Trabalho Final Redes Neurais Artificiais (Novembro 2020)

Guilherme Barbosa Amorim - 2017089081

I. INTRODUÇÃO

Esse trabalho tem como objetivo aplicar redes neurais artificiais a problemas práticos, visando comparar os métodos estudados em sala em problemas diferentes. Para tanto, 3 bases de dados foram escolhidas: *Wine Quality Data Set, Behavior of the urban traffic of the city of Sao Paulo in Brazil Data Set* e *Breast Cancer Wisconsin (Diagnostic) Data Set*. Essas bases de dados foram encontradas na página *Center for Machine Learning and Intelligent Systems*. Os links dos *data sets* são https://archive.ics.uci.edu/ml/datasets/Wine+Quality, https://archive.ics.uci.edu/ml/datasets/Behavior+of+the+urban+traffic+of+the+city+of+Sao+Paulo+in+Brazil e https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29. Os métodos utilizados serão: Adaline, Perceptron Simples, ELM, RBF e MLPs. A inclusão de cada um deles na análise dependerá das bases de dados escolhidas. Os resultados serão comparados utilizando médias dos resultados em mais de um experimento. Na revisão de literatura, uma explicação teórica dos métodos utilizados e das bases de dados será feita.

I. REVISÃO DE LITERATURA

Redes neurais artificiais (RNAs) são modelos computacionais inspirados pelo sistema nervoso central de um animal (em particular o cérebro) que são capazes de realizar o aprendizado de máquina bem como o reconhecimento de padrões. Esses modelos são capazes de resolver vários problemas. Destaca-se os problemas de Regressão, Previsão e Classificação (RP), como é visto na Figura 1.

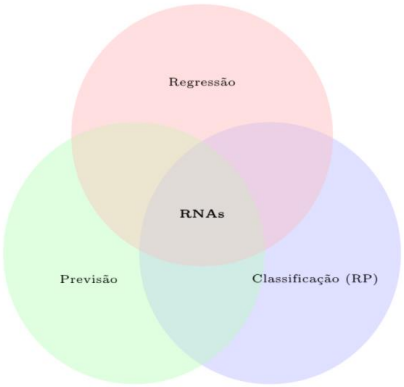

Fig. 1. Principais áreas de aplicação de RNAs.

A. *Adaline*

Adaline é uma rede neural de camada única com vários nós, onde cada nó aceita várias entradas e gera uma saída. O modelo Adaline é caracterizado pela utilização da função identidade $f(u) = u$ como função de ativação, assim sua saída corresponde exatamente à soma ponderada das entradas, ou seja, $\hat{y} = \sum w_i x_i$. Esse modelo utiliza a técnica do gradiente descontente para o ajuste do peso w. Essa técnica é iterativa, de forma que o algoritmo termina a execução se um número pré determinado de épocas foi atingido, ou se o erro da saída é menor que o valor de tolerância. A equação de ajuste dos pesos é $w(t+1) = w(t) + \eta e_i x_i$.

Como a função de ativação utilizada pelo Adaline é linear, esse modelo é utilizado em problemas de previsão e regressão, sendo um exemplo de Redes Neurais Lineares. A Figura 2 mostra a estrutura do algoritmo de treinamento do Adaline.

```
1:  função TRAINADALINE
2:      Entrada: X,y,η,tol,maxepocas
3:      Saída: w,erro por época
4:      N ← Número de linhas de X
5:      Inicializa vetor de pesos w
6:      enquanto (erroepoca > tol) e (nepocas < maxepocas) faça
7:          vetor xseq = permutação de 1 a N
8:          para i ← 1 até N faça
9:              iseq = xseq[i]
10:             erro = (y[iseq] - w^T x[iseq,])
11:             w = w + η erro x[iseq,]
12:             ei2 = ei2 + erro^2
13:         fim para
14:         erroepoca[nepocas] = ei2
15:         nepocas = nepocas + 1
16:     fim enquanto
17: fim função
```

Fig. 2. Algoritmo de treinamento do Adaline.

B. *Perceptron Simples*

O modelo do Perceptron é um exemplo de classificador linear. Classificadores lineares são caracterizados pela aplicação de uma função separadora linear sobre o resultado de uma operação (usualmente) linear aplicada ao vetor de entrada x. A Figura 3 mostra um exemplo de problema linear em uma dimensão e o separador linear resultante do modelo Perceptron simples. As Figura 4 mostra sistemas linearmente separáveis e não separáveis, respectivamente.

O Perceptron foi criado em 1958 por Rosenblatt, sendo a forma mais simples da configuração de uma rede neural artificial, uma vez que é constituída de uma única camada neural e de um único neurônio. A Figura 5 ilustra a rede Perceptron de única camada, onde podemos ter N entradas, mas apenas uma única saída com um valor de 0 e 1 ou de -1 e 1.

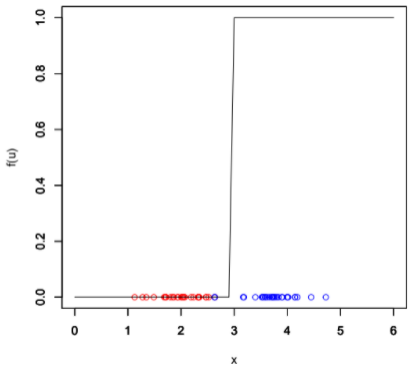

Fig. 3. Classificador linear resultante Perceptron Simples.

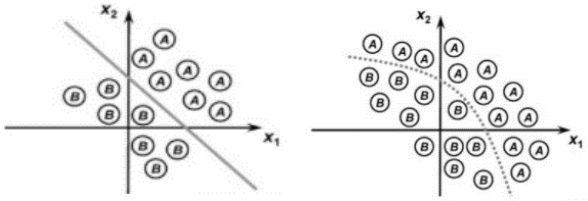

Fig. 4. Sistemas linearmente separáveis e não separáveis, respectivamente.

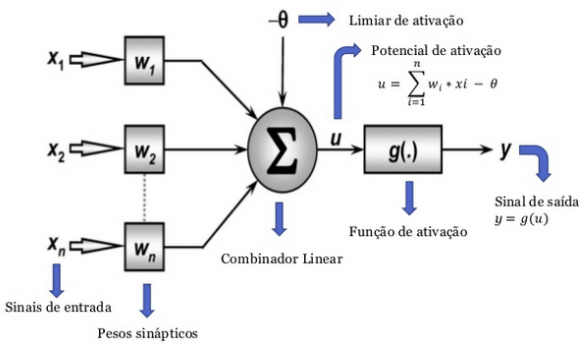

Fig. 5. Rede Perceptron de uma única camada.

Assim como no Adaline, o ajuste dos pesos w é feito pela técnica do gradiente descente (forma iterativa). A equação de ajuste dos pesos é dada por $w(t+1) = w(t) + \eta e_i x_i$. Dessa forma, percebe-se que o Perceptron Simples só se diferencia na função de ativação na camada de saída. O Adaline possui como função de ativação na camada de saída a função linear, enquanto o Perceptron tem uma função limiar, dada por:

$$f(u) = \begin{cases} -1(0), & u < 0 \\ 1, & u \geq 0 \end{cases}.$$

Assim, tem-se que o Percepron simples é utilizado em problemas de classificação.

C. *Extreme Learning Machines*

As ELMs (*Extreme Learning machines*), ou Máquinas de Aprendizado Extremo, fazem uso do teorema de Cover para resolver problemas não lineares. Segundo esse teorema, ao mapear um problema não linear em um espaço de alta dimensão, aumenta-se a chance de o problema ser linearizado. Assim, nesse método, há um mapeamento $\Phi h(\boldsymbol{x_i}, \boldsymbol{Z})$ do sistema para uma camada escondida. O princípio das ELMs é que a matriz **Z** seja selecionada de forma aleatória e que o número de funções $h_i(\boldsymbol{x_i}, \boldsymbol{Z})$ seja suficientemente grande para garantir a separabilidade no espaço da camada intermediária. A Figura 6 mostra a representação esquemática de uma ELM.

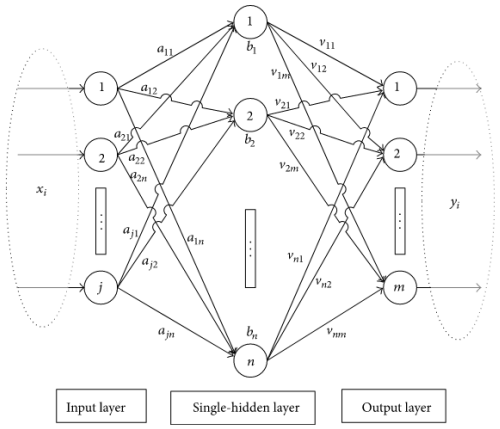

Fig. 6. Representação esquemática de uma rede ELM

Dessa forma, uma vez obtida a matriz de pesos **Z** de forma aleatória, calcula-se a matriz de mapeamento **H** através da operação $\boldsymbol{H} = \Phi h(\boldsymbol{x_i}, \boldsymbol{Z})$, em que $\Phi h(.)$ é usualmente uma função sigmoidal como a tangente hiperbólica. Encontrada a matriz **H**, basta aplicar uma função de ativação $\Phi o(\boldsymbol{HW}) = \boldsymbol{Y}$ para se encontrar a saída. Assim, a equação do erro passa agora a ser $(\boldsymbol{Y} - \boldsymbol{HW})^2$, que é quadrática nos parâmetros da matriz **W**. Por ser quadrática, a técnica do gradiente desentende não se faz mais necessária, uma vez que se tem informações sobre o mínimo global da função de erro. Isso faz com que a rede ELM não seja iterativa, como o Adaline e o Perceptron.

A Figura 7 mostra um problema de classificação que agora pode ser solucionado pela rede ELM.

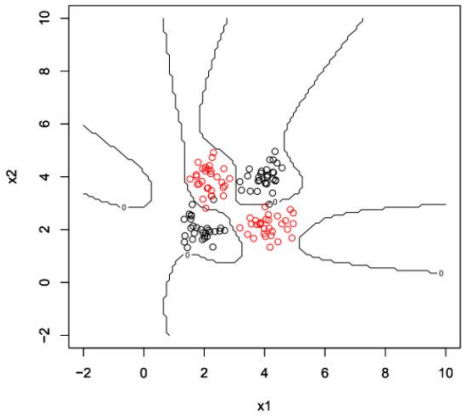

Fig. 7: Problema não-linear solucionado por uma ELM.

D. *Redes RBF*

As redes neurais do tipo RBF (*Radial Basis Functions Neural Networks)* também fazem uso do teorema de Cover e, portanto, também são capazes de solucionar problemas não lineares fazendo um mapeamento para uma camada escondida, assim como foi visto na Figura 6. A principal característica das redes RBFs é a utilização de funções radiais (gaussianas) nos neurônios de sua única camada intermediária, cujas respostas são combinadas linearmente para a obtenção da saída da rede.

Uma rede RBF pode ser descrita formalmente pela equação $f(\boldsymbol{x}, \theta) = \sum_{i=1}^{p} w_i h_i(\boldsymbol{x}, \boldsymbol{z}_i) + w_0$. A Figura 8 mostra de forma

esquemática um problema de duas variáveis solucionado por uma rede RBF.

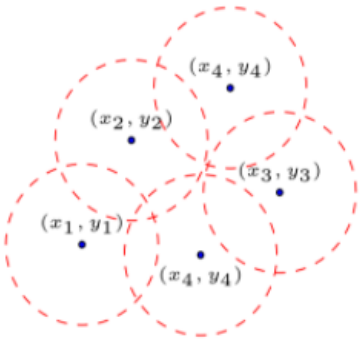

Fig. 8: Representação esquemática das funções radiais de uma rede RBF.

Assim, uma vez obtida a projeção na camada intermediária (a partir de raios e centros dos neurônios do modelo), tem-se a saída do modelo pela equação $\boldsymbol{Y} = \boldsymbol{Hw}$. O vetor de pesos **w** pode então ser obtido pela equação $\boldsymbol{w} = \boldsymbol{H}^{+}\boldsymbol{Y}$.

Para a escolha dos centros dos raios, duas estratégias principais são a alocação uniforme e a alocação com base no *k-médias.* Na alocação uniforme, distribui-se os centros de maneira uniforme sobre o espaço de entrada, ou seja, sobre o domínio da variável x. Na técnica do *k-médias,* dado um número de centros k, escolhe-se aletoriamente k amostras para serem centros. A partir desses k centros, calcula-se a distância até os pontos restantes. Assim, será identificado qual é o centro mais próximo de cada amostra. Com isso, para cada um dos k centros, calcula-se a média das distâncias até os pontos que o tem como centro mais próximo. Os novos centros do modelo passam agora a serem os valores dessas médias.

E. Redes MLP

Nos modelos ELM e RBF, utiliza-se parâmetros **Z** e **W** para a solução de problemas não lineares, onde $\boldsymbol{H} = \Phi h(\boldsymbol{x_i}, \boldsymbol{Z})$ e $\Phi o(\boldsymbol{HW}) = \boldsymbol{Y}$. Nesses modelos, a matriz **H** é obtida a *priori* pela projeção. Para as redes MLP, as matrizes **Z** e **W** são obtidas conjuntamente por meio da minimização do erro global do modelo. Contudo, o mínimo global de uma rede MLP não pode mais ser obtido diretamente com um único passo, como nos modelos ELM e RBF, já que somente informações locais são conhecidas sobre o seu comportamento.

A Figura 9 mostra as superfícies resultantes de modelos lineares e não-lineares, respectivamente.

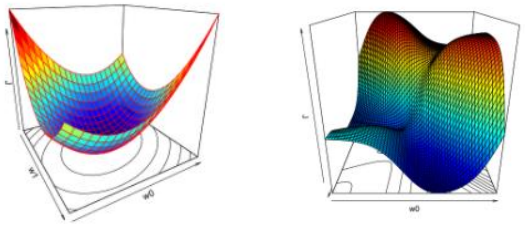

Fig. 9: Superfícies de erro resultantes de modelos lineares e não-lineares.

Redes MLP são aproximadores universais de funções e são tipicamente aplicadas a problemas de classificação, regressão ou previsão. Nesse modelo, faz-se uso do método do *backpropagation* (exemplo de algoritmo que faz uso do gradiente descendente) para a implementação da minimização do erro quadrático.

A Figura 10 mostra a estrutura de uma rede MLP com duas camadas. Tipicamente, os neurônios da camada escondida são sigmoides. O neurônio da camada de saída depende da aplicação. Em problemas de regressão, utiliza-se neurônios lineares. Já problemas de classificação faz uso de neurônios na camada de saída do tipo limiar.

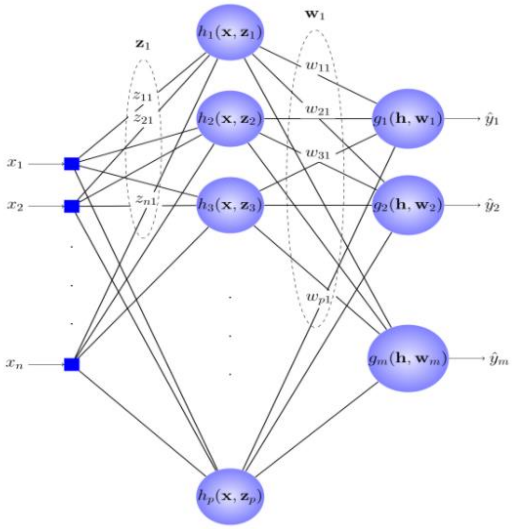

Fig. 10: estrutura de uma rede MLP com duas camadas.

A ideia do método *backpropagation* é utilizar a informação de erro na camada de saída e ir ajustando os pesos de forma iterativa. Assim, com o erro na camada de saída, ajusta-se os pesos **W** do modelo. Com as informações de erro nos neurônios **H**, ajusta-se os parâmetros **Z**.

A quantidade de camadas e de neurônios a serem utilizados dependem da aplicação, cabendo ao algoritmo encontrar valores de **Z** e **W** que minimizem o erro na saída, podendo ficar preso em mínimos locais.

F. Wine Quality Data Set

Esse conjunto de dados tem detalhes descritos em [Cortez et al., 2009]: [©Elsevier] [Pre-press (pdf)] [bib].
Para os experimentos feitos, utilizou-se somente o conjunto de dados relacionados ao vinho tinto (winequality-red.csv). Esse conjunto de dados pode ser tratado como um problema de regressão ou classificação, sendo, na verdade, um problema de multi-classes.

Para o estudo desse conjunto de dados, optou-se pela classificação entre vinhos com nota inferior e superior a seis (vinhos de baixa qualidade e de boa qualidade, respectivamente). Essa escolha tornou o conjunto de dados em um problema binário. Essa escolha fez com que os dados ficassem mais balanceados, como é visto na Figura 11.

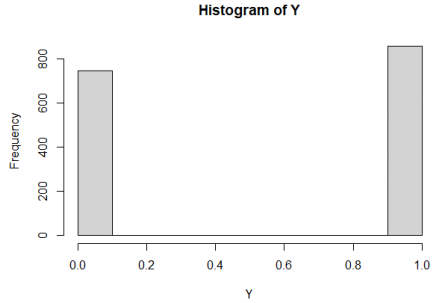

Fig. 11: Balanceamento das classes para o conjunto de dados *Wine Quality.*

Attribute Information:

1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Output variable (based on sensory data):
12 - quality (score between 0 and 10)

Citation Request:
P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

G. Behavior of the urban traffic of the city of São Paulo

O banco de dados foi criado com registros do comportamento do trânsito urbano da cidade de São Paulo no Brasil de 14 de dezembro de 2009 a 18 de dezembro de 2009 (de segunda a sexta-feira). Registrado das 7:00 às 20:00 a cada 30 minutos. O conjunto de dados foi utilizado na pesquisa acadêmica da Universidade Nove de Julho - Programa de Pós-Graduação em Informática e Gestão do Conhecimento.

Essa base de dados conta com 17 variáveis de entrada, têm como *target* a lentidão no trânsito (em porcentagem) e têm somente 135 amostras. A Figura 12 mostra o histograma para os valores de saída dessa classe. Pelo histograma, decidiu-se tornar o problema em um problema binário. Assim, saídas menores que 12 (%) serão classificadas como “sem trânsito”, enquanto saídas maiores que esse valor serão classificadas como “com trânsito”.

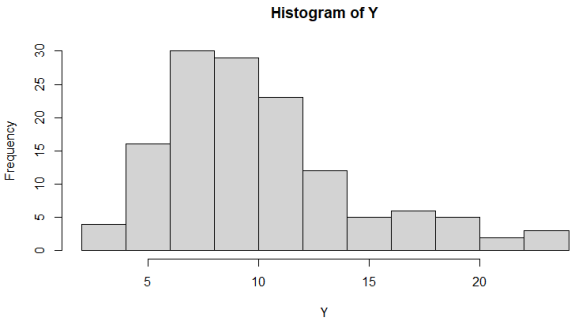

Fig. 12: Histograma original da saída para o conjunto de dados *Behavior of the urban traffic of the city of São Paulo.*

Attribute Information:
1 - Hour
2 - Immobilized bus
3 - Broken Truck
4 - Vehicle excess
5 - Accident victim
6 - Running over
7 - Fire Vehicles
8 - Occurrence involving freight
9 - Incident involving dangerous freight
10 - Lack of electricity
11 - Fire
12 - Point of flooding
13 - Manifestations
14 - Defect in the network of trolleybuses
15 - Tree on the road
16 - Semaphore off
17 - Intermittent Semaphore
18 - Slowness in traffic (%) (Target)

Citation Request:
Ferreira, R. P., Affonso, C., & Sassi, R. J. (2011, November). Combination of Artificial Intelligence Techniques for Prediction the Behavior of Urban Vehicular Traffic in the City of SÃ£o Paulo. In 10th Brazilian Congress on Computational Intelligence (CBIC) - Fortaleza, CearÃ¡ â€“ Brazil. (pp.1-7), 2011.

H. Breast Cancer Wisconsin (Diagnostic) Data Set

O banco de dados conta com 569 instancias, 32 atributos e é um problema de classificação binário. O objetivo dessa base de dados é diagnosticar instancias em M = *malignant* e B = *benign*.

O plano de separação original foi obtido usando *Multisurface Method-Tree* (MSM-T) [K. P. Bennett, "*Decision Tree Construction Via Linear Programming." Proceedings of the 4 Midwest Artificial Intelligence and Cognitive Science Society*, pp. 97-101, 1992], um método de classificação que usa programação linear para construir uma árvore de decisão. Os recursos relevantes foram selecionados usando uma pesquisa exaustiva no espaço de 1-4 recursos e 1-3 planos de separação.

O programa linear real usado para obter o plano de separação no espaço tridimensional é descrito em: [K. P. Bennett e O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

O problema em questão é separável e é balanceado, havendo 357 casos *benign* e 212 casos *malignant*, como é visto na Figura 13.

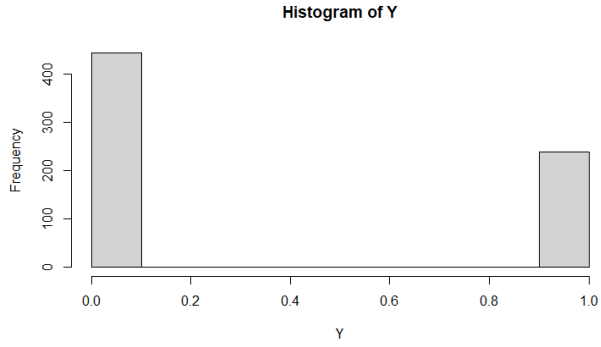

Fig. 12: Histograma da saída para o conjunto de dados *Breast Cancer Wisconsin (Diagnostic).*

Attribute Information:

1 - ID number
2 - Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

Citation Request:
Please refer to the Machine Learning Repository's citation policy.

II. Metodologia

Para a comparação entre os modelos, realizou-se mais de um experimento para cada base de dados. Assim, foi possível calcular a média e o desvio padrão do erro para cada modelo. Os dados foram divididos em 30% para teste e 70% para treinamento.

Para o Perceptron Simples, os parâmetros de treinamento foram $\eta = 0.1, tol = 0.01\ e\ maxepochs = 10000$. O experimento com esses parâmetros tem como saída a média e o desvio padrão do erro percentual para 5 repetições. As saídas binarias foram classificadas como 0 e 1.

Para a rede ELM, as saídas binarias foram classificadas como +1 e -1. Na camada de saída, fez-se uso da função tangente hiperbólica como função de ativação. Ademais, o resultado do experimento é o erro médio percentual e o desvio padrão de 20 repetições do algoritmo. Para cada repetição, utiliza-se números pares de neurônios entre p = 1 e p = 50. O erro de cada repetição é dado pelo menor erro. Assim, o experimento da rede ELM tem 50*20 iterações no total.

Para a rede RBF, foi-se utilizado o algoritmo da biblioteca RSNNS para R. As saídas binarias também foram classificadas como +1 e -1. O resultado do experimento é o erro médio percentual e o desvio padrão de 5 repetições do algoritmo. Para cada repetição, utilizou-se números pares de neurônios (p entre 1 e 16). O erro de cada repetição é dado pelo menor erro. Assim, o experimento envolvendo a rede RBF tem 8*5 iterações no total.

Para a rede MLP, um experimento similar ao experimento da rede ELM foi feito. Contudo, assim como foi dito na revisão literária, o modelo MLP não consegue encontrar o erro global da função de maneira eficiente, como ocorre nas redes RBFs e ELM. Assim, o tempo de execução elevado desse modelo fez com que o número de execuções tivesse que ser reduzido. Dessa forma, o experimento retorna o erro médio percentual e o desvio padrão para 3 repetições. Cada repetição conta com um número par de neurônios entre p = 1 e p = 10. Assim, o experimento da MLP conta com 3 * 10 iterações no total.

III. Resultados

A. *Wine Quality Data Set*

A tabela 1 mostra o erro e o desvio padrão para os experimentos com cada um dos modelos no conjunto de dados *Wine Quality*. O modelo com melhor desempenho, tanto para os dados de teste quanto para os dados de treinamento, foi o modelo MLP. Para essa base de dados, observou-se uma discrepância de 7% entre o erro de treinamento e o erro de teste para a rede ELM.

Modelo	**Erro para os dados de treinamento**	**Erro para os dados de teste**
Perceptron	0.309 +/- 0.005	0.357 +/- 0.087
ELM	0.230 +/- 0.007	0.314 +/- 0.010
RBF	0.241 +/- 0.000	0.241 +/- 0.000
MLP	0.240 +/- 0.002	0.230 +/- 0.004

tabela. 1: Resultado dos experimentos para a base de dados *Wine Quality Data Set*.

Para ter acesso à implementação desse experimento, basta acessar o *link* https://github.com/guiguitz/ANN-Artificial-Neural-Network/blob/main/Final%20Article/wine.R.

B. *Behavior of the urban traffic of the city of São Paulo*

A tabela 2 mostra o resultado do experimento para o segundo banco de dados. Observa-se que todos os modelos obtiveram um ótimo desempenho para os dados de treinamento. Contudo, para os dados de teste, todos tiveram uma alta considerável na porcentagem de erro. Isso se deu em função do baixo número de instâncias (135), o que faz com que o modelo se adeque bem aos dados, não ao modelo em si.

Para essa base de dados, uma característica pode ser mais bem observada: a boa adaptabilidade da rede RBF ao modelo, não aos dados de treinamento. Para os dados de treinamento, a rede RBF contou com o maior erro percentual. Contudo, para a base de testes, percebe-se uma taxa de erro muito baixa se comparada com a taxa dos demais métodos. Assim, todos os outros modelos obtiveram um erro percentual muito maior para a base de teste, o que indica *overfit* durante o treinamento.

Modelo	**Erro para os dados de treinamento**	**Erro para os dados de teste**
Perceptron	0.057 +/- 0.010	0.420 +/- 0.044
ELM	0.032 +/- 0.000	0.362 +/- 0.021
RBF	0.097 +/- 0.018	0.074 +/- 0.015
MLP	0.048 +/- 0.002	0.409 +/- 0.024

tabela. 2: Resultado dos experimentos para a base de dados *Behavior of the urban traffic of the city of São Paulo*.

Para ter acesso à implementação desse experimento, basta acessar o *link* https://github.com/guiguitz/ANN-Artificial-Neural-Network/blob/main/Final%20Article/traffic.R

C. *Breast Cancer Wisconsin (Diagnostic)*

Modelo	**Erro para os dados de treinamento**	**Erro para os dados de teste**
Perceptron	0.028 +/- 0.001	0.027 +/- 0.017
ELM	0.020 +/- 0.002	0.010 +/- 0.004
RBF	0.097 +/- 0.018	0.074 +/- 0.015
MLP	0.045 +/- 0.002	0.028 +/- 0.001

tabela. 3: Resultado dos experimentos para a base de dados *Breast Cancer Wisconsin (Diagnostic)*.

A tabela 3 mostra os resultados dos experimentos para esse banco de dados. Comparando com as demais bases de dados, essa em questão se mostrou com o melhor desempenho, sendo, portanto, a mais separável dentre as três utilizadas.

Para essa base de dados, a rede com melhor desempenho foi a rede ELM, tendo apenas 1% de erro para a base de teste. A rede RBF novamente tem o pior desempenho aos dados de treinamento, mas continua com um bom desempenho para os dados de teste.

Para ter acesso à implementação desse experimento, basta acessar o *link* https://github.com/guiguitz/ANN-Artificial-Neural-Network/blob/main/Final%20Article/breast.R

IV. Conclusão

A partir dos experimentos feitos, observa-se bons desempenhos dos modelos utilizados em problemas de classificação do dia a dia. Ademais, observa-se que, para cada base de dados, um modelo obtém uma resposta melhor. Assim, cabe ao projetista escolher qual modelo melhor descreve o problema de classificação em questão. Técnicas de validação, como o *cross validation*, fazem essa análise inicial de qual modelo melhor se encaixa ao problema em questão.

Ademais, observa-se que, principalmente o modelo RBF, os modelos têm respostas diferentes para dados de teste e validação. Essa discrepância de resultado é fruto de *overfit* durante o treinamento dos modelos. Um ajuste de parâmetros pode ajudar a solucionar esse problema. Assim, para cada modelo, o ideal é que o algoritmo se adeque durante o treinamento, a partir do ajuste de parâmetros de entrada, para que ele consiga ter melhor resposta.

Em relação aos tempos de execução, percebeu-se o quanto os modelos que têm um erro quadrático são mais eficientes. Durante a execução dos modelos, a rede MLP teve um tempo de execução superior à soma dos tempos individuais dos outros modelos.

Portanto, para cada problema, cabe ao projetista escolher o melhor modelo para ser utilizado, levando em consideração o tempo de execução e o possível formato da função geradora.

Referencias

[1] A. P. Braga, “Uma Perpsctiva de redes Neurais Artificias e de Reconhecimento de Padrões”, outubro 2020.
[2] https://archive.ics.uci.edu/, acesso em novembro, 2020.
[3] https://www.embarcados.com.br/, acesso em novembro, 2020.
[4] https://en.wikipedia.org/, acesso em novembro, 2020.