Departamento de Informática

Aprendizagem Computacional 2023/2024

Mestrado em Engenharia Biomédica

Assignment 1
OCR - Optical Character Recognition

Grupo 3, Prática Laboratorial 1

Duarte Ferreira (2020235393)

Cristiana Azevedo (2020221121)

22.outubro.2023

1. Introdução

O *Machine Learning* é um método de análise de dados que dá a capacidade aos computadores de aprenderem sem programação explícita. Partindo deste conceito, este trabalho teve como principal objetivo desenvolver modelos de redes neuronais para o reconhecimento de caracteres manuscritos, mais concretamente os algarismos: {1,2,3,4,5,6,7,8,9,0}. Para isso, recorreu-se à tecnologia OCR - *Optical Character Recognition* - que desempenha um papel fundamental na conversão de texto manuscrito ou impresso para formato digital.

Diferentes arquiteturas de redes neurais foram desenvolvidas e avaliadas. Inicialmente, foram desenvolvidas arquiteturas que combinavam um filtro (*Associative Memory* ou B*inary Perceptron*) com um classificador de uma única camada. Posteriormente, foram construídas outras arquiteturas que utilizaram apenas classificadores com uma ou duas camadas. As funções de ativação consideradas foram: *hardlim* (binária), *purelin* (linear), *logsig* (sigmoide) e, ainda, *softmax* (sigmoide) em uma camada, permitindo uma comparação abrangente de desempenho. A escolha dessas arquiteturas, bem como das funções de ativação associadas, foi deixada ao critério do utilizador, proporcionando flexibilidade para explorar várias abordagens e técnicas na busca pelo desempenho ideal.

É relevante destacar que todas as etapas deste projeto foram implementadas utilizando a linguagem de programação MATLAB. Para o desenvolvimento das redes neurais, a *Neural Network Toolbox* e o *App Designer* foram ferramentas fundamentais. Além disso, o conjunto de dados utilizado foi criado por ambos os elementos do grupo e serviu como base para o treino e teste das redes neuronais desenvolvidas.

2. Dataset

Neste trabalho, o primeiro passo realizado foi a criação de um dataset para treinar e testar o classificador. Utilizou-se a função fornecida *mpaper.m*, que possibilitou a geração de 1000 caracteres de 0 a 9, desenhados manualmente por ambos os membros do grupo. Por meio dessa função, foi gerada uma grelha de 5x10 células. Em cada célula era desenhada um algarismo, obtendo-se 50 algarismos em cada matriz e 20 matrizes no total (matrizes P1-P20). Esta abordagem proporcionou uma diversidade de dados, tornando o conjunto de dados robusto. Cada célula de cada matriz representava um neurónio numa matriz de 16x16, associado a um valor de ativação entre 0 e 1, onde 1 representava um neurónio branco pertencente à formação do dígito e 0 representava um neurónio preto que não fazia parte da representação do número. De seguida, as 20 matrizes foram organizadas numa única matriz com dimensões de 256x1000 – *P_treino*.

O resultado de *P_treino* foi um dataset abrangente e diversificado, que permitiu aos classificadores aprenderem com exemplos mais variados e obter um desempenho sólido no reconhecimento de caracteres manuscritos.

Para evitar o *overfitting* dos dados, o conjunto foi dividido aleatoriamente, onde 85% dos caracteres destinou-se ao conjunto de treino e os restantes 15% ao conjunto de teste. A matriz *P_treino* foi utilizada para treinar os classificadores, enquanto duas matrizes de teste, *P_teste1* e *P_teste2* de 256x50, foram criadas para avaliar o desempenho dos modelos. As duas matrizes de teste foram criadas pelo mesmo elemento do grupo, mas de formas diferentes.

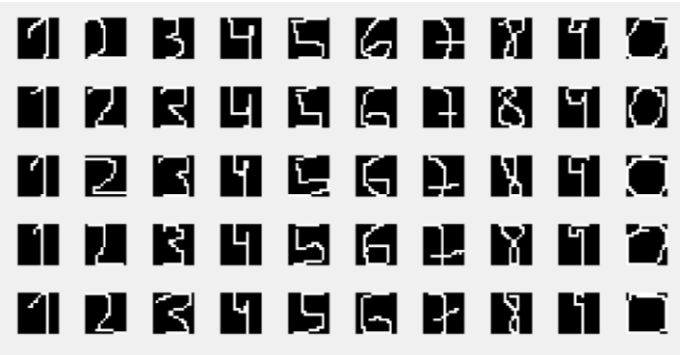

Figura 1. Matriz *P_teste1* representada pelo s*howim.m*

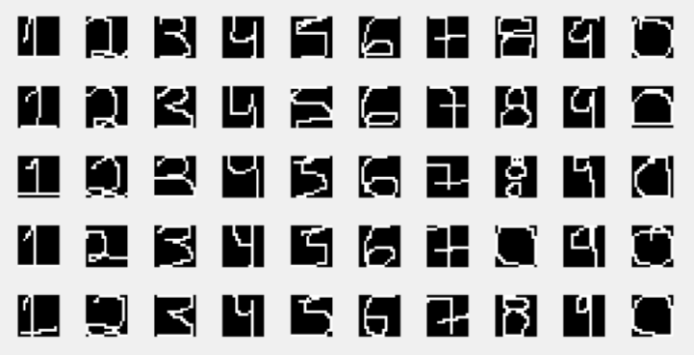

Figura 2. Matriz *P_teste2* representada pelo s*howim.m*

3. Arquitetura da Rede Neuronal

O objetivo deste trabalho é averiguar qual das seguintes arquiteturas de redes neuronais apresenta melhores resultados: classificador apenas, ou classificador + filtro.

3.1 Classificador

O classificador em questão pode possuir 1 camada, 2 camadas ou 1 camada com a função *softmax*. A função *softmax* é uma função de ativação frequentemente utilizada na camada de saída de uma rede neural, especialmente em problemas de classificação com várias classes (ou categorias). Esta função transforma um vetor de números reais em um vetor de probabilidades, onde cada elemento do vetor de saída representa a probabilidade de pertencer a uma determinada classe. A soma de todas as probabilidades resulta em 1, garantindo uma distribuição de probabilidade válida.

Para a classificação, foram também usadas diferentes funções de ativação: *hardlim* (binária), *purelin* (linear), *logsig* (sigmoide). As funções *purelin* e *logsig* não produzem resultados binários, ao contrário da função *hardlim.* Elas geram valores reais em intervalos específicos: *purelin* pode produzir qualquer valor real entre $[-\infty,+\infty][-\infty,+\infty]$ e *logsig* pode gerar valores no intervalo [0,1]. Quando desejamos obter uma saída binária, é necessário realizar um processo adicional para transformar esses valores contínuos em valores binários, através do método heurístico, onde o valor mais elevado é mapeado para 1 e os restantes para 0.

Quando é usado o classificador com duas camadas, a função binária (*hardlim*) não é utilizada uma vez que não uma função continua e por isso não é diferenciável. Neste classificador, a camada oculta e a camada de saída tinham 40 e 10 neurônios, respetivamente.

Para cada classificador foram utilizados os seguintes parâmetros;

- `Learning rate=0.5`
- `Maximum epochs epochs = 1000`
- `Goal= 1e-6;`
- `Criterion='sse'`

3.2 Filtro + Classificador

Neste caso, foram aplicados dois tipos de filtros, o *Associative Memory* e o *Binary Perceptron*. O propósito desses filtros é extrair características ou padrões relevantes dos dados de entrada, tornando-os mais adequados e compatíveis para a subsequente tarefa de classificação.

3.3 Treino

Após a definição da arquitetura desejada para a rede neural, o passo seguinte é treinar a rede de forma a alcançar os melhores resultados possíveis. O treino da rede envolve o ajuste dos parâmetros do modelo (*weights e bias)* para que o modelo possa realizar previsões precisas com base nos dados de treino. Esse ajuste é alcançado por meio da configuração de parâmetros derivados da comparação entre as saídas da rede e a matriz Target gerada a partir da função *PerfectArial.mat* (no caso do *perceptron*) ou das funções *T* de treino e teste, que representam matrizes diagonais.

Todas as redes foram armazenadas para permitir o acesso posterior por meio da interface criada.

4. Interface

A interface gráfica GUI (Graphic User Interface) do usuário, desenvolvida utilizando o App Designer do Matlab, foi criada para facilitar a interação com o usuário. Através dessa interface, o usuário pode selecionar a opção de aplicar um filtro e escolher o tipo específico desejado, assim como definir o número de camadas e as funções de ativação pretendidas. No caso de optar por uma camada, o usuário pode ainda decidir se quer utilizar a função de ativação *softmax*. Após fazer suas escolhas, o usuário pode treinar o modelo e avaliar seu desempenho. Além disso, há a possibilidade de testar o modelo desenhando e testando novos caracteres usando a função *mpaper.m*.

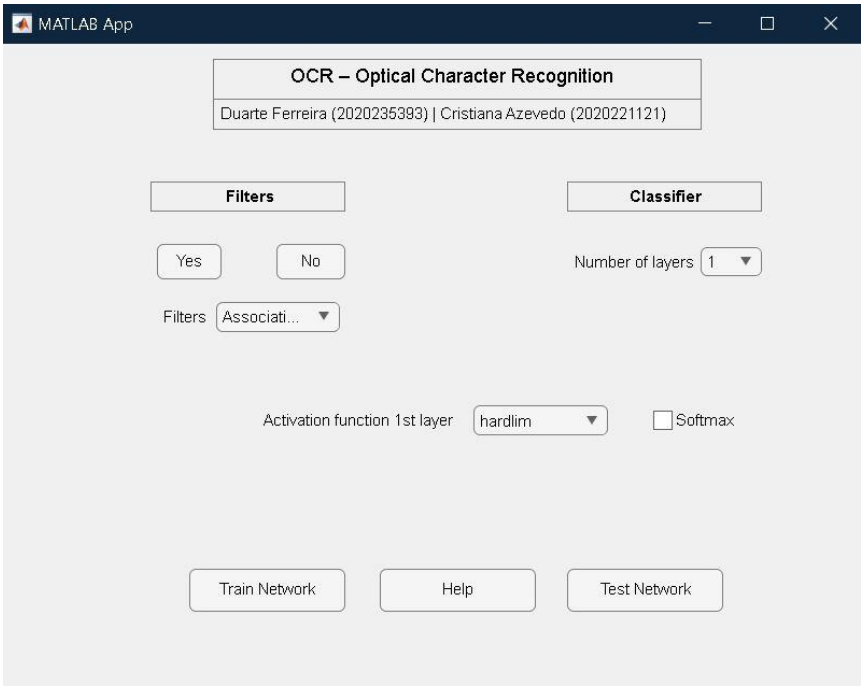

Figura 3. Interface GUI

5. Matlab Scripts e Variáveis

Variáveis	Descrição
P_treino	Matriz de 256x1000 que contém o conjunto de dados de treino
P_teste1	Matriz de 256x50 que contém o conjunto de dados de teste escritos por um elemento do grupo
P_teste2	Matriz de 256x50 que contém o conjunto de dados de teste escritos por um elemento do grupo de forma diferente
T	Matriz 10x1000 que contém as classificações corretas para o conjunto de dados de treino
T_teste1 e T_teste2	Matrizes 10x50 que contém as classificações corretas para o conjunto de dados de teste 1 e 2
Perfect_treino	Matriz de 256x1000 que contém os números de *PerfectArial.mat*
assoc_memory.mat	Matriz de weight 256x256 do filtro Associative Memory
perceptron_net.mat	Filtro Binary Perceptron
associative_*_trained_net.mat	Classificadores de uma camada com filtro Associative Memory
perceptron_*_trained_net.mat	Classificadores de uma camada com filtro Binary Perceptron
1L_*_trained_net.mat	Classificadores de uma camada sem filtro.
1LSft_*_trained_net.mat	Classificadores de uma camada com a função de ativação *softmax* e sem filtro.
2L_*_trained_net.mat	Classificadores de duas camadas sem filtro

Tabela 1. Variáveis utilizadas e a sua respetiva descrição

Matlab Script	Descrição
interface.mlapp	Interface que permite a interação com o utilizador, oferecendo a possibilidade de treinar e testar os modelos de redes neuronais
myclassify.m	Função chamada pelo *ocr_fun* que utiliza um modelo de rede neural já treinado para classificar os números sorteados pelo utilizador
dataset.mlx	Função utilizada para criar o dataset. Criar a matriz *P_treino* com todos os dados de treino, a matriz *Perfect_treino* a partir da *PerfectArial.mat*, as matrizes de teste *P_teste1* e *P_teste2* e os targets de treino T e de teste *T_teste1*, *T_teste2*
train_classifier.mlx	Função que a interface chama para treinar e testar uma rede neural ou para permitir que o utilizador desenhe novos caracteres e os teste. Contém o algoritmo de formação dos diferentes modelos de redes neuronais
filters.m	Função que é utilizada para aplicar diferentes filtros a um conjunto de dados. Os resultados são salvos em arquivos para uso posterior
heuristic.m	Função de pós-processamento que converte as saídas de uma rede neural para um formato onde apenas o maior valor em cada coluna é definido como 1, e todos os outros valores na mesma coluna são definidos como 0
mpaper.m	Função fornecida que permite o desenho dos dígitos manualmente e a sua gravação em ficheiros *.mat*
ocr fun.m	Função fornecida que permite mostrar os resultados da classificação na grelha
showim.m	Função fornecida utilizada para mostrar um número arbitrário de algarismos dados como inputs

Tabela 2. Matlab Scripts construídos/fornecidos e a sua respetiva descrição

6. Resultados

Os resultados da precisão para o grupo de treino e os dois grupos de teste foram obtidos ao treinar e testar todas as possíveis arquiteturas e funções de ativação. Esses resultados estão apresentados nas tabelas a seguir para todas as combinações de arquitetura e funções de ativação. Estes resultados foram dados por meio das matrizes de confusão, que mostra o desempenho do modelo ao comparar as classificações feitas pelo modelo com as verdadeiras classes dos dados.

6.1 Binary Perceptron + Classifier (1 layer)

	Hardlim	**Purelin**	**Logsig**
Treino	100%	100%	80%
Teste 1	82%	56%	66%
Teste 2	66%	34%	46%

Tabela 3. Resultados da combinação do filtro "Binary Perceptron" com as três funções de ativação (hardlim, purelin e logsig)

6.2 Associative memory + Classifier (1 layer)

	Hardlim	**Purelin**	**Logsig**
Treino	97,8%	98%	98,9%
Teste 1	90%	94%	94%
Teste 2	68%	76%	80%

Tabela 4. Resultados da combinação do filtro "Associative Memory" com as três funções de ativação (hardlim, purelin e logsig)

6.3 Classifier (1 layer)

	Hardlim	Purelin	Logsig
Treino	95,6%	95,8%	96,9%
Teste 1	94%	90%	96%
Teste 2	68%	76%	72%

Tabela 5. Resultados do classificador com uma camada

6.4 Classifier (1 layer) +softmax

	Hardlim	Purelin	Logsig
Treino	95,6%	95,8%	96,9%
Teste 1	94%	90%	96%
Teste 2	68%	76%	72%

Tabela 6. Resultados do classificador de uma camada com a função softmax associada

6.5 Classifier (2 layer)

	Purelin+Purelin	Purelin+Logsig	Logsig+Purelin	Logsig+Logsig
Treino	95,7%	96,1%	95,3%	94%
Teste 1	90%	96%	94%	86%
Teste 2	76%	72%	64%	60%

Tabela 7. Resultados do classificador com duas camadas

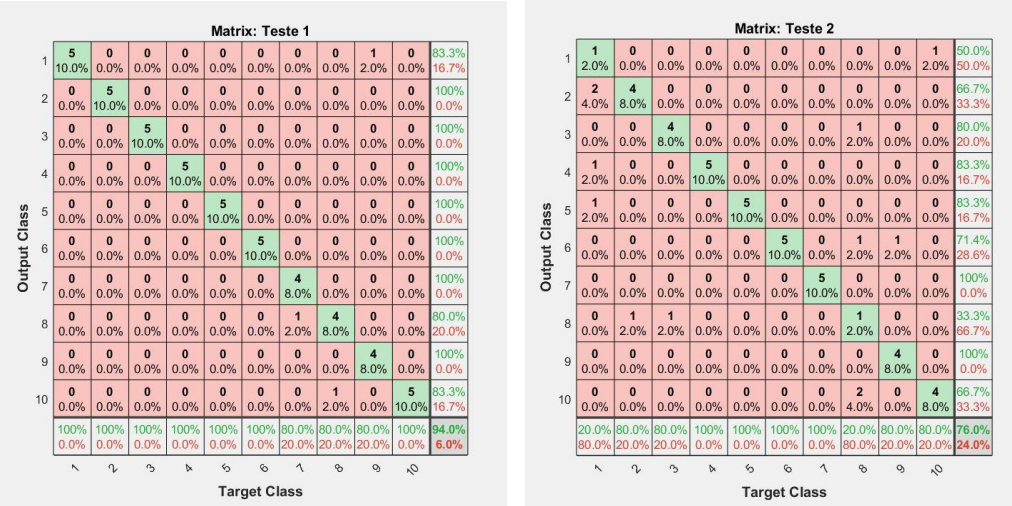

Figura 4. Exemplos de matrizes confusão obtidas

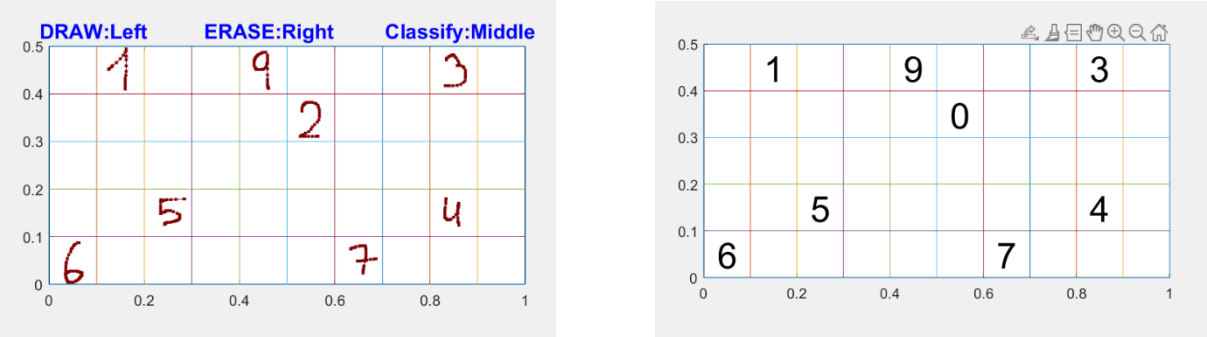

Figura 5. Grelhas obtidas através da *interface.mlapp*. Na esquerda a grelha de entrada com os algarismos escritos, na direita a grelha dos resultados com 8/9 algarismos bem classificados.

7. Discussão e Conclusão

Esse trabalho representa uma abordagem prática e interdisciplinar, que envolve conceitos de processamento de imagens, aprendizado de máquina e reconhecimento de padrões. O uso de redes neurais oferece uma solução flexível e adaptável para o desafio do OCR, com potencial para contribuir significativamente para o desenvolvimento de sistemas de reconhecimento de caracteres mais precisos e eficazes.

As diferenças nos diferentes tipos de redes neuronais, surgem associados ao diferente desempenho que cada arquitetura apresentou, tal como surge nos resultados apresentados anteriormente

Com base na análise desses resultados, observa-se que o desempenho alcançado é bastante satisfatório, com a maioria das classificações apresentando uma precisão acima de 80%. É notável que os resultados no conjunto de teste são superiores aos resultados nos conjuntos de teste, tal como era esperado.

No que toca às arquiteturas que apresentam filtros, verifica-se que a arquitetura com o filtro *"Associative Memory"* apresenta melhores resultados que a arquitetura com o filtro *"Binary Perceptron"*. Na verdade, os piores resultados são observados na arquitetura *"Binary Perceptron + Classifier (1 layer)"*, especialmente nos testes 1 e 2, ao utilizar as funções de ativação *"purelin"* e *"logsig",* contrastando com a arquitetura com o filtro *"Associative Memory"*.

Uma conclusão importante é que a aplicação do filtro *"Binary Perceptron"* parece deteriorar os resultados. Por exemplo, nos testes 1 e 2, utilizando a função de ativação *"purelin*, obteve-se uma precisão de apenas 56% e 34%, respetivamente. Uma possível razão para essa redução de desempenho pode ser o fato de que os números são filtrados para se assemelharem mais à fonte Arial. Isso pode tornar mais difícil para os caracteres desenhados de maneira muito distinta se ajustarem a essa fonte específica.

Ao comparar as arquiteturas, observamos que, no caso das arquiteturas sem filtros, os resultados das diversas arquiteturas são bastante consistentes e próximos entre si. No entanto, o mesmo não se verifica para as arquiteturas com filtros, onde há uma notável disparidade de desempenho entre as arquiteturas constituintes.

Ao analisar a arquitetura *"Classifier (2 layer),"* observamos que, em geral, os resultados são bastante similares à arquitetura *"Classifier (1layer)",* embora ligeiramente inferiores em alguns casos, como na combinação de funções de ativação *"logsig+logsig,"* que alcança uma precisão de 86% para o teste 1, um desempenho que não é visto na arquitetura *"Classifier (1layer)"* para o mesmo teste. Esses resultados sugerem que a inclusão de uma camada adicional na arquitetura não conduziu a melhorias substanciais na precisão da classificação.

Na arquitetura *"Classifier (1 layer)"* é curioso notar que os valores obtidos são idênticos aos alcançados na arquitetura *"Classifier (1 layer) + softmax"* o que sugere que a função softmax não teve nenhum impacto discernível nos resultados.

A comparação entre o teste 1 e o teste 2 revela que o teste 1 apresenta melhores resultados do que o teste 2. Uma possível explicação para isso pode ser o facto de que o teste 1 foi conduzido por um membro interno do grupo, enquanto o teste 2 foi realizado por uma pessoa externa ao grupo.

Apesar de terem sido realizadas 1000 iterações para reduzir a discrepância, os resultados sugerem que podem ser necessários mais dados para nivelar o desempenho entre o Teste 1 e o Teste 2 para arquitetura com filtro.

Da análise dos dados, é possível ainda concluir que, de forma geral, a arquitetura que utiliza a função de ativação *"logsig"* apresenta melhores resultados em termos de precisão. Isso é evidenciado pelos desempenhos obtidos em várias arquiteturas, onde a função de ativação *"logsig"* está presente. É importante ressaltar que a escolha da função de ativação pode ter um impacto significativo no desempenho de uma rede neural em uma determinada tarefa.

Portanto, é possível afirmar com confiança que todos os objetivos inicialmente estabelecidos foram plenamente alcançados, já que foi possível identificar não apenas uma, mas múltiplas arquiteturas que demonstraram resultados consistentemente aceitáveis e promissores. Este êxito representa não apenas uma validação da metodologia adotada, mas também oferece uma base sólida para futuras investigações e avanços neste campo. A diversidade de abordagens bem-sucedidas encontradas reforça a viabilidade e a eficácia deste trabalho.

8. Bibliografia

[1]. Dourado, A. 2023. Assignment 1 – Optical Character Recognition – Gui˜ao do trabalho.

[2]. Dourado, A., 2023. Chapter 4 - Neurons Networks - Slides das aulas de Computa¸c˜ao Neuronal e Sistemas Difusos.

[3]. Beale, M., Hagan, M. and Demuth, H., 2022. Deep Learning Toolbox - Getting Started Guide. MathWorks. Available at: https://www.mathworks.com/help/pdf doc/deeplearning/nnetgs.pdf.

[4]. Beale, M., Hagan, M. and Demuth, H., 2022. Deep Learning Toolbox - User's Guide. MathWorks. Available at: https://www.mathworks.com/help/pdf doc/deeplearning/nnet ug.pdf.