Análise Comparativa de Otimizadores em Tarefas de Classificação e Regressão Utilizando Redes Neurais Profundas

Gabriele S. Araújo[1], **Omar A. Carmona Cortes**[2]

[1]Departamento de Engenharia da Computação
Universidade Estadual do Maranhão – São Luís – MA – Brazil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Maranhão (IFMA)

gabimitusa@gmail.com

Abstract. *Apesar dos avanços na área de otimização para redes neurais, a seleção do otimizador mais adequado ainda representa um desafio significativo no desenvolvimento de modelos eficientes, especialmente quando consideramos diferentes tipos de tarefas, como classificação e regressão. Assim, este trabalho tem como objetivo realizar uma análise comparativa do desempenho dos otimizadores Adam, RMSprop, SGD com momentum e Adagrad em tarefas de classificação de sentimentos e regressão de dados meteorológicos. A metodologia empregou três arquiteturas MLP, utilizando o dataset IMDB para classificação e dados meteorológicos de Belém, Pará, Brasil para regressão de temperatura, além de testes estatísticos para analisar a significância das arquiteturas e otimizadores. Os resultados apresentaram superioridade significativa do RMSprop na classificação (acurácia 0.8847) e do SGD com momentum na regressão (MAE 0.3973), contribuindo com evidências empíricas para a seleção informada de otimizadores nas devidas tarefas.*

1. Introdução

A otimização tem sido um componente crítico no desenvolvimento de redes neurais profundas (do inglês *deep learning*) ao longo dos anos [Ruder 2016]. O principal objetivo de um algoritmo de otimização em redes neurais é ajustar os parâmetros do modelo de forma a minimizar ou maximizar uma função objetivo, também conhecida como função de custo ou perda. Métodos como a descida de gradiente são amplamente utilizados para encontrar os parâmetros ideais que minimizam essa função, o que, em última instância, melhora o desempenho do modelo em tarefas complexas como classificação e regressão [Du et al. 2019].

Contudo, a estrutura das redes neurais composta por múltiplas camadas de funções não lineares introduz desafios significativos no processo de otimização [Sun 2020]. Ao contrário de problemas convexos ou de programação linear, cuja otimização é bem compreendida, a otimização de redes neurais é um problema teórico complexo e específico, muitas vezes dependente da arquitetura e das condições do modelo [Sun 2020]. Assim, a escolha de um otimizador adequado pode impactar diretamente na convergência e a eficácia do treinamento [Sun 2020, Ruder 2016].

Nesse ensejo, otimizadores tradicionais como o *Stochastic Gradient Descent* (SGD) [Rosenblatt 1958] continuam mostrando vantagens em cenários específicos, principalmente devido à sua simplicidade e eficiência em problemas bem comportados [Ruder 2016]. No entanto, em tarefas mais complexas ou com dados altamente não lineares, o SGD pode ter dificuldades de convergência. Para contornar essas limitações,

otimizadores mais avançados como o *Adaptive Moment Estimation* (Adam) foram desenvolvidos, combinando técnicas de outros otimizadores como o *momentum descent* que ajuda o algoritmo a utilizar uma média dos gradientes anteriores para acelerar a convergência ao mínimo, o *Root Mean Square propagation* (RMSprop) [Tieleman 2012] e Adagrad [Duchi et al. 2011], oferecendo uma melhor adaptação às variações do gradiente, assim tornando-o popular na indústria desde sua introdução por [Kingma 2014].

Sendo assim, a análise comparativa do desempenho de otimizadores em diferentes tipos de tarefas de aprendizado profundo é essencial para determinar o método mais eficiente em termos de tempo de treinamento e precisão [Sun 2020]. Trabalhos anteriores mostraram que, em tarefas de classificação de imagens e texto, o Adam e o RMSprop frequentemente superam outros otimizadores, enquanto otimizadores como o SGD requerem um ajuste fino de hiperparâmetros para alcançar desempenhos comparáveis [Pomerat et al. 2019, Desai 2020, Saleem et al. 2020]. Por outro lado, em tarefas de regressão, a eficiência de cada otimizador pode variar conforme a natureza dos dados e as características da arquitetura da rede neural [Saleem et al. 2020].

Diante desse contexto, o presente estudo teve como objetivo realizar uma análise comparativa do desempenho de otimizadores Adam, RMSprop, SGD com *momentum* e Adagrad em três diferentes arquiteturas de redes neurais profundas, com foco em duas tarefas: a classificação de sentimentos utilizando o *dataset* com resenhas de filmes e a previsão de variáveis meteorológicas com dados coletados de estações meteorológicas da cidade de Belém, Pará, Brasil. Ao conduzir essa análise sobre o comportamento desses otimizadores em diferentes tipos de problemas, espera-se contribuir para a seleção mais informada de técnicas de otimização em aprendizado profundo.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados, seguindo para os materiais e métodos na Seção 3, incluindo detalhes sobre os conjuntos de dados, arquiteturas de redes neurais e configurações experimentais. A Seção 4 apresenta os resultados obtidos, comparando o desempenho dos diferentes otimizadores nas tarefas propostas. Por fim, as considerações finais são expostas na Seção 5, juntamente com sugestões para pesquisas futuras.

2. Trabalhos correlatos

Diversos estudos no estado da arte têm explorado e comparado o desempenho de diferentes otimizadores em redes neurais. A seguir, são descritos alguns dos trabalhos mais relevantes nesse contexto.

No estudo [Rainio et al. 2024], os autores discutem a avaliação e comparação de modelos de aprendizado de máquina (do inglês - *Machine Learning*) em diversas tarefas (*e.g.*, classificação binária, multiclasse, segmentação de imagens e detecção de objetos), utilizando métricas estatísticas e testes apropriados. Na classificação binária, os dados são normalmente categorizados como “positivos” ou “negativos”, onde um rótulo positivo pode indicar, por exemplo, a presença de uma condição, e um rótulo negativo indica a ausência. A avaliação desse tipo de tarefa é feita por meio de uma matriz de confusão, e com base nos valores, são calculadas métricas como acurácia, precisão, sensibilidade (ou *recall*), além de métricas compostas como o *F1-Score*. Já em problemas de regressão, o objetivo é prever valores contínuos, como a altura, preços de ações ou quantidade de chuva, sendo a avaliação baseada em métricas como erro absoluto médio (MAE - do inglês *Mean Absolute Error*) e erro quadrático médio (MSE - do inglês *Mean Squared Error*), onde o MSE é mais sensível a grandes desvios, além de que quanto menor for a medida do

erro, melhor será o desempenho do modelo. Ademais, os autores destacam a importância da validação cruzada, como o método *k-fold*, e o uso de testes estatísticos apropriados, como o teste de Friedman e o teste de Wilcoxon, para comparar o desempenho de modelos em diferentes rodadas de avaliação, ilustrando esse processo por meio de fluxogramas que facilitam a escolha das métricas e testes corretos para cada tarefa.

O estudo de [Eom 2021] explora diferentes otimizadores (SGD, Adam e RMSprop) para análise de sentimentos utilizando o conjunto de dados IMDB, que contém 50.000 análises de filmes classificadas como positivas ou negativas. Os hiperparâmetros incluíram um tamanho de *embedding* de 128 dimensões e um tamanho máximo de sentença de 100 palavras. No experimento, o RMSprop apresentou o melhor desempenho, com uma precisão de 0.8299 nos dados de teste, seguido por Adam com 0.8154, enquanto o SGD obteve apenas 0.5551.

Em [Desai 2020] foi realizada uma análise comparativa dos otimizadores SGD, SGD com *momentum*, RMSprop, Adagrad e Adam, com parâmetros como taxa de aprendizado de 0.01 e batch size de 64 aplicados a um problema de classificação binária usando o *Seattle Weather Dataset*, com a função de perda baseada em entropia cruzada e função de ativação ReLU nas camadas ocultas. Os resultados mostraram que apesar de o Adam ser amplamente utilizado por sua eficiência computacional, o SGD com *momentum* apresentou melhor desempenho tanto em precisão quanto em tempo de treinamento para este conjunto de dados específico.

3. Materiais e Métodos

Nesta seção, são apresentados os materiais e métodos utilizados neste estudo, incluindo a descrição dos *datasets*, o pré-processamento aplicado, as arquiteturas de redes neurais desenvolvidas, os otimizadores avaliados e os métodos estatísticos usados para análise comparativa.

3.1. Datasets

Para a tarefa de **classificação** de sentimentos, foi utilizado o *dataset* IMDB[1] composto por 50.000 resenhas de filmes, igualmente divididas entre sentimentos positivos e negativos. As suas colunas incluem a resenha (*review*) e o sentimento (*sentiment*) que estão como *"positive"* (positivo) e *"negative"* (negativo).

Para a tarefa de **regressão**, foram utilizados os *datasets* de informações meteorológicas do Brasil[2], coletados do Banco de dados da Instituto Nacional de Meteorologia (INMET). As principais *features* que compõem os *datasets* são detalhadas na Tabela 1.

Focando em uma região específica, foi realizada a filtragem apenas dos dados da estação A201, localizada em Belém, no Pará[3] com registros de padrões de precipitação diária de 1º de janeiro de 2014 a 31 de outubro de 2024.

3.2. Pré-processamento dos Dados

Cada tarefa consistiu em etapas específicas de pré-processamento. Assim, seguindo fases importantes para o pré-processamento de textos [Araújo et al. 2023], a **classificação** de textos incluiu: (i) conversão das palavras para minúsculas; (ii) remoção de tags HTML; (iii) remoção de caracteres especiais e números; e (iv) tokenização. Em seguida, foram aplicadas a padronização e o truncamento das sequências de *tokens*, garantindo que todas

[1] `https://bit.ly/3BXLvZm`
[2] `https://bit.ly/48qC54D`
[3] `https://mapas.inmet.gov.br/`

Tabela 1. *Features* do *dataset* meteorológico

Descrição	Função	Coluna
Estação	-	ESTACAO
Data (YYYY-MM-DD)	-	DATA (YYYY-MM-DD)
Precipitação total, horário (mm)	max	rain_max
Radiação global (KJ/m²)	max	rad_max
Temperatura do ar - bulbo seco, horária (°C)	mean	temp_avg
Temperatura máxima na hora ant. (AUT) (°C)	max	temp_max
Temperatura mínima na hora ant. (AUT) (°C)	mean	temp_min
Umidade relativa do ar, horária (%)	mean	hum_avg
Umidade rel. máx. na hora ant. (AUT) (%)	max	hum_max
Umidade rel. mín. na hora ant. (AUT) (%)	min	hum_min
Vento, rajada máxima (m/s)	max	wind_max
Vento, velocidade horária (m/s)	mean	wind_avg

as entradas do modelo tivessem o mesmo comprimento, definido com base no comprimento médio das avaliações do *dataset*. Para essas tarefas, foram utilizadas as bibliotecas NLTK e *Regular Expression* do Python.

Por outro lado, o pré-processamento dos dados meteorológicos para a **regressão** envolveu: (i) mesclagem dos *datasets* coletados por ano; (ii) filtragem da estação alvo (A201); (iii) tratamento de valores ausentes por meio de interpolação linear, quando necessário; e (iv) normalização dos dados utilizando o *StandardScaler*[4], garantindo que todas as *features* tivessem a mesma contribuição ao modelo. Após esse pré-processamento, o conjunto de dados finalizou com 3.870 registros e 10 colunas.

3.3. Arquitetura e Treinamento

As Redes Neurais Artificiais (ANNs - do inglês *Artifcial Neural Networks*) são amplamente utilizadas em tarefas de aprendizado supervisionado, tanto para problemas de classificação quanto de regressão, devido à sua capacidade de capturar padrões complexos em grandes volumes de dados. Dentre as diversas arquiteturas de redes neurais, o MLP é uma das mais utilizadas por sua simplicidade e eficiência em uma variedade de tarefas [Shrivastava et al. 2023].

O MLP consiste em múltiplas camadas de neurônios, onde cada neurônio de uma camada está conectado a todos os neurônios da camada seguinte. Cada uma dessas conexões possui um peso que é ajustado durante o processo de treinamento para minimizar a função de custo do modelo [Shrivastava et al. 2023]. A capacidade do MLP de aprender representações não lineares é determinada pela função de ativação utilizada nas camadas ocultas, sendo a ReLU uma escolha popular por facilitar a convergência durante o treinamento [Shrivastava et al. 2023].

A escolha das funções de ativação e a estrutura das camadas impactam diretamente a capacidade do MLP de generalizar o aprendizado e ajustar-se aos dados [Nwankpa et al. 2018]. Nas camadas ocultas, a função ReLU foi selecionada devido ao seu desempenho em diversos domínios de aprendizado profundo, enquanto na camada de saída a função utilizada varia conforme o tipo de tarefa: a função *Sigmoid* é adequada para classificação binária, enquanto a função identidade (Linear) é usada para regressão [Nwankpa et al. 2018].

[4]`https://bit.ly/3YCkljD`

Com o foco na comparação dos otimizadores, alguns mais populares relatados anteriormente foram selecionados para ajustar os pesos das redes neurais e minimizar a função de custo, dos quais incluem:

- SGD com *momentum*: Utiliza uma taxa de aprendizado constante para atualização dos parâmetros. O termo *momentum* acumula gradientes de passos anteriores, acelerando a convergência em direções relevantes e amortecendo oscilações [Liu et al. 2020, Pomerat et al. 2019];
- Adagrad: Adapta as taxas de aprendizado individualmente para cada parâmetro do modelo. As adaptações são baseadas no histórico de atualizações, permitindo taxas maiores para parâmetros menos frequentemente atualizados [Duchi et al. 2011];
- RMSprop: Aplica uma taxa de aprendizado adaptativa que decai exponencialmente para cada parâmetro. Utiliza médias móveis dos gradientes ao quadrado para ajustar as taxas de atualização, acelerando a convergência [Tieleman 2012];
- Adam: Integra as vantagens do Adagrad e RMSprop, ajustando as taxas de aprendizado com base em momentos acumulados [Kingma 2014].

Dessa forma, as arquiteturas desenvolvidas neste estudo foram ajustadas de maneira a otimizar a *performance* tanto para a tarefa de previsão da temperatura média quanto para a classificação de dados textuais. Em ambas tarefas foram implementadas três variações de MLP com arquiteturas similares, diferindo apenas na camada de saída. A estrutura geral das arquiteturas está apresentada na Tabela 2.

Tabela 2. Arquiteturas MLP

Modelo	Camadas	Neurônios por Camada	Função de Ativação
MLP1	2	64, 1	ReLU, Saída*
MLP2	3	64, 32, 1	ReLU, ReLU, Saída*
MLP3	4	64, 32, 16, 1	ReLU, ReLU, ReLU, Saída*

*Sigmoid para classificação e Linear para regressão

Para a tarefa de classificação, os modelos incluem uma camada de *embedding* inicial, configurada com dimensão de saída 100 e comprimento máximo de entrada de 250 *tokens*, seguida pelas camadas densas. A função de ativação *sigmoid* na camada de saída realiza a classificação binária de sentimentos, utilizando *binary cross-entropy* como função de perda. Para regressão, a função de ativação linear na camada de saída permite previsões contínuas das temperaturas. Em ambos os casos, as camadas intermediárias utilizam ReLU como função de ativação.

O treinamento foi realizado com diferentes otimizadores (Adam, RMSprop, SGD com *momentum*, e Adagrad). Para a classificação foram utilizadas 10 épocas e *batch size* de 32, enquanto para regressão foram 100 épocas e *batch size* de 64. Em ambos os casos, o SGD utilizou taxa de aprendizado de 0.01 com *momentum* 0.9, enquanto os demais otimizadores utilizaram taxa de 0.001.

3.4. Avaliação

Para ambas as tarefas foi empregada a validação cruzada com 5 *folds*. Na classificação, utilizou-se a validação cruzada estratificada (*Stratified K-Fold*), que mantém a proporção de classes em cada *fold*. Para a regressão, devido à natureza temporal dos dados meteorológicos (2014-2024), foi aplicada a técnica de *Time Series Split*, dividindo os dados

em 5 períodos consecutivos, onde cada divisão mantém um ano (365 dias) para teste e incrementa progressivamente o conjunto de treino.

As métricas de avaliação incluíram acurácia, precisão, *recall* e *F1-Score* para classificação, adequadas para medir o desempenho em problemas de classificação binária. Para regressão foi utilizado o MAE, MSE e raiz do erro quadrático médio (RMSE - do inglês *root mean-square error*) que quantificam a diferença entre os valores reais e previstos em problemas de previsão contínua.

Seguindo as análises de [Demšar 2006] e [Rainio et al. 2024], o teste de Friedman foi escolhido por ser adequado para a comparação de múltiplos tratamentos (otimizadores) em cenários com múltiplos blocos (como os *folds* da validação cruzada), tornando-o uma ferramenta apropriada para a avaliação de métodos de ML. Este teste não paramétrico é utilizado para identificar diferenças significativas entre os grupos de tratamento, neste caso, os diferentes otimizadores aplicados aos modelos [Friedman 1937]. Quando o *valor-p* do teste de Friedman é menor que 0.05 indica que há diferenças estatisticamente significativas entre os grupos. Nesses casos, o teste *post-hoc* de Nemenyi é aplicado para identificar quais pares de otimizadores se diferenciam significativamente entre si, proporcionando uma análise detalhada das comparações pareadas [Nemenyi 1963].

Os experimentos foram realizados em um computador com processador Intel Core i5-1135G7 @ 2.40GHz, 8GB de memória RAM e sistema operacional Windows de 64 bits. A implementação foi desenvolvida em Python, utilizando as bibliotecas TensorFlow/Keras para as redes neurais, Scikit-learn para pré-processamento e validação cruzada, e SciPy para análises estatísticas. Mais detalhes sobre os experimentos e resultados estão disponíveis em um repositório no GitHub[5].

Na próxima seção, serão apresentados os resultados obtidos a partir desta metodologia, incluindo o desempenho dos diferentes otimizadores nas tarefas de classificação e regressão, bem como as análises estatísticas correspondentes.

4. Resultados e Discussão

Nesta Seção serão apresentados os resultados e análises comparativas do desempenho dos diferentes otimizadores nas tarefas de classificação de sentimentos e regressão de temperatura.

4.1. Comparativo do otimizadores na classificação

Os resultados do desempenho dos modelos com os diferentes otimizadores na tarefa de classificação de sentimentos são apresentados na Tabela 3, correspondendo à média e ao desvio padrão obtidos após a validação cruzada com 5 *folds*.

Esses resultados demonstram uma superioridade dos otimizadores Adam e RMSprop na tarefa de classificação de sentimentos em termos de métricas, assim como observado por [Eom 2021]. O MLP2 com RMSprop alcançou o melhor desempenho geral, com acurácia de 0.8847 ± 0.0073 e *F1-Score* de 0.8827 ± 0.0105. O Adam também apresentou resultados consistentes, com melhor desempenho no MLP1 (acurácia de 0.8760 ± 0.0138) e MLP3 (acurácia de 0.8670 ± 0.0189). Em contraste, SGD e Adagrad apresentaram desempenho significativamente inferior, com acurácias em torno de 0.52-0.53 em todas as arquiteturas.

Para validar estatisticamente estas observações, foi aplicado o teste de Friedman separadamente para cada arquitetura, revelando diferenças significativas entre os otimi-

[5] https://github.com/GabrieleAraujo/comparative-optimizer-deeplearning.git

Tabela 3. Resultados por Modelo e Otimizador na Classificação de Sentimentos

Arquitetura	Otimizador	Acurácia	*F1-Score*
MLP1	Adam	**0.8760 ± 0.0138**	**0.8734 ± 0.0192**
	RMSprop	0.8737 ± 0.0142	0.8733 ± 0.0196
	SGD com *momentum*	0.5243 ± 0.0032	0.5123 ± 0.0517
	Adagrad	0.5172 ± 0.0157	0.5723 ± 0.0387
MLP2	RMSprop	**0.8847 ± 0.0073**	**0.8827 ± 0.0105**
	Adam	0.8750 ± 0.0074	0.8768 ± 0.0101
	SGD com *momentum*	0.5280 ± 0.0068	0.4557 ± 0.1515
	Adagrad	0.5262 ± 0.0123	0.4931 ± 0.0929
MLP3	Adam	0.8670 ± 0.0189	**0.8718 ± 0.0132**
	RMSprop	**0.8741 ± 0.0111**	0.8702 ± 0.0166
	SGD com *momentum*	0.5389 ± 0.0173	0.5133 ± 0.1225
	Adagrad	0.5302 ± 0.0086	0.4530 ± 0.1791

zadores. A arquitetura MLP2 apresentou a evidência mais forte (estatística = 13.5600, p = 0.0035696), seguida pelo MLP1 (estatística = 12.6000, p = 0.0055865) e MLP3 (estatística = 12.1200, p = 0.0069832). Como todos os p-valores foram menores que 0.05, rejeitou-se a hipótese nula de que não há diferença entre os otimizadores, sendo necessário realizar uma análise post-hoc de Nemenyi para identificar quais pares de otimizadores diferem significativamente entre si, onde os resultados são apresentados na Figura 1.

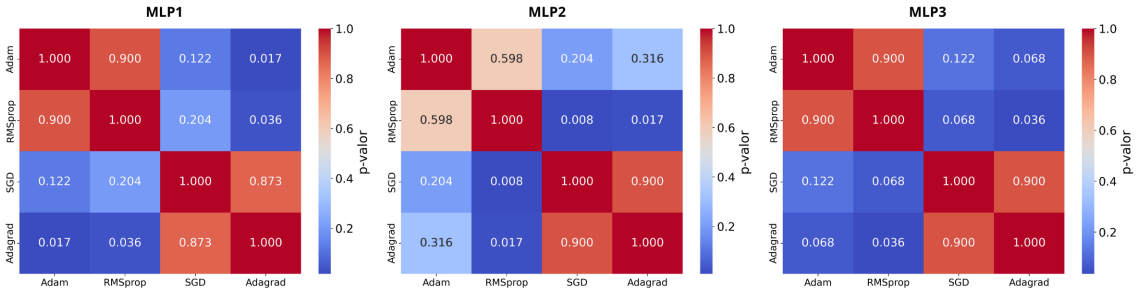

Figura 1. Resultados do teste post-hoc de Nemenyi para cada arquitetura

O teste confirmou que Adam e RMSprop formam um grupo estatisticamente similar (p = 0.900), enquanto diferem significativamente do SGD e Adagrad ($p < 0.05$). Por sua vez, SGD e Adagrad também formam um grupo sem diferença significativa entre si (p = 0.872). Esta separação em dois grupos de desempenho é evidenciada nos mapas de calor, onde tons mais escuros indicam maior similaridade estatística entre os otimizadores.

4.2. Comparativo dos otimizadores na regressão

Já para avaliar o desempenho dos otimizadores na tarefa de regressão, foi calculada a média e o desvio padrão do MAE ao longo dos 5 *folds* da validação cruzada temporal. Os resultados para cada combinação de modelo e otimizador são apresentados na Tabela 4.

Os resultados demonstram que o SGD com *momentum* apresentou o melhor desempenho em todas as arquiteturas testadas, com o menor erro registrado na arquitetura mais simples (MLP1), alcançando um MAE de 0.3973 ± 0.0549, entrando em consonância com os resultados de [Desai 2020]. O RMSprop e Adam apresentaram desem-

Tabela 4. Resultados de MAE por Modelo e Otimizador

Arquitetura	Otimizador	MAE
MLP1	SGD com *momentum*	**0.3973 ± 0.0549**
	RMSprop	0.4590 ± 0.1539
	Adam	0.7719 ± 0.2631
	Adagrad	23.4123 ± 0.7222
MLP2	SGD com *momentum*	**0.4140 ± 0.0669**
	RMSprop	0.4713 ± 0.1029
	Adam	0.5356 ± 0.1084
	Adagrad	11.3584 ± 3.9989
MLP3	SGD com *momentum*	**0.4513 ± 0.1523**
	Adam	0.4914 ± 0.1511
	RMSprop	0.5860 ± 0.1056
	Adagrad	7.0055 ± 2.7402

penhos intermediários, com o RMSprop se destacando ligeiramente nas arquiteturas mais simples (MLP1 e MLP2), enquanto o Adam mostrou melhor adaptação à arquitetura mais complexa (MLP3). O Adagrad apresentou resultados significativamente inferiores em todas as arquiteturas.

No teste de Friedman, arquitetura mais simples apresentou evidências mais fortes (MLP1: estatística = 14.0400, p = 0.0028512) em comparação com as mais complexas (MLP2: estatística = 12.1200, p = 0.0069832; MLP3: estatística = 9.9600, p = 0.0189092). Assim, os resultados do post-hoc são apresentados na Figura 2.

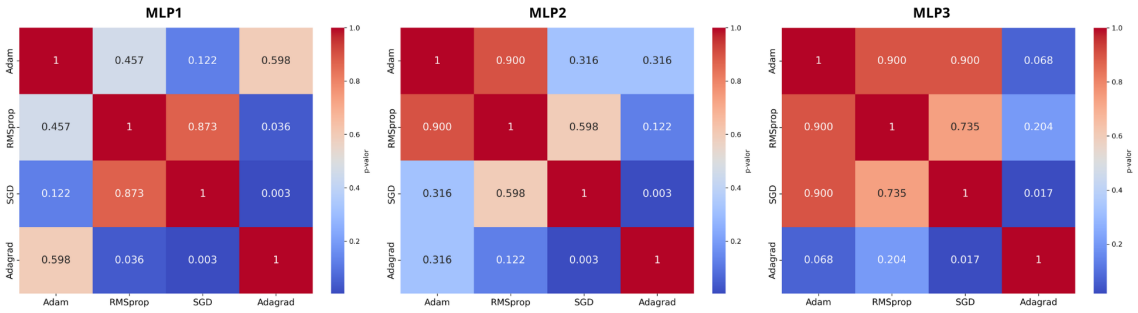

Figura 2. Resultados do teste post-hoc de Nemenyi para cada arquitetura

Como visto, na MLP1, o Adagrad mostrou diferenças significativas quando comparado ao RMSprop (p = 0.035540) e ao SGD com *momentum* (p = 0.003389). Este padrão se manteve nas arquiteturas MLP2 e MLP3, onde o Adagrad também diferiu significativamente do SGD com *momentum* (p = 0.003389 e p = 0.017331, respectivamente). O SGD com *momentum* e o RMSprop apresentaram comportamento estatisticamente similar em todas as arquiteturas ($p > 0.7$), enquanto o Adam não mostrou diferenças significativas com os demais otimizadores, exceto com o Adagrad. Os mapas de calor evidenciam estas relações, onde tons mais escuros indicam maior similaridade entre os otimizadores.

5. Considerações Finais

Este trabalho apresentou uma análise comparativa do desempenho de diferentes otimizadores em tarefas de classificação de sentimentos e regressão de dados meteorológicos,

utilizando três arquiteturas MLP e validação cruzada com 5 *folds*, além de testes estatísticos (Friedman e *post-hoc* de Nemenyi) para validar as diferenças observadas. Na classificação, utilizando dados do IMDB e validação cruzada estratificada, os otimizadores adaptativos Adam e RMSprop demonstraram superioridade estatisticamente significativa ($p < 0.05$), com o MLP2 utilizando RMSprop alcançando a melhor acurácia (0.8847 ± 0.0073), corroborando com resultados anteriores como observado em [Eom 2021]. Já para a regressão de temperatura em Belém, utilizando validação cruzada temporal, o SGD com *momentum* apresentou o melhor desempenho em todas as arquiteturas ($p < 0.05$), especialmente na mais simples (MLP1, MAE = 0.3973 ± 0.0549), alinhando-se com os achados de [Desai 2020].

Para trabalhos futuros, sugere-se uma investigação mais aprofundada das razões para o baixo desempenho de SGD e Adagrad, possivelmente explorando diferentes configurações de hiperparâmetros. Além disso, seria interessante expandir a análise para incluir outros tipos de arquiteturas de rede neural, como LSTMs ou *Transformers*, as quais são frequentemente utilizadas em tarefas de processamento de linguagem natural.

Referências

Araújo, G. d. S., Leite, J. B. P., da Silva, M. S., Junior, A. F. J., and Lobato, F. M. (2023). Natural language processing and social media: a systematic mapping on brazilian leading events. In *Anais do XX Encontro Nacional de Inteligência Artificial e Computacional*, pages 741–755. SBC.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

Desai, C. (2020). Comparative analysis of optimizers in deep neural networks. *International Journal of Innovative Science and Research Technology*, 5(10):959–962.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Eom, S. H. (2021). Developing sentimental analysis system based on various optimizer. *International Journal of Internet, Broadcasting and Communication*, 13(1):100–106.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liu, Y., Gao, Y., and Yin, W. (2020). An improved analysis of stochastic gradient descent with momentum. *Advances in Neural Information Processing Systems*, 33:18261–18271.

Nemenyi, P. (1963). *Distribution-free multiple comparisons*. PhD thesis, Princeton University.

Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

Pomerat, J., Segev, A., and Datta, R. (2019). On neural network activation functions and optimizers in relation to polynomial regression. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6183–6185. IEEE.

Rainio, O., Teuho, J., and Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, 14(1):6086.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Saleem, M. H., Potgieter, J., and Arif, K. M. (2020). Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers. *Plants*, 9(10):1319.

Shrivastava, V. K., Shrivastava, A., Sharma, N., Mohanty, S. N., and Pattanaik, C. R. (2023). Deep learning model for temperature prediction: an empirical study. *Modeling Earth Systems and Environment*, 9(2):2067–2080.

Sun, R.-Y. (2020). Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 8(2):249–294.

Tieleman, T. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26.