

***Aprendizagem Computacional***

***Assignment 1***

***OCR – Optical Character Recognition***

**Professor: António Dourado Pereira Correia**

**Grupo: 1**

**Diogo Alexandre Santos Rosário nº 2023185395**

**Arthur Francisco Navickas Itacarambi nº2020115569**

1. **Introdução**

Este relatório é o resultado de um trabalho proposto no âmbito da disciplina de Aprendizagem Computacional.

Apresentaremos neste relatório uma visão periférica do problema em estudo *OCR - Optical Character Recognition*. Utilizando ferramentas de machine learning, disponibilizadas pelo matlab (aplicação externa), iremos tentar formular modelos de redes neuronais que consigam reconhecer os algarismos de 0-9 escritos a mão.

Nos próximos capítulos será discutido o passo-a-passo que os autores seguiram para chegar aos presentes resultados, o que esses resultados representam e o que podemos concluir sobre eles.

1. ***Dataset’s***

O *dataset* e treino foram criados utilizando a aplicação *mpaper* como instruído:

*“A) to generate, from a grid five-by-ten opened when it is runed, where the user draws manualy digits 0 to 9. The cell of each digit is divided into a grid 16x16, and each grid is transformed into a 1 or 0 if it is filled or not by the line drawn in this cell.”*

Para o data set foi desenhado, pelos autores deste relatório, 34 tabelas de 50 números. O resultado disso foi uma matriz com 1700 números e cada um possuindo 256 atributos. Essa mesma matriz foi usada para treinar todos os modelos apresentados neste relatório.

Para a *data* do teste foram desenhados outras 3 tabelas de 50 números, dessa vez, porem, foram usados individualmente em conjunto com uma tabela “.txt” (de mesmo valor) para que pudéssemos testar diferentes modelos que estão avaliando um mesmo caso.

Para criar o target do data set usamos um programa autoral “createTarget.m”, este programa gerou o arquivo “Target.mat” que também foi usado em todos os modelos apresentados neste relatório. O target consiste em um vetor com 10 posições, onde, todas as posições possuem o valor de 0, menos uma, a que possui o valor 1 e representa o valor correto (e cada vetor representa 1 único número apenas). Ou seja, o arquivo “Target.mat” eh uma matriz de 10 linhas com 1700 colunas.

No início do trabalho foi nos disponibilizado uma matriz “PerfectArial.mat” utilizada para treinar os modelos de filtro (Perceptron, Associative Memory). Essa matriz, da mesma forma que as feitas no data set e no data treino possui 256 atributos. Essa matriz, entretanto, possui informação de como seria cada um dos algarismos de 1 a 10 feitos perfeitamente. Essa matriz também teve de ser adaptada para possuir 1700 números ao invés de 10.

O leitor poderá observar os resultados dos testes e dos modelos na terceira secção deste relatório.

1. **Treino**

Parâmetros utilizados no treino dos classificadores

* Número máximo de epochs = 1000;
* Goal = 1e-6;
* Performance = 1e-6;
* Validation checks = 6;

Nota: No caso dos modelos que contém duas camadas, foi definido o uso de 10 neurónios na camada escondida *( Hidden layer)*

1. **Resultados**

Neste capítulo, apresentamos os resultados obtidos a partir dos modelos treinados, destacando as funções de ativações utilizadas em cada um. Foram realizados quatro tipos de testes distintos, sendo estes: Teste A (256x50), Teste B(256x50), Teste C (256x50) e o conjunto de dados que foi usado para treinar todos os modelos, P(256x1700).

Nas próximas figuras conseguimos ver o Teste A, B e C.

A number in squares with white squares

Description automatically generated with medium confidence A number in a row

Description automatically generated with medium confidence

***Figura X – Representação do teste A Figura X – Representação do teste B***

A number in a square

Description automatically generated with medium confidence

***Figura X – Representação do teste C***

**3.1 Memória Associativa + Classificador de uma camada**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Purelin** | **Hardlim** | **Logsig** |
| Treino | 21.24% | 19.53% | 27.18% |
| Teste A | 18.00% | 16.00% | 28.00% |
| Teste B | 14.00% | 18.00% | 28.00% |
| Teste C | 20.00% | 18.00% | 32.00% |
| **Média dos testes (A, B, C)** | 17.33% | 17.33% | **29.33%** |

Tabela X:Resultados obtidos da percentagem de acerto com filtro de memória associativa e um classificador de uma camada

**3.2 Perceptron + Classificador de uma camada**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Purelin** | **Hardlim** | **Logsig** |
| Treino | 47.35% | 49.94% | 49.29% |
| Teste A | 42.00% | 36.00% | 42.00% |
| Teste B | 32.00% | 36.00% | 48.00% |
| Teste C | 42.00% | 40.00% | 46.00% |
| **Média dos testes (A, B, C)** | 38.67% | 37.33% | **45.33%** |

Tabela X:Resultados obtidos da percentagem de acerto com filtro de *perceptron* e um classificador de uma camada

**3.3 Classificador de uma camada**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Purelin** | **Hardlim** | **Logsig** |
| Treino | 84.82% | 90.00% | 86.53% |
| Teste A | 80.00% | 86.00% | 82.00% |
| Teste B | 88.00% | 78.00% | 82.00% |
| Teste C | 92.00% | 90.00% | 90.00% |
| **Média dos testes (A, B, C)** | **86.67%** | 84.67% | 84.67% |

Tabela X:Resultados obtidos da percentagem de acerto com classificadores de uma camada.

**3.4 Classificador + softmax**

|  |  |  |
| --- | --- | --- |
|  | **Purelin** | **Logsig** |
| Treino | 87.82% | 84.24% |
| Teste A | 90.00% | 86.00% |
| Teste B | 90.00% | 78.00% |
| Teste C | 88.00% | 86.00% |
| **Média dos testes (A, B, C)** | **89.33%** | 83.33% |

Tabela X:Resultados obtidos da percentagem de acerto com classificadores de duas camadas, sendo o segundo sempre o “*softmax*”.

**3.5 Classificador com 2 camadas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Purelin e Purelin** | **Purelin e Logsig** | **Logsig e Purelin** | **Logsig e Logsig** |
| Treino | 86.12% | 86.94% | 87.00% | 85.24% |
| Teste A | 84.00% | 80.00% | 86.00% | 74.00% |
| Teste B | 88.00% | 84.00% | 88.00% | 72.00% |
| Teste C | 88.00% | 88.00% | 88.00% | 82.00% |
| **Média dos testes (A, B, C)** | 86.67% | 84.00% | **87.33%** | 76.00% |

Tabela X:Resultados obtidos da percentagem de acerto com classificadores de duas camadas.

A screenshot of a math game

Description automatically generated A screenshot of a game

Description automatically generated

Figuras X e X: Exemplo da classificação com o modelo Hardlim de uma camada.

1. **Guia do Projeto**

Na pasta que contém este relatório, existem outros ficheiros e pastas que possuem componentes importantes na realização do projeto.

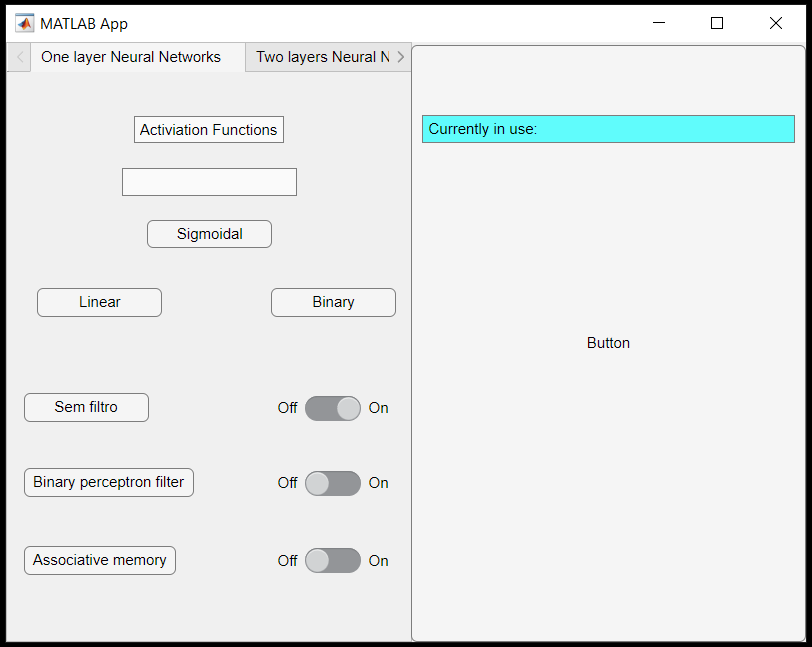
|  |  |
| --- | --- |
| **Função** | **Descrição** |
| app\_OCR.mlapp |  |
| createTarget.m |  |
| mpaper.m |  |
| myclassify.m |  |
| ocr\_fun.m |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Pasta** | **Descrição** |
| *data* | Contém datasets (treino e de testes). |
| *Models* | Contém os modelos já treinados (1 camada com ou sem filtro e, modelos de duas camadas). |
| *myClassifyArguments* | Pasta auxiliar da aplicação |
| *Utils* | Contém as funções que foram usadas para criar os modelos. |

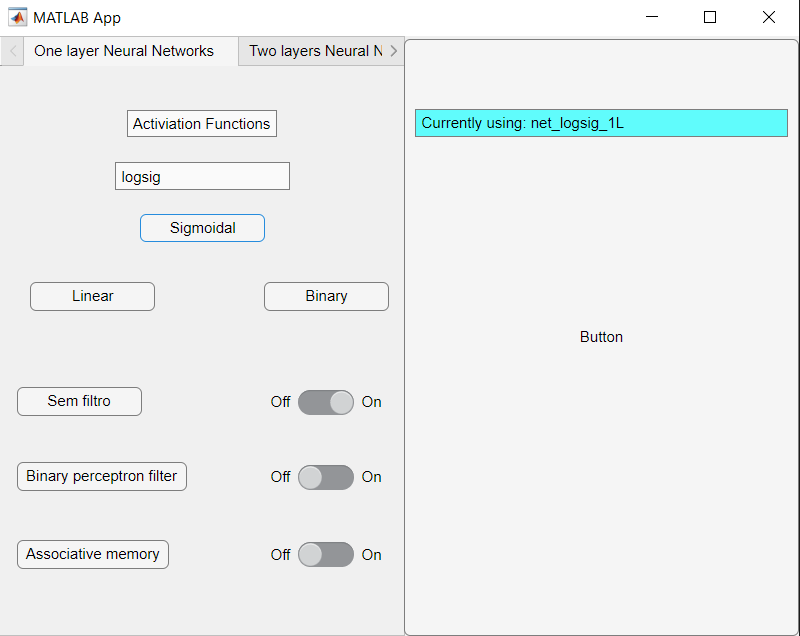
Tabela 5.2 – Tabela de pastas do projeto

1. **Guia de Utilização da Aplicação**

Para utilizar a aplicação vá na pasta disponibilizada por nosso grupo e abra o programa “app\_OCR”. Ao abrir este programa a seguinte janela deve abrir:



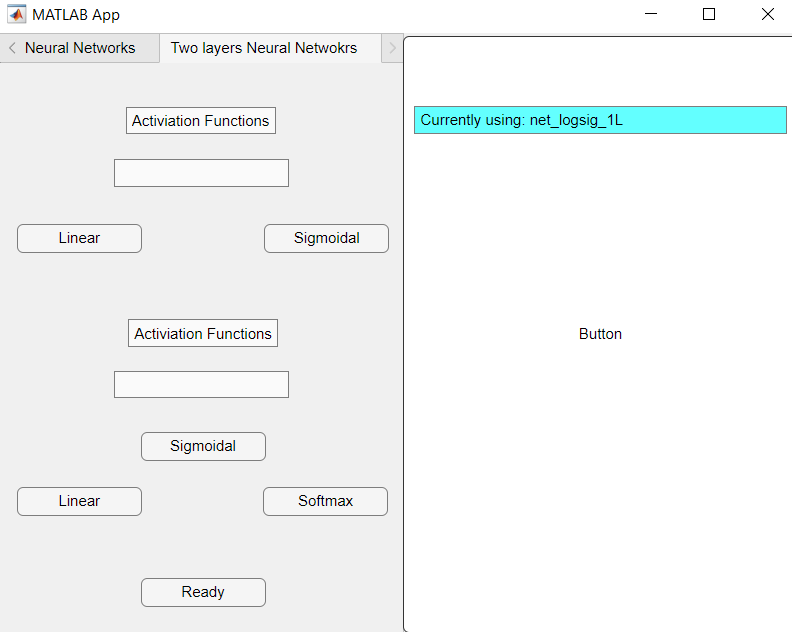
Nesta primeira aba chamada “One layer Neural Networks”, o utilizador pode selecionar qual função de ativação e o filtro que gostaria de utilizar (note que os modelos foram treinados previamente e a aplicação foi feita apenas para testá-los). Ao selecionar uma das funções de ativação as duas caixas de textos presente na imagem devem mudar, como tal:



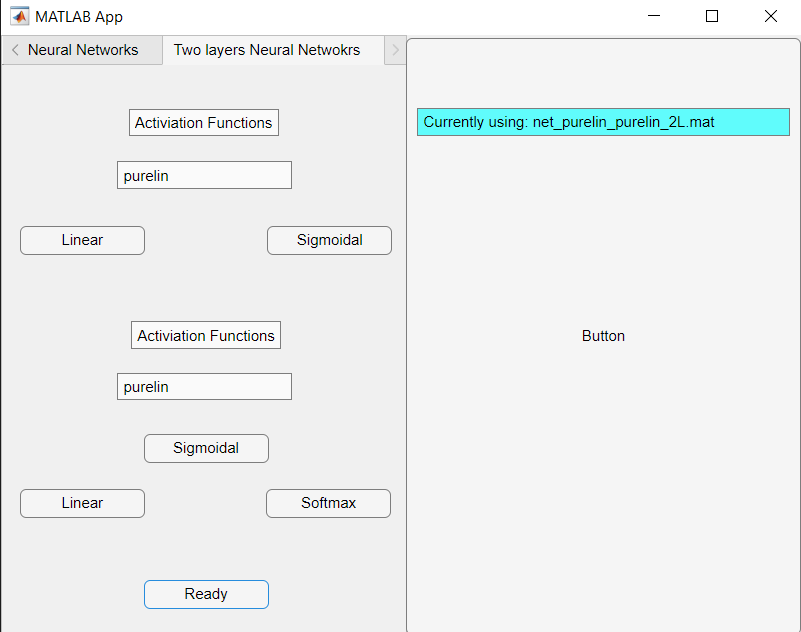
Como pode-se ver pela imagem a função sem filtro esta ligada, para utilizar outra basta ligar um dos outros dois interruptores.

Apos selecionar a função de ativação e o filtro basta clicar no “Button”, o que fazer apos isso será explicado na secção “Mpaper”.

Na segunda aba o utilizador possui a seguinte visualização:



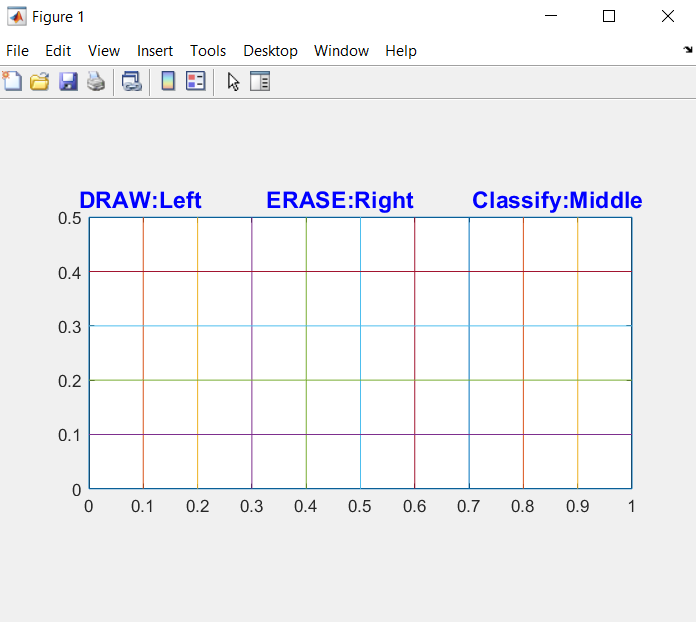
Neste caso deve-se selecionar duas funções de ativação e clicar no botão “Ready”. Isso resultara em algo parecido a:



Note que em ambos o caso a caixa de texto azul possui o nome do modelo a ser utilizado.

**Mpaper:**

Apos clicar em “Button” a seguinte janela deve aparecer para o utilizador:



Neste caso o utilizador deve desenhar pelo menos um número e no máximo 50 números, um em cada um dos pequenos quadrados. Para tal basta utilizar o botão esquerdo do rato. O utilizador pode também apagar um dos números desenhados com o botão direito do rato (note que ao utilizar o botão direto do rato todo o desenho feito em um único quadrado eh inteiramente apagado).

Apos ter feito os desenhos que deseja o utilizador precisara apenas pressionar o botão do meio do mouse para utilizar o modelo selecionado na app. Apos pressionado duas janelas irão aparecer, a primeira com os números que o modelo adivinhou a segunda com uma representação do modelo selecionado.

Caso queria testar outro modelo não eh necessário seguir todo o procedimento de novo. Para utilizar outro modelo volte a aplicação, selecione o modelo que gostaria de utilizar, volte a onde foi desenhado os números e pressione o botão de meio novamente.

**Nota 1:** Caso não possua o botão do meio do rato basta pressionar Shift + botão esquerdo do rato.

**Nota 2:** Não eh possível utilizar filtros na rede neuronal de duas camadas, a aplicação tem proteções contra tal acontecimento.

**Nota 3:** A aplicação não possui suporte para tela cheia, sendo seu uso eh contraindicado.

1. **Conclusão**