Redes Neurais Artificiais: Artigo 2

Victor São Paulo Ruela
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
Email: victorspruela@ufmg.br

***Resumo*— Este trabalho tem como objetivo avaliar o desempneho de diferentes modelos de redes neurais artificiais estudados durante a disciplina sobre bases de dados de benchmark presentes na literatura. Serão considerados o Perceptron, Adaline, Redes RBF, ELM e ELM com aprendizado Hebbiano. Para três problemas de regressão e classificação binária escolhidos, um experimento foi desenhado seguindo as recomendações da literatura. Os resultados de cada modelos são comparados por meio de testes estatíscticos para as metrícas AUC (classficação) e coeficiente de correlação linear (regressão). Os resultados mostraram que ...**

I. Introdução

A Rede Neural Artificial (RNA) é uma classe de modelos muito popular em problemas de classificação, reconhecimento de padrões, regressão e predição [1]. Inspirado pelas características do cérebro humano, elas possuem como elementos básicos neurônios artificiais capazes de executar operações matemáticas, representando desta forma modelos de neurônios biológicos. Através de sua organização em diferentes estruturas de rede, tais modelos são capazes de se adaptar e representar funções matemáticas bastante complexas.

II. Revisão da literatura

Nesta seção é feita uma breve descrição dos modelos de redes neurais utilizados neste trabalho.

A. Perceptron

Proposto inicialmente por Rosenblatt [2], este é um modelo geralmente utilizado para a solução de problemas de classificação lineares. No seu trabalho original, o autor descreve formas de adaptação dos parâmetros, ou pesos, da rede com o objetivo de reduzir a discrepância entre as saídas esperadas e estimadas e aprender associações entre os neurônios, o que é a base da indução para diversos algoritmos atuais. Este trabalho é considerado um marco na literatura por diversos autores.

Se considerarmos uma função de ativação contínua e diferenciável, os pesos da rede poderão ser inferidos de forma explícita, através do cálculo da pseudo-inversa, ou pelo algoritmo do gradiente descendente [3]. Exemplos de funções de ativação com esta característica frequentemente empregadas na literatura são a função logística, tangente hiperbólica e linear [1]. Vale a pena ressaltar que a covergência destas abordagens está condicionada aos dados utilizados para treinamento serem linearmente independentes [3].

B. Adaline

O Adaline foi inicialmente desenvolvido por Widrow em 1960 [4], sendo principamente aplicado em problemas de regressão lineares. Assim como o Perceptron, originalmente este modelo considera somente um neurônio MCP em sua formulação, entretando sua função de ativação é a identidade. Seu treinamento é formulado como um problema de otimização com custo quadrático, onde originalmente foi utilizado o algoritmo do gradiente descendente para sua solução.

Para este agoritmo, em cada iteração é dado um passo na direção oposta ao gradiente da função objetivo, resultanto em uma convergência gradual para o mínimo do problema. Este gradiente pode ser calculado de forma análitica para a estrutura de rede do Adaline, o qual é no fim proporcional à diferença entre os valores estimados e reais [4], bastante similar ao Perceptron simples. É fácil notar que o treinamento também pode ser realizado de forma direta através do cálculo da pseudo-inversa dos dados de entrada, já que este é um problema de mínimos quadrados [5].

C. Redes RBF

As redes RBF foram inicialmente introduzidas por [6] e são caracterizadas por um aprendizado que envolve duas etapas: (i) aplicar uma transformação aos padrões para um espaço onde a probabilidade de serem linearmente separáveis é alta (ii) encontrar os pesos usando o estimador mínimos quadrados usado no Perceptron simples. Essa estrutura pode ser representada por um rede de três camadas, onde sua camada escondida é reponsável pela transformação não-linear das estradas para o novo espaço, geralmente para uma dimensão muito alta.

D. ELM

Inicialmente proposto por [7], as máquinas de aprendizado extremo (ELM) são redes neurais com uma única camada escondida, as quais possuem o atrativo de poucos parâmetros a serem ajustados, generalização maior ou similar e redução do tempo de treinamento das redes em relação aos métodos convencionais. Seu treinamento é baseado na teoria de minimização de risco empírico, necessitando de somente uma iteração para este processo, evitando múltiplas iterações e algoritmos de otimização local [8].

ELMs são capazes de definir adaptivamente o número neurônios da rede e aleatoriamente escolher os pesos das entradas e viéses da camada escondida [9]. Isso faz com o que a rede possa ser considerada como um sistema linear, o qual

pode ser calculado de forma analítica através de uma operação de inversão da matriz de pesos da camada de saídas [9]. Essa característica permite uma drástica redução do esforço computacional do treinamento, geralmente de 10 vezes ou mais [10].

III. Metodologia

A. Bases de Dados

Neste trabalho serão consideradas três bases de dados referentes a problemas de classificação binários e regressão multivariada disponíveis no repositório da UCI [11], totalizando seis problemas. Antes do treinamento, os dados de entrada serão normalizados para o intervalo $[-1, 1]$ e filtrados para remoção de valores inválidos.

B. Desenho do experimento

A partir das recomendações para desenho de experimento para comparação de algoritmos proposta em [12], a seguinte metodologia será adotada:

- Para cada base de dados:
  1) Particionar os dados D em k partições para validação cruzada, mantendo a mesma proporção entre os rótulos
     a) Criar o conjunto de treino $T = D - k$
     b) Para cada modelo:
        i) Executar busca exaustiva com validação cruzada sobre T para os coeficientes de regularização λ
        ii) Escolher λ que obtém o melhor ajuste médio
        iii) Avaliar a métrica do modelo sobre k
  2) Estimar o intervalo de confiança do valor médio da métrica sobre k usando *bootstraping*

O número de partições consideradas será de $k = 10$. Para o item 1(b)i, serão considerados um número fixo de valores igualmente espaçados dentro de um intervalo pré-definido. Além disso, serão considerados cinco partições para a validação cruzada, como forma de controlar um pouco o tamanho do experimento. Serão consideradas as métricas AUC e coeficiente de correlação linear R^2 para classificação e regressão, respectivamente.

A implementação deste experimento, bem como dos modelos utilizados, será feita em Python e utilizando principalmente os pacotes *numpy* [13] e *scikit-learn* [14]. O experimento será realizado em um Notebook Intel Core i7 Quad Core com 8Gb de memória RAM, sendo que o uso de paralelização será utilizado sempre que possível visto a enorme quantidade de vezes que os modelos serão treinados.

IV. Resultados

A. Problemas de Classificação

Para os problemas de classificação foram considerados os algoritmos Perceptron, RBF, ELM e ELM com apredizado Hebbiano. Foi estabelecido um número fixo de 20 neurônions na camada escondida e a regularização foi considerada somente para os algoritmos RBF e ELM. 50 valores de coeficiente de regularização foram escolhidos no intervalo $[0, 1]$. O Perceptron foi executado com 100 épocas de treinamento e uma taxa de aprendizado de 0.01, não sido feito nenhum ajuste de hiper-parâmetros. Os gráficos boxplot para cada conjunto de dados considerado é exibido nas Figuras 1, 2 e 3.

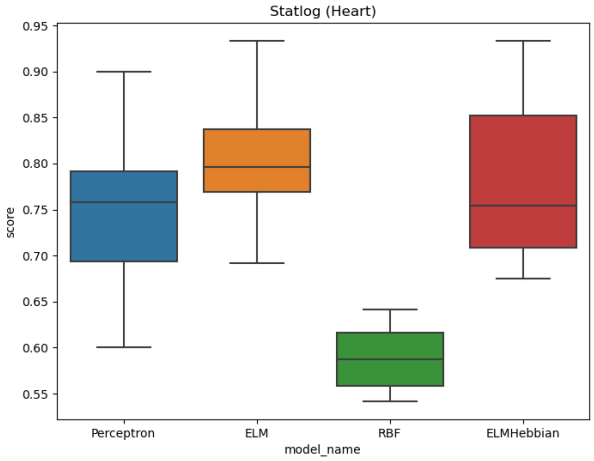

Fig. 1. Boxplots para a base de dados *Statlog (Heart)*

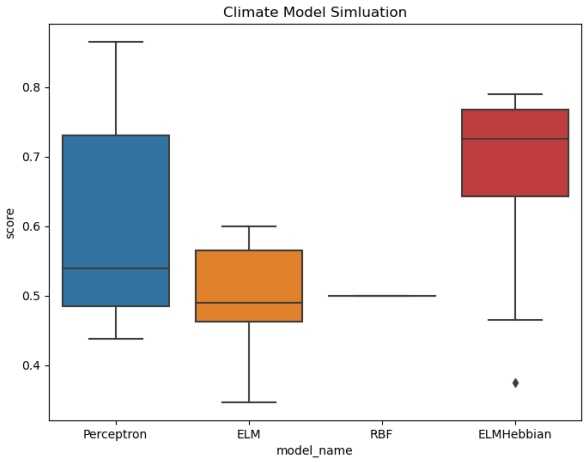

Fig. 2. Boxplots para a base de dados *Climate Model Simulation*

B. Problemas de Regressão

Para os problemas de classificação foram considerados os algoritmos Adaline, RBF e ELM. Não foi possível realizar uma implementação do ELM Hebbiano que funcionasse com problemas de regressão, logo este modelo teve que ser descartado. Foi estabelecido um número fixo de 20 neurônions na camada escondida e a regularização foi considerada somente para os algoritmos RBF e ELM. 50 valores de coeficiente de regularização foram escolhidos no intervalo $[0, 1]$. O Adaline foi executado com 100 épocas de treinamento e uma taxa de

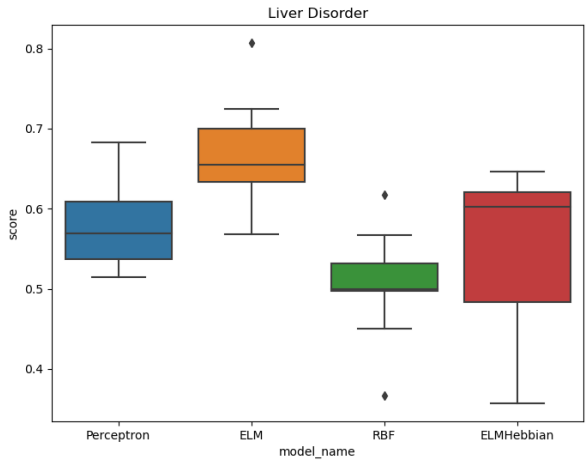

Fig. 3. Boxplots para a base de dados *Liver Disorder (Bupa)*

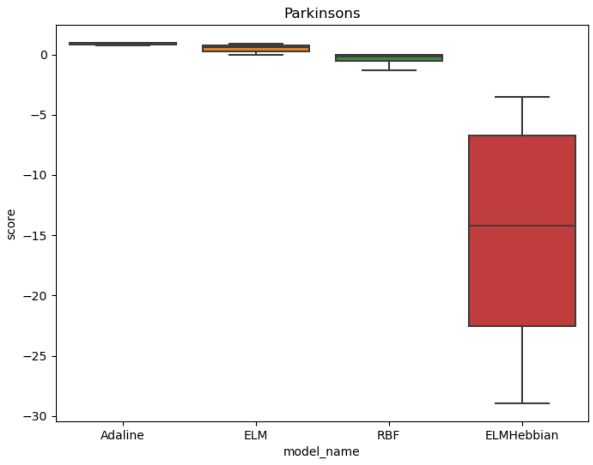

Fig. 5. Boxplots para a base de dados *Parkinsons*

aprendizado de 0.01, não sido feito nenhum ajuste de hiperparâmetros. Os gráficos boxplot para cada conjunto de dados considerado é exibido nas Figuras 4, 5 e 6.

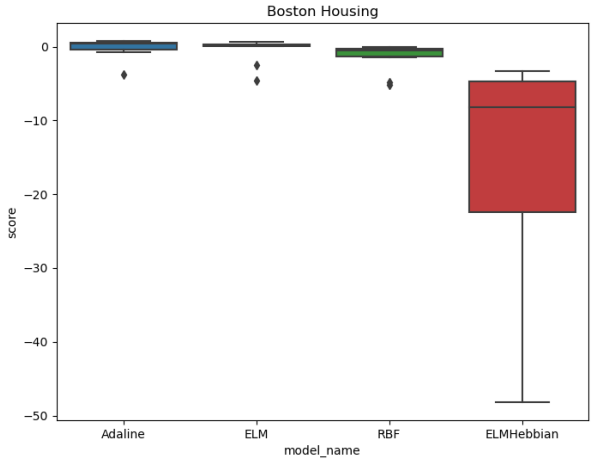

Fig. 4. Boxplots para a base de dados *Boston Housing*

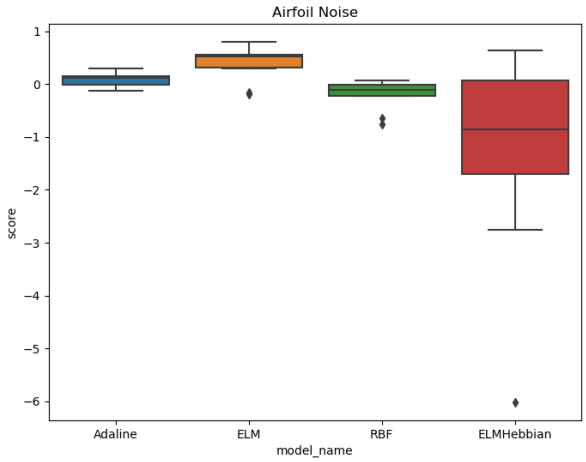

Fig. 6. Boxplots para a base de dados *Airfoil Noise*

V. Conclusões

Neste trabalho foi feita uma revisão bibligráfica de alguns dos principais trabalhos sobre redes neurais aritificiais. Realizando a divisão entre modelos para aprendizado supervisionado e não-supersvisionado, os conceitos básicos dos modelos estudados foram apresentados para contextualização, bem como uma breve análise das principais evoluções e aplicações propostas na literatura. Cada um destes modelos possui uma enorme quantidade de trabalhos publicados, portanto é de se esperar que publicações importantes tenham sido omitidos.

Um aspecto não muito abordado neste trabalho foram as aplicações de RNAs, dado que o foco deste trabalho foi em entender um pouco mais de sua teoria. Em [15] está disponível uma lista das diferentes áreas em que RNAs são comumente aplicadas. Conforme observado durante a realização deste trabalho, grande partes das evoluções visam aprimorar eficiência dos algoritmos treinamento. Isso também é observado por [15], o qual considera uma tendência trabalhos futuros visando aprimorar este aspecto.

Referências

[1] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[2] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[3] John Hertz, Anders Krogh, Richard G Palmer, and Heinz Horner. Introduction to the theory of neural computation. *PhT*, 44(12):70, 1991.

[4] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

[5] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice-Hall, Inc., 2007.

[6] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Syst.*, 2, 1988.

[7] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 985–990. IEEE, 2004.

[8] Shifei Ding, Han Zhao, Yanan Zhang, Xinzheng Xu, and Ru Nie. Extreme learning machine: algorithm, theory and applications. *Artificial Intelligence Review*, 44(1):103–115, 2015.

[9] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[10] Wan-Yu Deng, Qing-Hua Zheng, Lin Chen, and Xue-Bin Xu. Research on extreme learning of neural networks. *Chinese Journal of Computers*, 33(2):279–287, 2010.

[11] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[12] Steven L Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[13] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.