baixa.

Em relação a topologia das RNAs, a topologia mais enxuta que foi considerada foi a que apresentou os melhores resultados. Esse comportamento pode ser explicado por características das bases de dados adotadas. Possivelmente, os padrões de separação entre as classes não são muito complexos, sendo possível separá-las com uma rede mais simples. Outra característica que pode ter impactado é a quantidade de observações. Pode ser que a avaliação dos modelos não tenha sido capaz de ponderar algumas peculiaridades dos padrões das classes, pois as amostras de teste e validação tinham apenas algumas dezenas de observações.

Em geral, as variações de topologia com até 2 camadas escondidas obtiveram acurácia igual ou próxima à topologia que atingiu os melhores resultados. Em contrapartida, os modelos com 3 camadas apresentaram resultados bem abaixo das demais. Em alguns casos, foi incapaz de diferenciar as classes. classificando todas as observações como pertencentes à uma mesma classe. Esse comportamento também pode ser justificado pela mitigação da propagação do erro no processo de *Backpropagation*.

De um modo geral, o modelo *Perceptron* Multicamadas atingiu acurácia próxima a 100% para a amostra de validação para todas as classes, sendo que todas possuem um balanceamento na quantidade de elementos por classe.

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.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533, 1986. 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.