Projeto 01 - Multi-Layer Perceptrons

Matheus Arataque Uema - 10276949

SCC5809 - Redes Neurais e Aprendizado Profundo

1 Introdução

Este projeto explorou o uso de Redes Neurais para Classificação e para Regressão Multivariada. Em classificação, a rede neural é treinada a partir de uma base de dados para, dado uma série de características de um elemento da amostra, prever a qual classe determinado elemento pertence. Para isso, foi utilizado o *Wine dataset*[AF92] a qual analisa 3 tipos de vinhos com base em três constituentes. Por outro lado, a regressão multivariada caracteriza-se por construir um modelo que possa descrever estatisticamente as relações entre diversas variáveis de um determinado conjunto de dados. E para isso, explorou-se a *Geographical Origin of Music dataset*[Zho14] que analisou 1059 registros musicais de 33 países/áreas com base em 68 *features*.

Para construção da rede neural, foi utilizada a linguagem *Python*[1], com o auxílio das seguintes bibliotecas:

- *NumPy*[2]: para manipulações algébricas realizadas nas redes neurais no *Forward*, *Backward propagation* e no cálculo de erro.
- *Scikit-Learn*[3]: Para pré-processamento dos dados
- *Pandas*[4]: Para acesso aos dados
- *Matplotlib*[5]: Para visualização gráfica e análise das métricas da rede neural

Como arquitetura, foi desenvolvida uma classe para o Multi-Layer Perceptron o qual pode possuir múltiplas camadas intermediárias e realiza o *Backward Propagation* com taxa de aprendizado e *momentum*. As camadas intermediárias utilizaram a função (1) $ReLU$ como função de ativação e para camada de saída foi utilizada a função (2) *SoftMax* para classificação e a função (3) Linear para regressão multivariada.

$$R(z) = max(0, z) \tag{1}$$

[1]Python.org Disponível em: https://www.python.org/ na versão 3.8.0. Acesso em: 12 de set. de 2024.

[2]NumPy Disponível em: https://numpy.org/. Acesso em: 12 de set. de 2024.

[3]Scikit-Learn Disponível em: https://scikit-learn.org/stable/. Acesso em: 17 de set. de 2024.

[4]Pandas Disponível em: https://pandas.pydata.org/. Acesso em: 17 de set. de 2024.

[5]Matplotlib Disponível em: https://matplotlib.org/. Acesso em: 17 de set. de 2024.

$$R(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

$$R(z) = z \tag{3}$$

Com base nessa implementação, os funcionamento da rede neural foi explorado para 1 e 2 camadas intermediárias onde as seguintes propriedades foram variadas de forma individual, e ao fim conjuntamente: a quantidade de iterações utilizadas no treino será variada entre 1,000 e 5,000, a taxa de aprendizado da *Backward Propagation* entre 0.01 e 0.1, o coeficiente de *momentum* entre 0.9 e 0.5 e a proporção da base utilizada para treino e validação entre 4:1 e 2:1. Nota-se que para cada iteração foram ajustados os pesos de forma matricial para toda entrada, e não individualmente para cada elemento de entrada, o que justifica o número elevado de iterações. A partir desses diferentes valores, foi analisado a acurácia do modelo na classificação e o erro médio quadrático e sua raiz na regressão multivariada. Por fim, vale ressaltar que as redes neurais foram construídas com a seguinte configuração por padrão:

- *Epochs*: 1000
- *Learning Rate*: 0.01
- *Momentum*: 0.9
- Propoção treino/validação: 4:1

A implementação inteira do código, assim como a base de teste criada, pode ser visualizada no repositório criado no *GitHub*[6], onde também contêm instruções para a execução do código.

2 Desenvolvimento

2.1 Classificação - *Wine*

Na classificação, como cada amostra da base de dados possui 13 características e três dimensões na saída, referente a qual tipo determinado vinho pertence (classe 1, 2 ou 3), decidiu-se portanto utilizar as mesmas dimensões na camada de entrada e saída. Somado a isso, foram utilizados seis neurônios na primeira camada intermediária e, para o caso em que possui duas camadas, treze na primeira e seis na segunda. Como pré-processamento dos dados de entrada, aplicou-se a média e o desvio padrão do conjunto de treinamento de forma a tratar os diferentes tipos de escala aplicada nas características da base (*Alcohol*, *Color intensity*, *Ash*, etc). E na saída foi aplicado a codificação *one-hot* para conversão dos dados categóricos em formato binário. Para medida da acurácia, foi utilizado a média de acertos da predição em relação a saída desejada. Essa métrica foi registrada para todas variações dos modelos com uma e duas camadas

[6]Repositório do Código Disponível em: https://github.com/MatheusUema/neural-networks-study. Acesso em: 17 de set. de 2024.

intermediárias na Tabela 1, e a evolução da acurácia ao longo do treino dos modelos é visível nas Figuras 1, 2, 3, 4, 5 e 6.

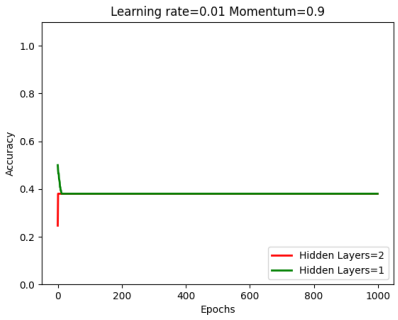

Figura 1: Acurácia por iteração: ajustes padrão

Com base nos resultados, o modelo com duas camadas intermediárias atingiu acurácia inferior em quase todas variações, o que é explicado dado seu tamanho. Como ele possui duas camadas intermediárias, há uma quantidade maior de neurônios cujo os pesos precisaram ser ajustados para que houvesse alguma convergência na acurácia. Apesar disso, mesmo na Figura 6 onde ele atingiu sua maior precisão, foi necessário em torno de 1500 iterações para isso, em comparação com o modelo de uma camada que, por possuir um tamanho menor, obteve o mesmo resultado com menos de 500 iterações. Outro fator que comprovou essa observação foi o fato do modelo de uma camada apresentar resultados melhores no geral, conforme representado na Tabela 1. Também pôde ser observado no MLP com uma camada intermediária, pelas Figuras 2 e 3, que a taxa de aprendizado e o número de iterações foram os coeficientes com mais impacto na precisão de um modelo, apesar disso também poder representar um risco de *overfitting* - dado que elevaria o treino do modelo para atender o *dataset* utilizado -, mesmo que o *bias* tenha sido considerado na implementação. Por fim, interpretou-se pela Figura 5 que alterar a proporção da base entre treino/validação não alterou os resultados porque o desenvolvimento do código manipula algebricamente as camadas e seus respectivos pesos de forma matricial para todo

Tabela 1: Acurácia para cada variação em ambos modelos do MLP

Variação	1 Camada	2 Camadas
Padrão	47.22%	47.22%
Epochs	94.44%	44.44%
Taxa de aprendizado	97.22%	44.44%
Momentum	61.11%	38.89%
Proporção treino/validação	43.33%	43.33%
Todas	96.67%	96.67%

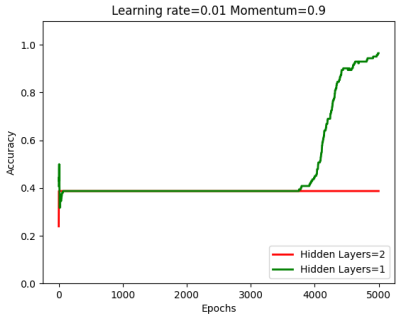

Figura 2: Acurácia por iteração: aumento do número de iterações

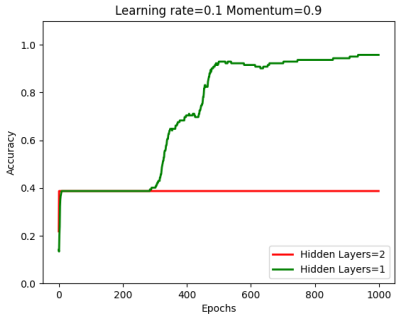

Figura 3: Acurácia por iteração: aumento da taxa de aprendizado

conjunto de entrada a cada iteração.

2.2 Regressão multivariada - *Geographical Origin of Music*

Para regressão multivariada, algumas alterações foram realizadas com relação à classificação na implementação do código. Primeiramente, foram utilizadas 68 e 2 nas camadas de entrada e saída, atendendo à nova base utilizada onde existem 68 *features* relacionadas a originalidade de 1059 registros musicais que possuem como saída a latitude e longitude, já nas camadas intermediárias foi utilizado 32 neurônios em ambos modelos e foi adicionado uma camada de 64 no modelo com duas camadas ocultas. A função de ativação da saída utilizada foi a Linear, já que tratou-se de um problema de caráter contínuo. E para análise dos modelos, variou-se o número de iterações de 500 para 1000. Com relação ao pré-processamento dos dados, a mesma normalização de escalas do problema de classificação foi feita nos dados de entrada, mas não foi necessário aplicar a codificação *one-hot* nos dados de saída. Para esse problema, utilizou-se como perda a raiz do erro médio quadrático dado que este mantém a mesma escala

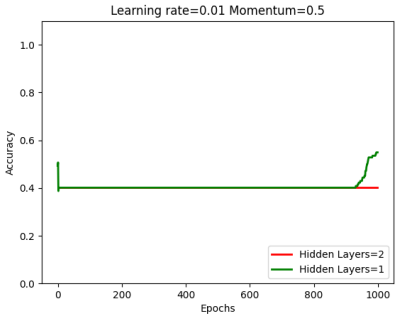

Figura 4: Acurácia por iteração: aumento do coeficiente de *momentum*

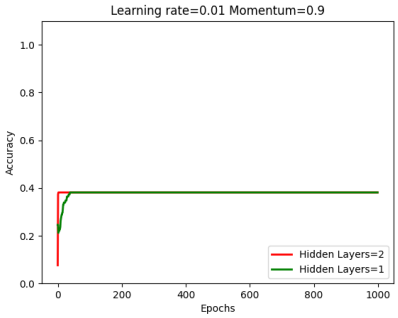

Figura 5: Acurácia por iteração: ajuste da proporção para 2:1

que a saída e permite uma análise melhor da eficiência do modelo. Os valores obtidos para cada modelo com relação a base de validação estão representados pela Tabela 2, e a evolução dos modelos para cada iteração na base de treino está disposta nas Figuras 7, 8, 9, 10 e 11. Como esses resultados apresentaram valores suficientes para análise tanto do modelo com uma camada oculta como com duas, optou-se por não treinar os modelos variando-se todos ajustes.

Como observado no problema de classificação, notou-se pela Tabela 2 que os resultados do modelo com uma camada oculta no geral estão melhores em relação ao segundo modelo também no problema de regressão. No caso em que este modelo prevaleceu, na variação do número de iterações, observou-se pela Figura 8 que ele passou a mostrar uma evolução instável, enquanto o primeiro modelo apresentou maior estabilidade mesmo com um erro maior. Outro ponto similar destes resultados em relação ao anterior foi a convergência no geral mais rápida do MLP de 1 camada em relação ao de duas, o que apoia o argumento de que uma rede neural com tamanho menor precisa de menos iterações para treinar dada as duas bases de dados utilizadas. Por fim, um ponto que chamou atenção foi a instabilidade da rede neural com duas camadas ocultas como representado nas Figuras 9 e 10, o que pode ser explicado pela sensibilidade maior que este

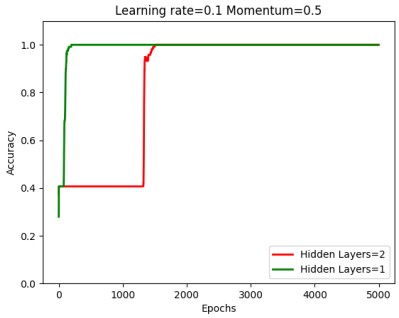

Figura 6: Acurácia por iteração: todas variações aplicadas

Tabela 2: *RMSE* para cada variação em ambos modelos do MLP

Variação	1 Camada	2 Camadas
Padrão	34.39	44.64
Epochs	33.00	34.08
Taxa de aprendizado	34.05	39.52
Momentum	34.87	35.24
Proporção treino/validação	35.65	43.28

modelo tem com relação a etapa de atualização dos pesos, dado seu tamanho e o fato de que essas duas figuras representam variações na equação do *Backward Propagation*.

3 Conclusão

Neste projeto, foram implementadas duas redes neurais, uma para o problema de classificação e outra para o problema de regressão multivariada respeitando suas individualidades. A partir dessas implementações, variou-se coeficientes destes modelos e observou-se pelos resultados que a rede neural com uma camada oculta no geral prevaleceu para os dois problemas, o que pode ser justificado pelo fato das bases de dados utilizadas possuírem tamanho e complexidade pequenos de forma a não ser necessário um grande treinamento. Por conta disso, a adição de uma camada oculta fez com que a rede neural precisasse de mais iterações para atualizar seus pesos devidamente, e nem por isso estaria completamente adequada para estes problemas. Conclui-se então que na aplicação de redes neurais, é importante conhecer o tamanho e a complexidade dos dados assim como qual o problema que será resolvido para que os resultados da implementação possam ser otimizados.

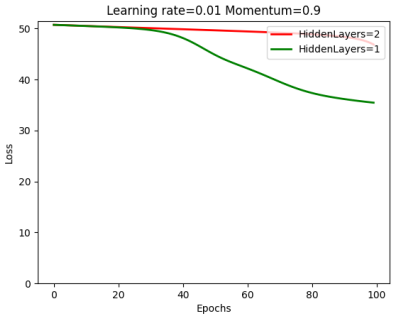

Figura 7: *RMSE* por iteração: ajustes padrão

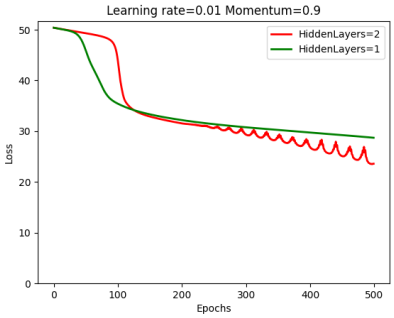

Figura 8: *RMSE* por iteração: aumento do número de iterações

Referências

[AF92] Stefan Aeberhard e M. Forina. *Wine*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5PC7J. 1992.

[Zho14] Fang Zhou. *Geographical Origin of Music*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5VK5D. 2014.

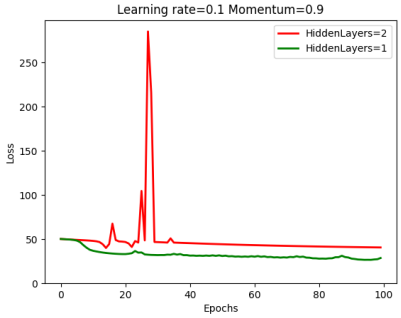

Figura 9: *RMSE* por iteração: aumento da taxa de aprendizado

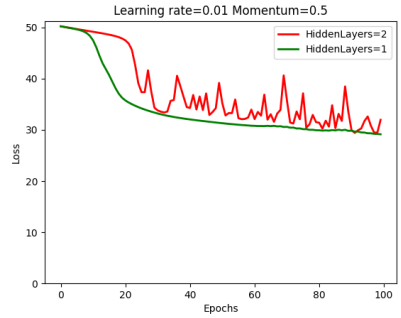

Figura 10: *RMSE* por iteração: aumento do coeficiente de *momentum*

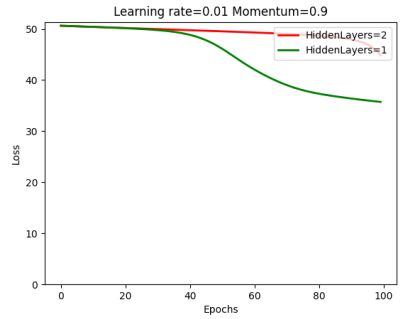

Figura 11: *RMSE* por iteração: ajuste da proporção para 2:1