Trabalho Computacional III – Busca/Otimização Meta-heurística

Yasmim Santos Rodrigues - 2116925  
*Ciência da Computação*  
*Universidade de Fortaleza*Fortaleza, Brasil  
ysrodrigues@edu.unifor.br

Samuel Zaire Rocha - 2110997  
*Ciência da Computação  
Universidade de Fortaleza*Fortaleza, Brasil  
samuel.zairerocha@edu.unifor.br

*Resumo*—Este artigo descreve a implementação e avaliação de modelos baseados em redes neurais artificiais em duas etapas distintas de experimentação. Na primeira etapa, foram empregados os modelos lineares Perceptron Simples e ADALINE (Adaptive Linear Element) para lidar com um problema de classificação binária utilizando dados sintéticos. Na segunda etapa, utilizou-se o modelo de Perceptron de Múltiplas Camadas (MLP) para realizar reconhecimento facial a partir de um conjunto de dados abrangendo 1888 arquivos de imagens de 20 indivíduos distintos. Em ambos os cenários, o paradigma de aprendizado supervisionado foi seguido, no qual os modelos foram treinados com amostras e valores observados para executar tarefas de classificação. Ao final da execução de cada fase da primeira etapa, foram calculadas as médias, desvios padrão e valores máximo e mínimo da acurácia, sensibilidade e especificidade dos resultados obtidos, sendo que, para a etapa de classificação binária, o algoritmo ADALINE obteve um desempenho relativamente melhor. Para a segunda etapa, o MLP foi ajustado quanto às camadas e quantidade de neurônios para representação dos casos de superdimensionamento e sobredimensionamento, além do caso de topologia ideal. Além disso, a técnica de *early stopping* foi implementada para fins de comparação. Por fim, foi possível encontrar um equilíbrio que permitisse alta acurácia na classificação e convergência do modelo.

Palavras-chave—Redes Neurais Artificiais; Perceptron Simples; ADALINE; Perceptron de Múltiplas Camadas (MLP).

# INTRODUÇÃO

Na área de inteligência artificial, redes neurais artificiais (RNA) são modelos computacionais inspirados no sistema nervoso de seres vivos, definidas por um conjunto de unidades de processamento caracterizadas por neurônios artificiais interconectados através de uma matriz de pesos que representam sinapses artificiais.

Uma forma de representar o neurônio biológico é pelo modelo de neurônio de Mculloch-Pitts (M-P), que possui entradas (x1 e x2), bem como pesos sinápticos (w), limiar (θ) e ativação (u). Com x1 e x2, é possível traçar uma reta que separe duas classes bem isoladas uma da outra. Dois algoritmos que podem resolver tais problemas de classificação binária são o Perceptron Simples e ADALINE (*Adaptive Linear Element*).

Para o Perceptron Simples, o neurônio M-P é dotado de uma regra de aprendizagem cujos parâmetros - pesos e limiar – são ajustados de forma guiada pela informação fornecida na entrada e pelo erro entre a saída desejada e a gerada pela rede. Já ADALINE tem seus parâmetros ajustados por meio de uma regra de atualização recursiva de forma a identificar um conjunto ótimo de parâmetros que minimize o Erro Quadrático Médio (EQM) do modelo. Ambos utilizam uma função de ativação do tipo degrau, e a saída pode possuir apenas dois valores.

Outro tipo de rede neural artificial é o de Redes Perceptron Multicamadas (MLP), caracterizado pela presença de pelo menos uma camada intermediária de neurônios, situada entre as camadas de entrada e saída. Diferentemente dos algoritmos mencionados anteriormente, uma rede MLP pode ter diversos neurônios na camada de saída.

A primeira etapa deste trabalho, por lidar com dados sujeitos a classificação binária, envolveu a implementação – treinamento e teste - tanto do Percepton Simples e do ADALINE durante um número definido de rodadas. Para a segunda etapa, como os dados possuíam bem mais que duas classes, o modelo MLP foi treinado e testado para tal.

Tais implementações são discutidas e detalhadas na seção de desenvolvimento juntamente com a descrição do conjunto de dados. A seção de resultados, logo após, traz medidas estatísticas como acurácia, sensibilidade, especificidade calculadas durante o processo, bem como gráficos e tabelas para ilustrar a discussão em torno do desempenho dos algoritmos implementados.

# DESENVOLVIMENTO

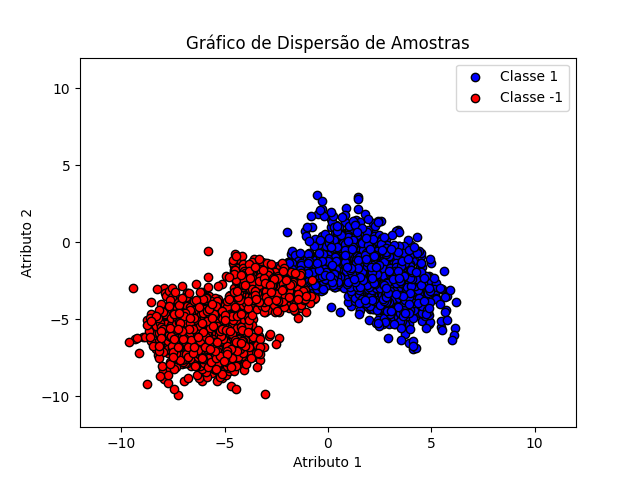
Os conjuntos de dados referentes a cada etapa foram importados e tratados para visualização. Primeiramente, para a tarefa de classificação binária, o arquivo "*DataAV2.csv*" foi gerado através do algoritmo suporte fornecido *GerarDados\_Parte1*; os dados foram sintetizados em formato de planilha com 3600 linhas e 3 colunas, em que as duas primeiras colunas referem-se aos atributos das amostras e, a terceira coluna, à classe a qual elas pertencem, sendo estas -1 ou 1.

Para a segunda etapa, foram utilizados arquivos de imagens separados por indivíduo. No total, 1888 imagens de 20 pessoas distintas foram tratadas através do algoritmo suporte *OrganizaImagens\_Parte2* que realiza a aquisição de tais imagens e as organizam em um conjunto de dados com seus rótulos associados.

## Implementação da tarefa de minimização/maximização de função custo/objetivo

### Visualização inicial e hipóteses

Com a planilha gerada, foi feita uma visualização inicial dos seus dados através do gráfico de dispersão, evidenciando as duas classes possíveis para os dados (1 e -1), como é possível ver na figura 1.



1. Gráfico de dispersão de *DataAV2.csv*

Os dados exibem características que sugerem a ausência de uma separação linear clara, com diversas áreas de sobreposição entre as classes. O propósito fundamental dos algoritmos de classificação binária reside na tarefa de diferenciar as amostras em duas categorias distintas, tomando como base suas características. Apesar do Perceptron Simples buscar realizar essa distinção por meio da otimização dos pesos de tal forma que possa convergir quando os dados de treinamento são linearmente separáveis, este cenário não se aplica aos dados deste estudo. Em contraste, o ADALINE é capaz de encontrar uma solução mesmo quando os dados não puderem ser claramente separados por uma fronteira linear. Como resultado, foi esperado que, ao considerar as métricas estatísticas, o algoritmo ADALINE demonstrasse médias superiores em termos de acurácia, sensibilidade e especificidade ao término da análise.

### Definição das variáveis

Após importação e visualização do conjunto de dados, estes foram organizados para que a variável ***X***, representante do conjunto de atributos da amostra, tivesse uma nova dimensão, sendo adicionada a ela uma coluna de 1 (uns), também conhecida como viés ou bias.

Para ambos os modelos foi definido o passo de aprendizagem η como 0,001. Tal valor é como o modelo ajusta seus pesos para se tornar mais preciso, melhorando suas previsões. Um valor muito pequeno pode levar a um treinamento lento, enquanto um valor muito grande pode fazer o modelo oscilar e nunca convergir para uma solução estável. O valor escolhido para o presente trabalho deve-se à definição feita em sala em algoritmos pré-disponibilizados.

Para o modelo ADALINE apenas, foi definido o valor de precisão como 0,0000001. A precisão é usada para determinar quando o treinamento do ADALINE deve parar. Se a diferença entre o erro quadrático médio (EQM) das épocas consecutivas for menor que a precisão definida, o treinamento será interrompido. Tal valor também foi selecionado a partir de definição realizada em sala.

### Aplicação dos modelos

A aplicação dos modelos previamente estabelecidos se deu, primeiramente, com a necessidade de se criar várias rodadas de treino para aplicar os dados de teste. Treino ou treinamento se refere à fase que o modelo ajusta seus parâmetros para tentar fazer previsões corretas com base em exemplos que contenham respostas corretas. Teste é um conjunto de dados separado que será aplicado como input “inédito” de dados para o modelo treinado a fim de determinar se dado modelo atende as expectativas de uma análise estatística determinada. Para definir ambos os elementos, por convenção estabelecida pelo material de apoio ao artigo em questão, foram utilizados 80% dos dados para treinamento e 20% dos dados para teste através de um método de embaralhamento de dados.

Para ambos os modelos, o número de rodadas e o de épocas foi de 100, ou seja, cada rodada fazia a execução de 100 épocas para treino do algoritmo, para que, ao final de cada rodada, se realizasse o teste com base nas amostras de teste. Além disso, o final de cada rodada também armazenava dados como acurácia, sensibilidade e especificidade após os testes, para que, ao final da execução dos algoritmos, fossem calculadas as medidas de tais valores.

Apesar de ambos algoritmos seguirem uma estrutura semelhante para treinamento e teste, há diferenças importantes a serem apontadas. O Perceptron Simples realiza o treinamento seguindo a condição de existência de erro, parando apenas quando não encontrar mais erro. Já o ADALINE para o treinamento quando a diferença entre o EQM das épocas consecutivas for menor que o valor de precisão. Como os dados não são linearmente separáveis, foi necessário estabelecer, também, que o treinamento parasse caso o número máximo de épocas fosse atingido, de forma a evitar repetições infinitas de treinamento. Ambos os algoritmos tiveram tal condição adicionada aos seus *loops* de treinamento.

O treinamento do Perceptron Simples envolve, primeiramente, inicialização dos pesos com valores aleatórios ou zeros, para então seguir para um *loop* que percorre todas as amostras do conjunto de treinamento. Para cada uma dessas amostras, é calculada a soma ponderada dos atributos multiplicados pelos pesos, seguida da aplicação da função de ativação, que, como discutido anteriormente, é a função degrau que produz uma saída binária. Sempre que a saída do Perceptron não corresponder à classe desejada, os pesos são ajustados com base na diferença entre a saída real e a saída desejada, multiplicada pelo valor dos atributos. Os pesos são, então, atualizados para minimizar o erro, com a taxa de aprendizado controlando o tamanho do ajuste. Por fim, é verificado se houve alguma mudança nos pesos durante a época. Não havendo mudanças, o treinamento é concluído pois o modelo convergiu; caso contrário, o *loop* é repetido para a próxima época.

O ADALINE realiza o treinamento de forma similar ao Perceptron, inicializando pesos e perpassando pelas amostras em *loop*. No cálculo da saída, porém o ADALINE calcula a soma ponderada dos atributos multiplicados pelos pesos sem aplicar uma função de ativação, resultando em uma saída contínua. O erro é calculado comparando a saída real com a saída desejada, e é então utilizado para ajustar os pesos. A taxa de aprendizagem também é utilizada nesses ajustes, porém os pesos são atualizados com o objetivo de minimizar o EQM em relação às amostras de treinamento. O algoritmo então verifica se houve alguma mudança nos pesos durante a época: se não houver mudança ou a mudança for menor que a precisão definida, o modelo convergiu e o treinamento é concluído; caso contrário, o *loop* continua na próxima época.

Os testes de ambos algoritmos realizam a classificação de novas amostras com base nos pesos aprendidos durante o treinamento, indicando a qual das duas classes a amostra pertence. Os valores reais e preditos são utilizados para preencher uma matriz de confusão que é usada para calcular a acurácia, sensibilidade e especificidade.

## Implementação da tarefa de domínio discreto

### Coleta de dados e definição das variáveis

A etapa de reconhecimento facial envolveu, como dito anteriormente, a implementação do algoritmo de Perceptron de Múltiplas Camadas (MLP). O conjunto de dados utilizado foi gerado a partir do algoritmo suporte *OrganizaImagens\_Parte2,* que recebeu uma pasta contendo 20 pastas que, por sua vez, continham 1888 arquivos de imagem no total. As 20 pastas referem-se aos indivíduos presentes nas fotos, e estas seguiam o seguinte padrão para nome: *<userid> <pose> <expressão> <olhos> <escala> .pgm*. Algumas fotos terminavam com *.bad* propositalmente. O algoritmo de organizar imagens perpassou por todos os arquivos e fez redimensionamento de cada um para 60x60, a dimensão padrão do algoritmo.

Ao final do redimensionamento, o conjunto de dados possuiu 640 amostras, 3600 preditores (60x60) e 20 classes. Assim, as duas matrizes *X* e *Y* ficaram com tamanho 3600x640 e 20x640, respectivamente.

Assim como o Perceptron Simples e o ADALINE, o algoritmo MLP também faz uso do passo de aprendizagem η, definido como 0,05, ajustado diversas vezes durante o processo de implementação a fim de encontrar um valor que não fosse muito alto de forma que o treinamento não convergisse, nem muito baixo, tornando o treinamento lento.

Um valor de precisão também foi testado e ajustado constantemente para servir de condição de parada do treinamento; enquanto o EQM calculado continuasse se sobrepondo ao valor de precisão, o treinamento continuaria.

Outra condição de parada do treinamento foi o número máximo de épocas, também reajustado constantemente e definido como 1000.

### Aplicação do modelo

O algoritmo MLP é caracterizado pela presença de pelo menos uma camada intermediária ou oculta de neurônios, situada entre a entrada e a saída. A definição do número de camadas ocultas *L* e da quantidade de neurônios em cada depende de experimentações e reajustes.

A rede MLP faz uso do aprendizado supervisionado com arquitetura *feedforward*, ou seja, cada neurônio em qualquer camada da rede está conectado a todos os neurônios da camada anterior e o sinal progride da esquerda para a direita.

Uma das diferenças do MLP para os algoritmos descritos anteriormente neste trabalho refere-se à quantidade de neurônios na camada de saída: a camada de saída do MLP pode possuir múltiplos neurônios. Para o presente trabalho, a camada de saída foi de 20, referente às 20 classes. Além disso, diferentemente do Perceptron Simples e do ADALINE, em que um único neurônio era responsável pelo mapeamento, o conhecimento relacionado ao comportamento entrada/saída é distribuído por todos os neurônios da rede e, conforme o processo de aprendizagem da rede avança, estes neurônios começam a descobrir as características salientes presentes nos dados.

Outra diferença do MLP é a função de ativação, sigmoidal, que permite que a saída seja uma variável real. Para o presente trabalho, a função de ativação utilizada foi a de tangente hiperbólica, caracterizada como *g(x)* no código, e sua derivada caracterizada como *g\_linha(x)*.

O treinamento do MLP se dá com a aplicação de duas fases: propagação adiante (*forward*) e propagação reversa (*backward*). Na primeira, os sinais de entrada de uma amostra do conjunto de treinamento são propagados através das camadas até a saída. As respostas produzidas são comparadas com as respostas desejadas, produzindo um sinal de erro. A segunda fase ajusta os pesos e limiares. As aplicações sucessivas de ambas as fases fazem com que os pesos e limiares se ajustem em cada iteração com a finalidade de diminuir a soma dos erros.

A configuração da topologia requereu a inicialização de listas para armazenar ***WL***, a *L*-ésima matriz de pesos, ***iL***, o *L*-ésimo vetor de entrada ponderada para a *L*-ésima camada, e ***yL***, o *L*-ésimo vetor de saída após a aplicação da função de ativação em cada neurônio na camada *L*.

A fase de testes envolveu o embaralhamento de dados de forma que os dados de treino abrangeram 80% da amostra, e, os dados teste, 20%. Além disso, os dados foram normalizados, ou seja, tiveram a escala original modificada por meio do método min-max.

Um ponto a ser abordado neste trabalho é de treinamentos inadequados causados por possíveis subdimensionamento ou sobredimensionamento da rede MLP. Nesses casos, a capacidade de generalização, ou seja, de aprendizagem da relação entrada-saída do mapeamento, é ruim. O subdimensionamento leva ao *underfitting*, que ocorre quando a rede não tem poder computacional suficiente para aprender o mapeamento de interesse, resultando em baixas taxas de acurácia tanto na fase de treino quanto de teste. O *overfitting*, causado pelo sobredimensionamento, ocorre quando a rede possui neurônios ocultos demais e memoriza os dados de treinamento, resultando em uma alta taxa de acurácia para a fase de treino, mas baixa acurácia para a fase de teste.

Ambos os casos foram testados com base em diferentes configurações topológicas da rede. Para o caso de *underfitting*, utilizou-se apenas uma camada oculta com 2 neurônios apenas. Para o *overfitting*, foram utilizadas 4 camadas ocultas com 200 neurônios cada.

Outro ponto abordado neste trabalho foi o uso de *early stopping*, ou parada prematura, um método em que o conjunto de treinamento é dividido em duas partes, uma para estimação dos parâmetros e outra para validação durante o treinamento. Este último deve ser utilizado a cada número de épocas de treinamento especificados previamente; no caso do algoritmo implementado, a cada 5 épocas de treinamento o EQM de validação era medido, com o objetivo de interromper o treinamento quando o EQM de validação assumisse uma tendência de crescimento, que é um indicativo de que a rede está começando a se especializar demais nos dados de estimação, podendo levar a um *overfitting*.

Bem como o Perceptron Simples e o ADALINE, a implementação do MLP neste trabalho envolveu o uso de matrizes de confusão, porém de tamanho 20x20, para visualização dos casos de *underfitting*, *overfitting*, e no caso da topologia ideal escolhida. A acurácia média, seu desvio padrão e valores máximo e mínimo também foram computadas para esta última.

# Resultados

A presente seção trata dos resultados obtidos para ambas as tarefas executadas, apresentando discussão sobre os modelos e confirmações ou não de hipóteses previamente levantadas.

## Resultados da tarefa de minimização/maximização de função custo/objetivo

Como mencionado na seção de desenvolvimento, as medidas de acurácia, sensibilidade e especificidade foram calculadas e armazenadas em listas para que se pudesse calcular suas respectivas médias, desvios padrão e valores máximo e mínimo.

Para isso, uma matriz de confusão foi gerada a cada rodada de teste, para que assim as medidas fossem calculadas. A matriz de confusão é uma tabela comumente usada em problemas de classificação para avaliar o desempenho do modelo. Este estudo gerou matrizes de 2x2, com quadrantes que representaram os seguintes elementos:

* **Verdadeiro Positivo (VP):** número de amostras classificadas corretamente como pertencentes à classe 1.
* **Verdadeiro Negativo (VN):** número de amostras classificadas corretamente como pertencentes à classe -1.
* **Falso Positivo (FP):** número de amostras erroneamente classificadas como pertencentes à classe 1 quando na verdade eram da classe -1.
* **Falso Negativo (FN):** número de amostras erroneamente classificadas como pertencentes à classe -1 quando na verdade eram da classe 1.

Assim, VP e VN são medidas de acertos do modelo, enquanto FP e FN são medidas de erro. Tais medidas são utilizadas para calcular acurácia, sensibilidade e especificidade.

A acurácia foi calculada seguindo a fórmula:

Quanto mais próxima de 1, melhor a acurácia.

As medidas, desvios padrão e valores máximos e mínimos das acurácias do Perceptron Simples e ADALINE foram computadas no quadro 1.

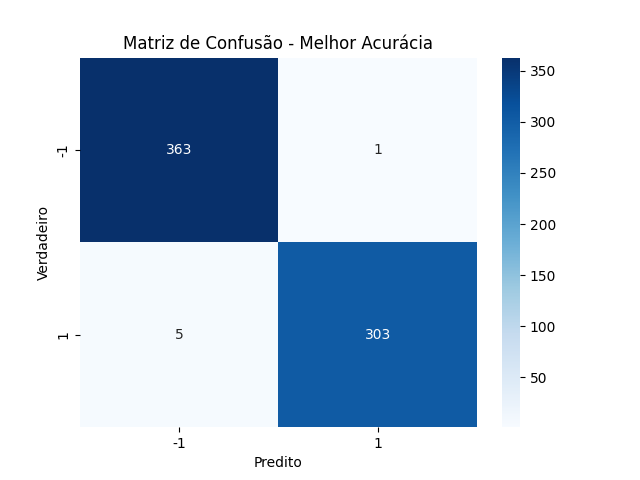
Como é possível perceber, ambos os modelos tiveram médias de acurácia altas, próximas de 1, com o modelo ADALINE obtendo um desempenho relativamente melhor. Além disso, os valores máximo e mínimo também ficaram um pouco acima do Perceptron Simples. O desvio padrão maior do Perceptron Simples também indica uma variação maior da acurácia de uma execução para a outra, o que significa que o modelo é sensível a diferentes conjuntos de dados ou inicializações de pesos se comparado com o desempenho do ADALINE.

**Quadro 1** - Comparação da Acurácia

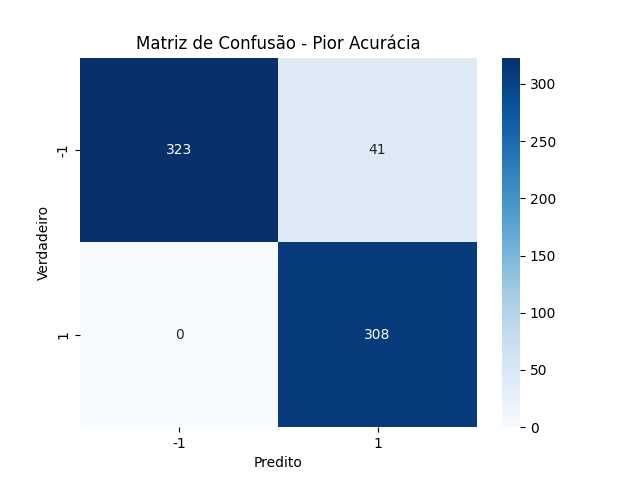
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acurácia | | | | |
| Modelos | Média | Desvio padrão | Máximo | Mínimo |
| Perceptron Simples | 0.977083 | 0.00939 | 0.991071 | 0.938988 |
| ADALINE | 0.978899 | 0.00559 | 0.99256 | 0.965774 |

Isso confirma que o fato de o Perceptron Simples não lidar tão bem com conjuntos de dados que não são linearmente separáveis levou a um desempenho pior para este.

O maior e pior valor de acurácia de ambos os modelos foram computados juntamente com as matrizes de confusão respectivas à rodada em que foram obtidos. As matrizes, então, foram geradas, como se pode ver nas Figuras 2 e 3, para o Perceptron, e Figuras 4 e 5, para o ADALINE.

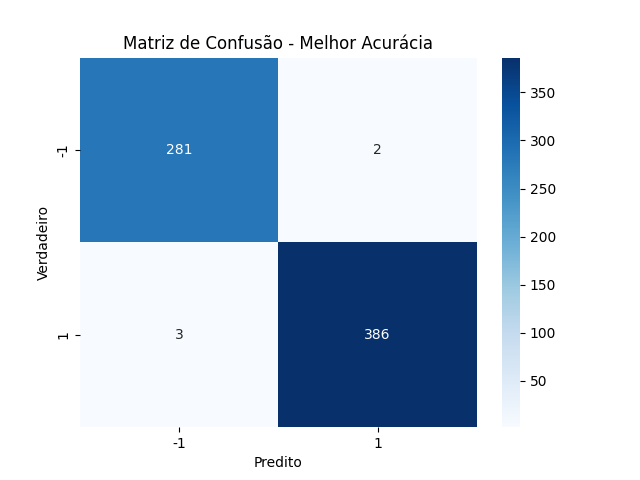


1. Matriz de confusão da melhor acurácia obtida pelo Perceptron Simples.

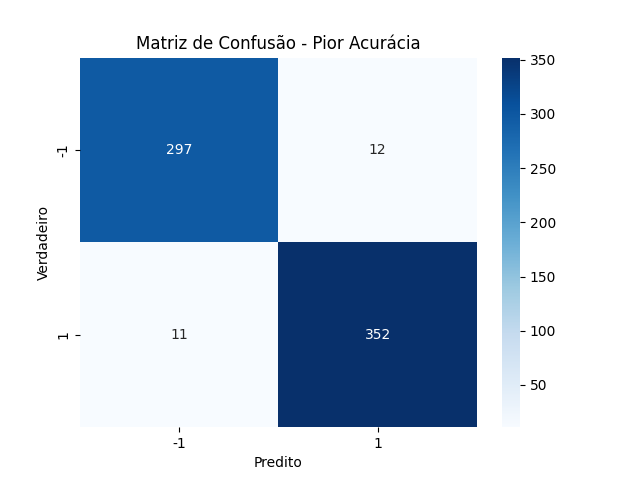


1. Matriz de confusão da pior acurácia obtida pelo Perceptron Simples.

Na Figura 2, os falsos negativos e positivos não tiveram números expressivos, totalizando 6 classificações errôneas, já na Figura 3, apesar de não ter havido falsos negativos, o número de falsos positivos foi bastante elevado, com 41 classificações errôneas.



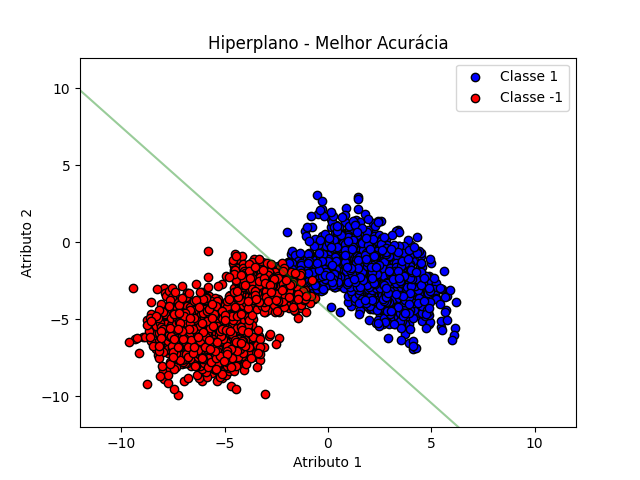
1. Matriz de confusão da melhor acurácia obtida pelo ADALINE.



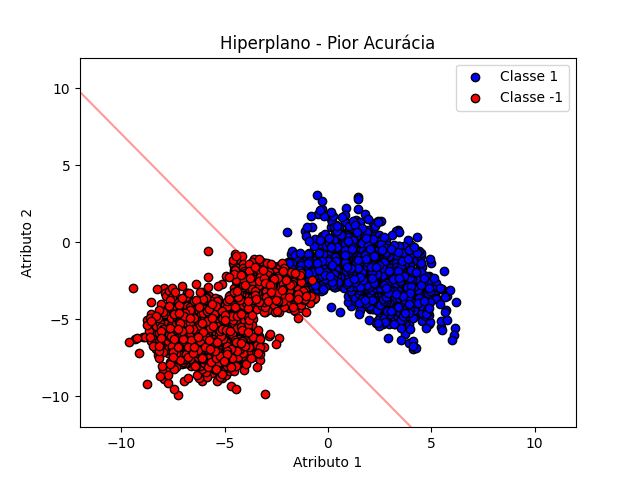
1. Matriz de confusão da pior acurácia obtida pelo ADALINE.

As matrizes de confusão do ADALINE mostram valores menores para os erros, com 5 classificações errôneas na rodada de melhor acurácia, e 23 classificações errôneas na rodada de pior acurácia.

É possível visualizar as tentativas de separação das classes nos gráficos gerados nas figuras 7 e 8 para o Perceptron Simples, e nas figuras 9 e 10 para o ADALINE.

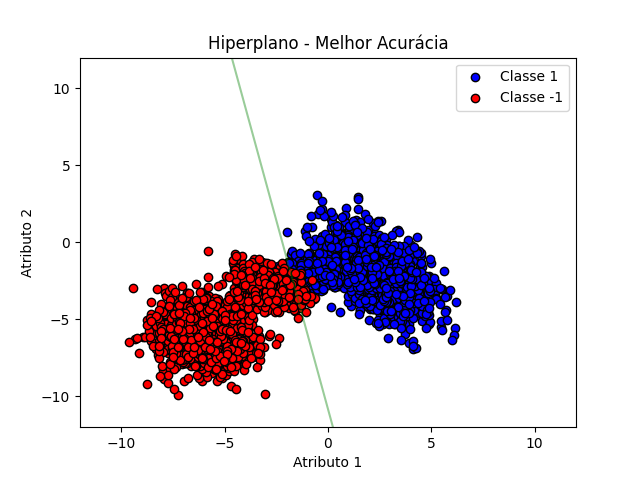


1. Gráfico que mostra o hiperplano de separação do Perceptron Simples para a rodada com melhor acurácia.

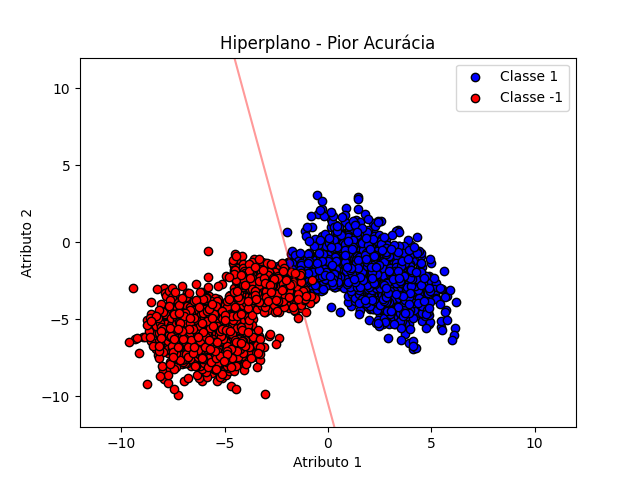


1. Gráfico que mostra o hiperplano de separação do Perceptron Simples para a rodada com pior acurácia.

A olho nu, é possível ver que a reta traçada na figura 6 está mais próxima dos pontos em que as classes se sobrepõem, já a reta da figura 7 está bastante distante de tais pontos, separando uma parte considerável da classe -1, o que pode ter levado ao alto número de falsos positivos obtido.



1. Gráfico que mostra o hiperplano de separação do ADALINE para a rodada com melhor acurácia.



1. Gráfico que mostra o hiperplano de separação do ADALINE para a rodada com pior acurácia.

Ambas as retas geradas pelo ADALINE para tentativa de separação das classes são bastantes semelhantes em sua inclinação e posição, evidenciado pelos baixos valores de desvio padrão obtidos pelo modelo em comparação com o Perceptron Simples, o que significa baixa sensibilidade às variações nas amostras, fazendo com que tanto a pior quanto melhor rodada no quesito acurácia tenham tido um desempenho quase que igualmente bom.

Os valores de sensibilidade também foram calculados e computados no quadro 2.

**Quadro 2** - Comparação da Sensibilidade

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sensibilidade | | | | |
| Modelos | Média | Desvio padrão | Máximo | Mínimo |
| Perceptron Simples | 0.977584 | 0.01588 | 1.000000 | 0.926396 |
| ADALINE | 0.976846 | 0.01108 | 0.99670 | 0.942675 |

A sensibilidade mede a proporção de amostras da classe positiva corretamente classificadas em relação ao número total de amostras da classe positiva. Ela segue a seguinte fórmula:

A diferença entre os desempenhos do Perceptron Simples e do ADALINE foi baixa, e, em alguns casos, como na média e no valor máximo, o Perceptron Simples se sobressaiu. Isso se deve ao fato de que ambos os algoritmos obtiveram uma acurácia bastante elevada, pois classificaram corretamente as amostras de forma geral.

Por fim, a especificidade também foi medida e computada para ambos os algoritmos, e seus resultados podem ser analisados no quadro 3.

**Quadro 3** - Comparação da Especificidade

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Especificidade | | | | |
| Modelos | Média | Desvio padrão | Máximo | Mínimo |
| Perceptron Simples | 0.978135 | 0.02447 | 1.000000 | 0.882521 |
| ADALINE | 0.980670 | 0.00807 | 1.00000 | 0.963158 |

A especificidade mede a proporção de amostras da classe negativa corretamente classificadas em relação ao número total de amostras da classe negativa, seguindo a fórmula:

O ADALINE obteve resultados melhores de especificidade, com todas as métricas acima do Perceptron Simples.

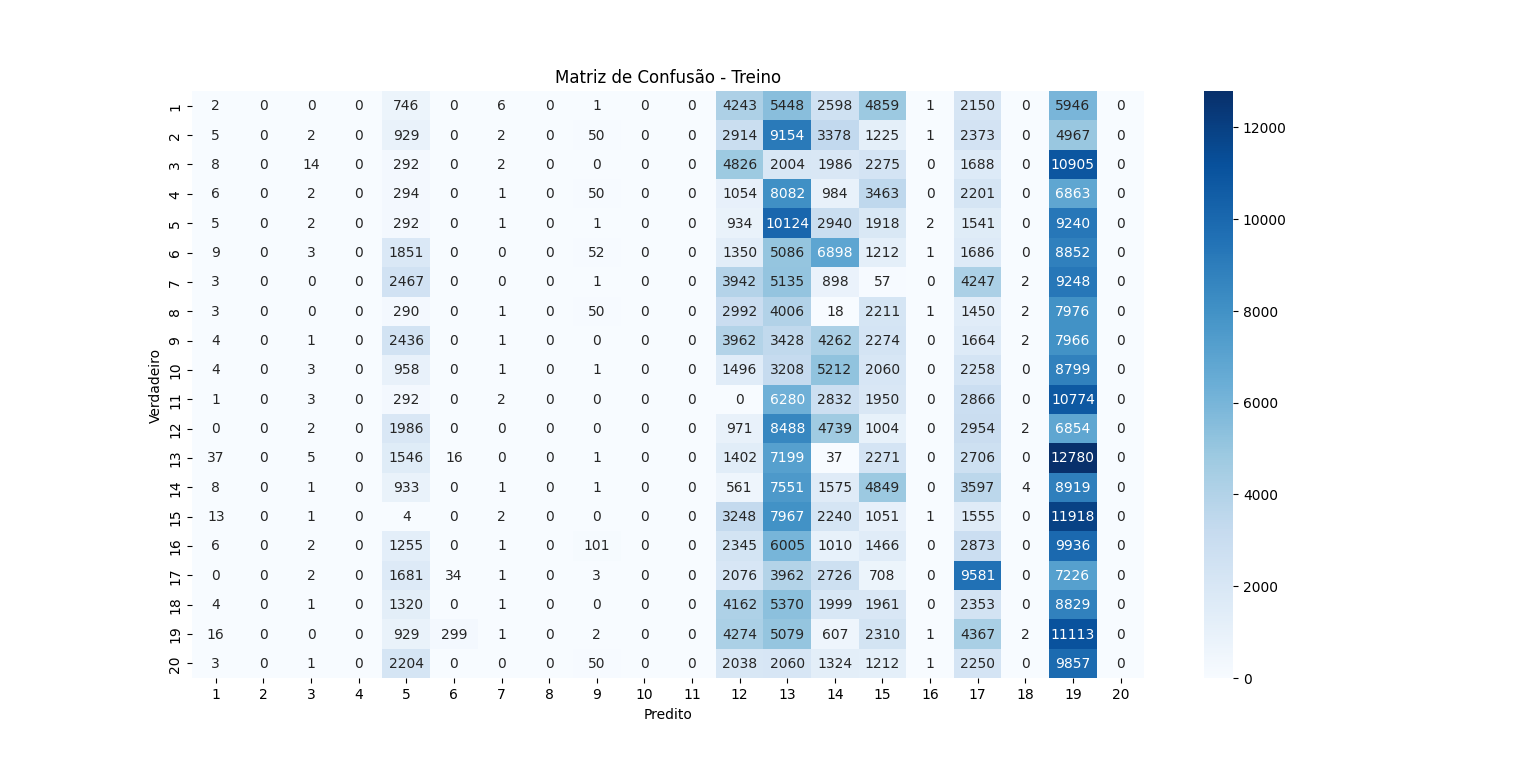
Em geral, o ADALINE teve um desempenho melhor dos dois modelos, confirmando novamente que os dados não linearmente separáveis impossibilitou o Perceptron Simples de se sair melhor nos testes.

## Resultados da tarefa de domínio discreto

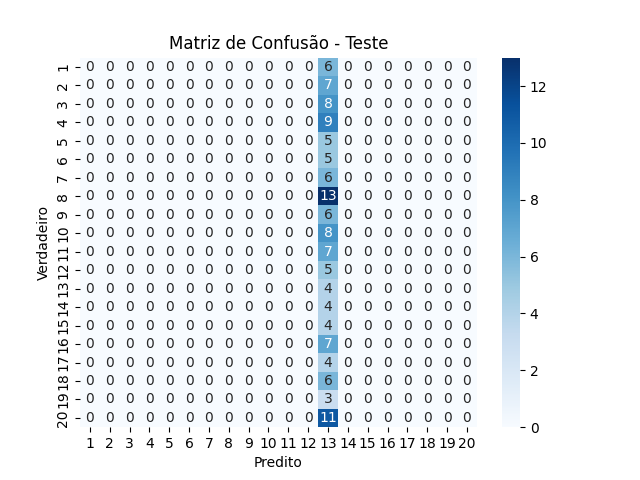
Os resultados das execuções do MLP para o conjunto de dados descrito anteriormente foram computados em matrizes de confusão para a fase de treino e de teste e em tabelas para valores de acurácia. Para este trabalho, foram testados casos de *overfitting* e *underfitting*, dimensionamento ideal e *overfitting* com *early stopping*.

Com apenas 1 camada de 2 neurônios e precisão de 0,01, o MLP passou pelas 1000 épocas com uma média de acurácia na fase de treino de 4,69% e, na fase de teste, de 0,78%.

Para ambas as fases, foram geradas as matrizes de confusão nas figuras 10 e 11.



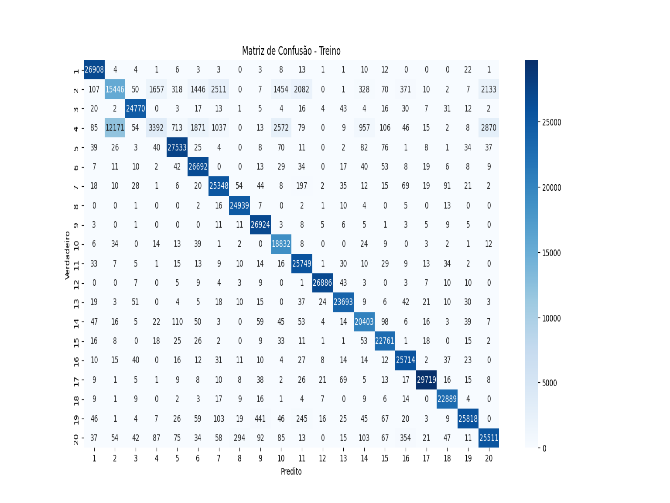
1. Matriz de confusão da fase de treino do MLP para o caso de *underfitting*.



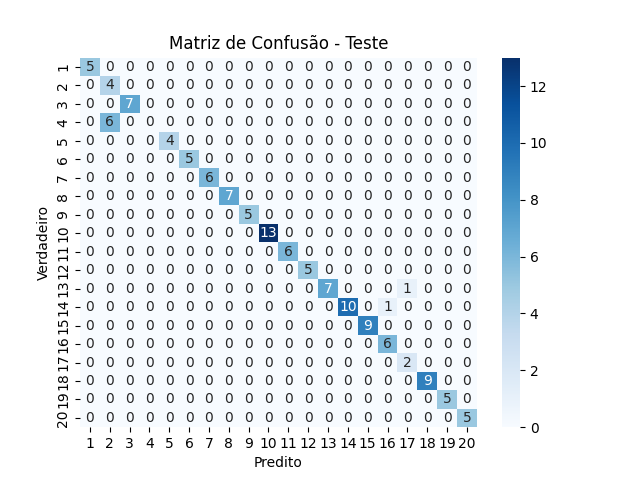
1. Matriz de confusão da fase de teste do MLP para o caso de *underfitting*.

Na matriz de confusão 20x20, a diagonal é populada pelos acertos. Como é possível perceber, ambas as matrizes das figuras 10 e 11 não tiveram as diagonais populadas pelos acertos, reflexo da baixa acurácia nas fases. Isso se deve ao número baixo de camadas e neurônios, não sendo suficiente para o modelo aprender as relações entrada/saída, levando ao *underfitting*.

Para a tentativa de reprodução de caso de *overfitting*, as 4 camadas ocultas com 200 neurônios cada geraram as matrizes de confusão das figuras 12 e 13.



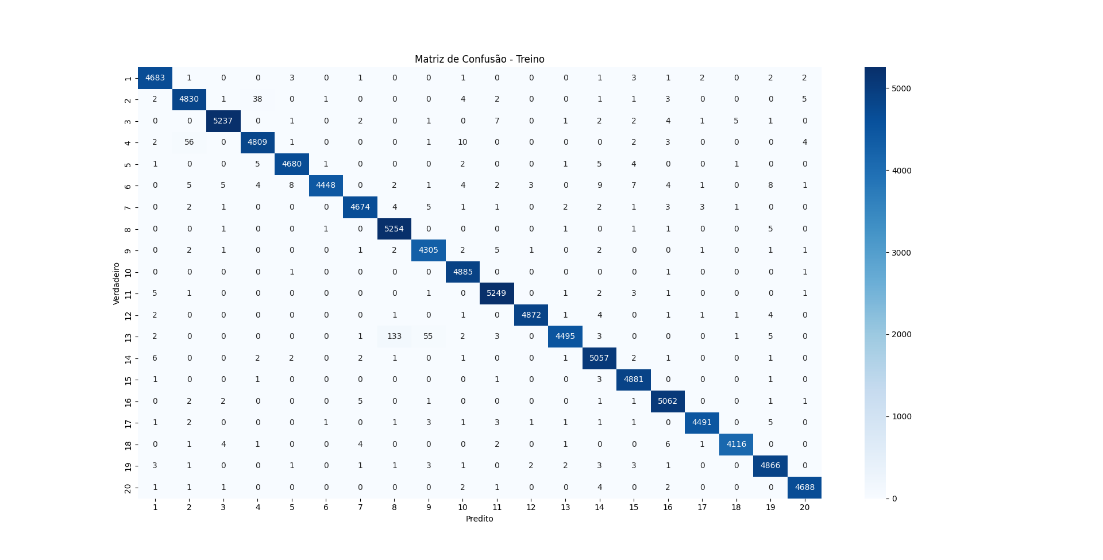
1. Matriz de confusão da fase de treino do MLP para o caso de *overfitting*.



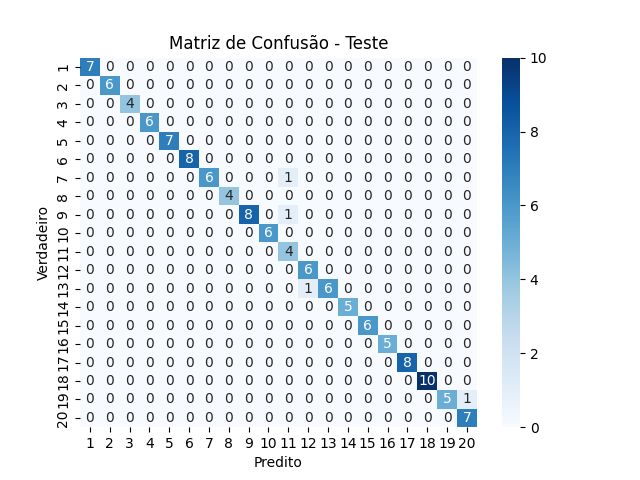
1. Matriz de confusão da fase de teste do MLP para o caso de *overfitting*.

Como é possível perceber, mesmo com um suposto superdimensionamento, o modelo ainda manteve alta acurácia para treino e teste (91,78% e 93,75%, respectivamente), o que sugere que o *overfitting* esperado não aconteceu, e que os números de camadas e neurônios não foram suficientes para que o modelo fosse considerado superdimensionado. Devido à limitação de poder computacional utilizado para o trabalho, contudo, tornou-se difícil executar um modelo superdimensionado para além dos valores descritos.

Em contrapartida, ajustando os valores das camadas, neurônios e precisão, foi estabelecida a topologia ideal para este trabalho: 2 camadas ocultas com 100 e 50 neurônios, respectivamente, e precisão de 0,001. Os resultados podem ser observados nas figuras 14 e 15.



1. Matriz de confusão da fase de treino do MLP para a topologia ideal.



1. Matriz de confusão da fase de teste do MLP para a topologia ideal.

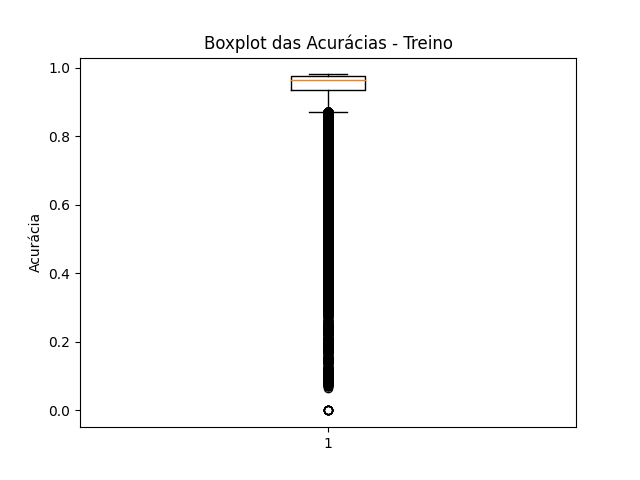
As matrizes de confusão de treino e de teste mostram as diagonais populadas inteiramente, com quase nenhum outro elemento fora delas preenchido, representando a taxa de acerto de treino de 99,3% e, de teste, 96,8%.

No Quadro 4, a acurácia média, seu desvio padrão e valores máximo e mínimo da fase de treino podem ser observados, confirmando novamente a taxa de acerto elevada, com pouca variação entre os valores.

**Quadro 4** – Valores da acurácia de treino para a topologia ideal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Medida | Média | Desvio padrão | Máximo | Mínimo |
| Acurácia | 0.968979 | 0.07582 | 0.992998 | 0.0 |

Tais valores foram computados e dispostos na forma de um *boxplot*, como se pode ver na figura 16.



1. Boxplot das acurácias de treino do MLP para a topologia ideal.

A fase de treino foi executada 10 vezes com essa mesma configuração topológica de forma a calcular uma média do número de épocas necessário para a convergência. O resultado foi que o MLP tendia a atingir a convergência aproximadamente na época 319.

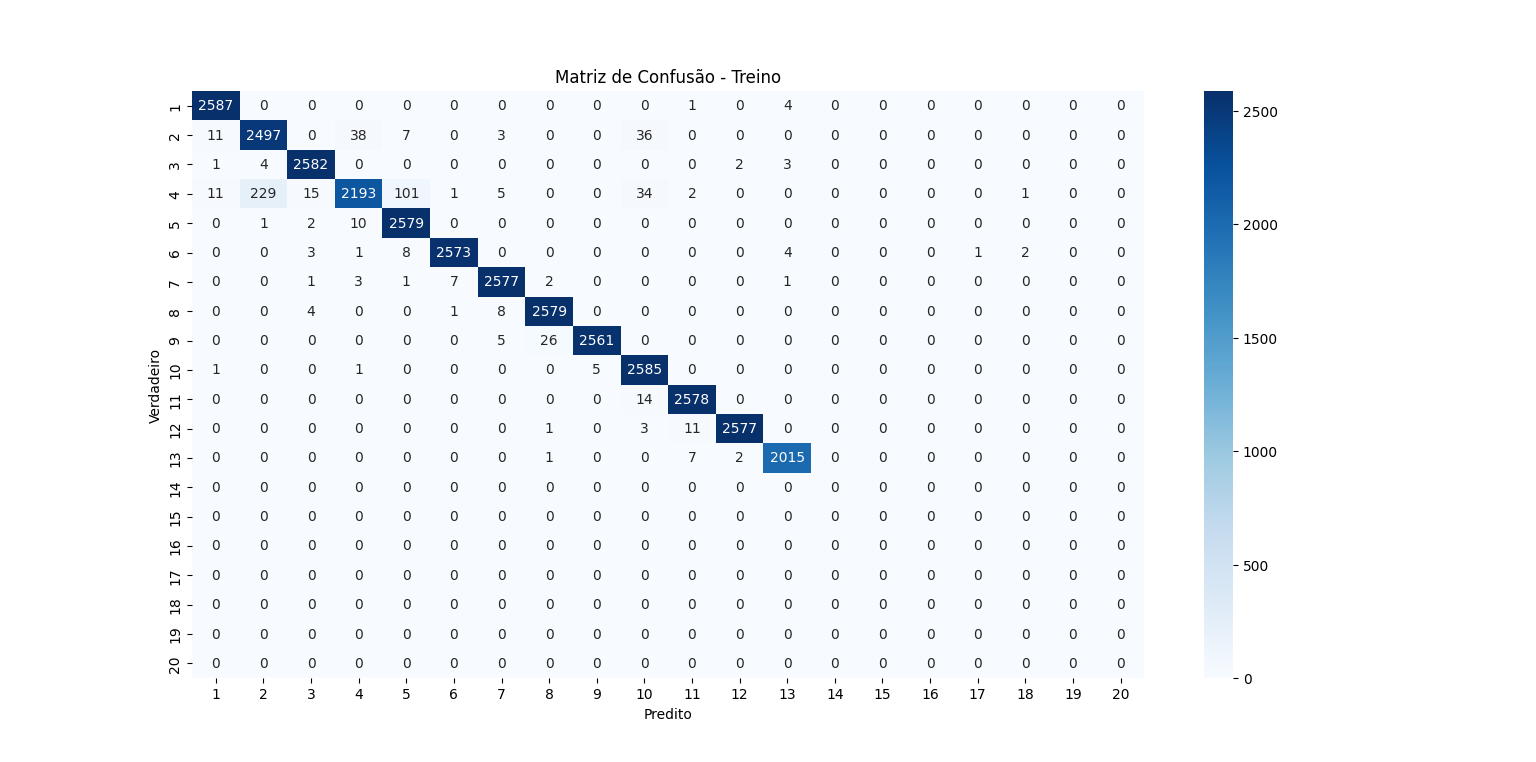
O modelo também foi ajustado para o caso de *early stopping*. Assim, os dados de treinamento foram subdivididos entre dados de treino e dados de validação, seguindo a lógica de embaralhamento dos dados para que dados de treino compusessem 80%, enquanto os de validação compusessem 20%. Além disso, foi determinado um limiar de checagem desses dados de validação, ou seja, a cada 5 épocas, como sugerido, é calculado o EQM de validação que, se não tiver apresentado melhora nas últimas 10 épocas, o treinamento seria interrompido pois isso significaria uma tendência de crescimento que poderia levar a *overfitting*.

Utilizando a topologia de 3 camadas ocultas com 100 neurônios cada e precisão de 0,001, a tendência de crescimento do EQM foi observada na 80ª época. Após interrompido o treinamento, a acurácia média foi calculada, dentre outros valores, como se pode ver no Quatro 5.

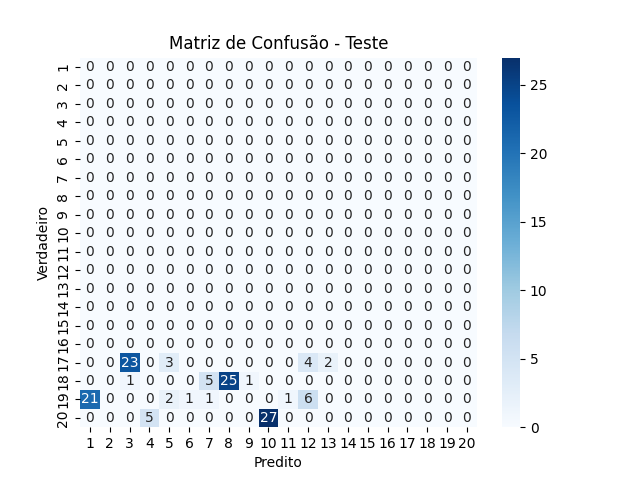
**Quadro 5** – Valores da acurácia de treino após *early stopping*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Medida | Média | Desvio padrão | Máximo | Mínimo |
| Acurácia | 0.965895 | 0.03380 | 1.00000 | 0.79178 |

Apesar da alta acurácia média na fase de treino, a interrupção na 80ª época fez com que o modelo tivesse 0,0% de acerto durante a fase de testes. As matrizes de confusão nas figuras 17 e 18 atestam essa discrepância.



1. Matriz de confusão da fase de treino do MLP após *early stopping*.



1. Matriz de confusão da fase de teste do MLP após *early stopping*.

Assim, chegou-se à conclusão de que o modelo, neste presente trabalho, não se beneficiou da técnica de *early stopping*, pois esta não só não preveniu o *overfitting*, quanto produziu resultados insatisfatórios na fase de testes. Em contrapartida, a topologia denominada ideal descrita anteriormente conseguiu atingir convergência e altas médias de acurácia para ambas as fases do modelo.

##### Referências

1. BARBOSA, Paulo Cirillo Souza. Redes Neurais Introdução. Fortaleza: Unifor, 2023. 297 slides, color.