Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática
Conhecimento e Raciocínio

Leandro Adão Fidalgo | a2017017144
Pedro dos Santos Alves | a2019112789

Laboratório P4
Trabalho Prático 1

Coimbra, 27 de junho de 2021

Índice

1. Introdução

O presente relatório descreve o projeto desenvolvido pelos alunos: Leandro Fidalgo e Pedro Alves, no âmbito da disciplina de Conhecimento e Raciocínio da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

Neste caso foi escolhido o tema 1 sobre Redes Neuronais e pretende-se aprofundar os conhecimentos sobre redes neuronais. O objetivo consiste na implementação e teste de diferentes arquiteturas de redes neuronais feedforward para classificar corretamente 10 carateres gregos: alpha, beta, gamma, epsilon, eta, theta, pi, rho, psi e omega.

O objetivo do presente trabalho é consolidar todos os conhecimentos adquiridos nas aulas teóricas e práticas ao longo de todo o semestre.

2. Tratamento das imagens

Como as imagens se encontram com uma dimensão de 3024x3024 foi necessário o seu redimensionamento, para que não haja tempos excessivos nos treinos e testes das redes neuronais. Como é possível ver na imagem abaixo todas as imagens foram redimensionadas para uma dimensão de 21x21.

```
resolution = 21;
imageInput = zeros(resolution * resolution, numFiles);
imageTarget = zeros(numImageTypes, numFiles);
for i = 1 : numFiles
    image = imread(strcat(files(i).folder, '\', fileNames2(i)));
    if (size(image, 3) > 1)
        image = rgb2gray(image);
    end
    image = imresize(image, [resolution resolution]);
    imageInput(:,i) = image(:);
    j = floor((i - 1) / numFilesPerType) + 1;
    imageTarget(j, i) = 1;
end
```

3. Alínea a

Começou-se por utilizar uma rede neuronal de uma camada com 10 neurónios e treinou-se a rede utilizando todos os caracteres da Pasta_1. De seguida forma testadas outras arquiteturas (topologias), funções de ativação e de treino e no final foram registados os valores de desempenho das diferentes parametrizações num ficheiro excel. Comparando os resultados obtidos verificou-se que o número e a dimensão das camadas escondidas influenciaram o desempenho. Como foi possível constatar no estudo feito ao aumentar o número de camadas escondidas o tempo de processamento aumenta. Foi possível ainda verificar que ao aumentar o número de neurónios pode influenciar positivamente no tempo de processamento porém o aumento dos neurónios não pode ser excessivo, sendo que 10 neurónios talvez seja o melhor tanto para o desempenho como para a precisão global do problema. É ainda possível notar que apenas com 5 neurónios o processamento torna-se bem mais lento e com piores resultados de precisão global.

As funções de treino influenciam o desempenho da rede neuronal, sendo que durante os teste foi possível constatar que a função de treino “trainbfg” foi das piores tanto a nível de tempo de processamento como a nível de precisão global.

Durante a realização dos testes nas funções de ativação, é possível verificar que não influenciam muito no tempo de processamento porém a precisão global é bastante influenciada, sendo que a pior de todas foi a “purelin, logsig” e a segunda pior “purelin, tansig”. Segundo os testes realizados a pior função de ativação a nível de tempo de processamento foi a “logsig” pois apenas apresentou melhores resultados quando combinada com a função “tansig”.

4. Alínea b

Usando o modelo base implementado anteriormente foram feitas as alterações necessárias para implementar e testar várias topologias e parametrizações de Redes Neuronais de forma a obter um bom desempenho para a classificação dos caracteres fornecidos na pasta Pasta_2.

Foi criada uma configuração inicial contendo um conjunto de camadas escondidas e neurónios, um conjunto de funções de ativação para a camada de saída, um conjunto de funções de treino e ainda um conjunto de divisões de exemplos. O processo de treino e simulação da rede foi automatizado criando as várias combinações de redes neuronais com as diversas configurações iniciais. Abaixo estão demonstradas as diversas configurações iniciais gerando 144 combinações distintas, sendo que estas estão contidas no ficheiro "estudoNN-alinea-b.xlsx".

```
layers = [{[10]},...
    {[20]},...
    {[5 5]},...
    {[10 10]}];

transferFcns = [{'logsig'},...
    {'purelin'},...
    {'tansig'}];

%'trainbfg',...
trainFcns = [{'traingdx'},...
    {'trainlm'}];

divideFcns = [{'dividerand'},...
    {'divideint'}];

divideParams = [{[0.7 0.15 0.15]},...
    {[0.8 0.1 0.1]},...
    {[0.6 0.2 0.2]}];
```

Ao longo de todo este processo automático foram gravadas as melhores redes em ficheiros ".mat", também foi guardada a melhor das melhores redes, sendo esta uma rede neuronal com 1 camada escondida com 20 neurónios, com funções de ativação "tansig, tansig", com a função de treino "trainlm" e com uma divisão de exemplos "divideint ={0.8, 0.1, 0.1}".

Nos vários testes realizados com as diversas redes neuronais que utilizaram a Pasta2 para aprender, foi possível constatar que a divisão dos exemplos com o "divideint" aparenta obter melhores resultados que o "dividerand", pois o "divideint" divide os dados usando uma seleção intercalada e o "dividerand" utiliza uma seleção aleatória o que o torna pouco eficiente para o problema em questão. Ao logo dos testes realizados também foi possível constatar que a função de treino "trainlm" obteve ligeiramente melhores resultados que a "traingdx". Nas funções de ativação foi possível notar que para este problema a função "tansig" para a saída foi a que obteve melhores resultados, sendo que a função "logsig" demonstrou resultados piores e a "purelin" demonstrou resultados bastante próximos da "tansig". Todos estes testes que foram efetuados, foram realizados com a função de ativação "tansig" para as camadas escondidas.

5. Alínea c

Como é possível verificar na imagem abaixo, sendo esta a melhor rede neuronal encontrada na alínea b, sem treinar a rede neuronal a precisão total é de 85%, após treinar com as imagens da Pasta3 não conseguiu obter 100% de precisão total na Pasta3, porém conseguiu 100% nas restantes, após treinar com todas as pastas foi possível verificar que conseguiu obter uma precisão total de 100% em todas as pastas (rede neuronal com 1 camada escondida de 20 neurónios, com função de ativação "tansig", com função de ativação de saída "tansig", função de treino "trainlm", com divisão de exemplos "divideint = {0.8, 0.1, 0.1}").

```
Carregada a rede neuronal com 100.000000 de precisão total e 100.000000 de precisão de teste.

Precisão total: 85.000000

Treinar a rede com imagens da Pasta_3

Precisão total Pasta_1: 100.000000

Precisão total Pasta_2: 100.000000

Precisão total Pasta_3: 85.000000

Treinar a rede com imagens da (Pastal + Pasta_2 + Pasta_3)

Precisão total Pasta_1: 100.000000

Precisão total Pasta_2: 100.000000

Precisão total Pasta_3: 100.000000
```

Para além da utilização da melhor rede neuronal da alínea b também foram feitos testes com outras redes que demonstraram bons resultados, sendo estas também da alínea b.

```
Carregada a rede neuronal com 99.000000 de precisão total e 100.000000 de precisão de teste.

Precisão total: 82.500000

Treinar a rede com imagens da Pasta_3

Precisão total Pasta_1: 100.000000

Precisão total Pasta_2: 99.000000

Precisão total Pasta_3: 82.500000

Treinar a rede com imagens da (Pastal + Pasta_2 + Pasta_3)

Precisão total Pasta_1: 100.000000

Precisão total Pasta_2: 99.000000

Precisão total Pasta_3: 82.500000
```

Como é possível notar na imagem acima, um teste com uma precisão total de 99% e com uma precisão de teste de 100%, a rede neuronal sem testes tem 82.5% de precisão total e apenas consegue 100% na Pasta1 tanto treinando com as imagens da Pasta3 ou com todas as pastas (rede neuronal com 1 camada escondida de 10 neurónios, com função de ativação "tansig", com função de ativação de saída "tansig", função de treino "traingdx", com divisão de exemplos "divideint = {0.8, 0.1, 0.1}").

```
Carregada a rede neuronal com 39.000000 de precisão total e 30.000000 de precisão de teste.

Precisão total: 35.000000

Treinar a rede com imagens da Pasta_3

Precisão total Pasta_1: 40.000000

Precisão total Pasta_2: 55.000000

Precisão total Pasta_3: 55.000000

Treinar a rede com imagens da (Pastal + Pasta_2 + Pasta_3)

Precisão total Pasta_1: 90.000000

Precisão total Pasta_2: 90.000000

Precisão total Pasta_3: 87.500000
```

Para finalizar os testes, como é possível ver acima, foi utilizada uma rede neuronal com valores mais baixos de precisão tanto total como de teste e os resultados até foram surpreendentes, pois a rede após treinar com as imagens de todas as pastas foi possível notar um aumento incrível na sua precisão total (rede neuronal com 1 camada escondida de 10 neurónios, com função de ativação "tansig", com função de ativação de saída "tansig", função de treino "traingdx", com divisão de exemplos "dividerand = {0.8, 0.1, 0.1}").

6. Alínea d

Após ambos os membros do grupo desenharem os caracteres gregos, foram testados na melhor rede neuronal da alínea c e foi obtida uma precisão total de 30%, tendo esta rede neuronal uma precisão total e de teste de 100% (rede neuronal com 1 camada escondida de 20 neurónios, com função de ativação “tansig”, com função de ativação de saída “tansig”, função de treino “trainlm”, com divisão de exemplos “divideint = {0.8, 0.1, 0.1}”).

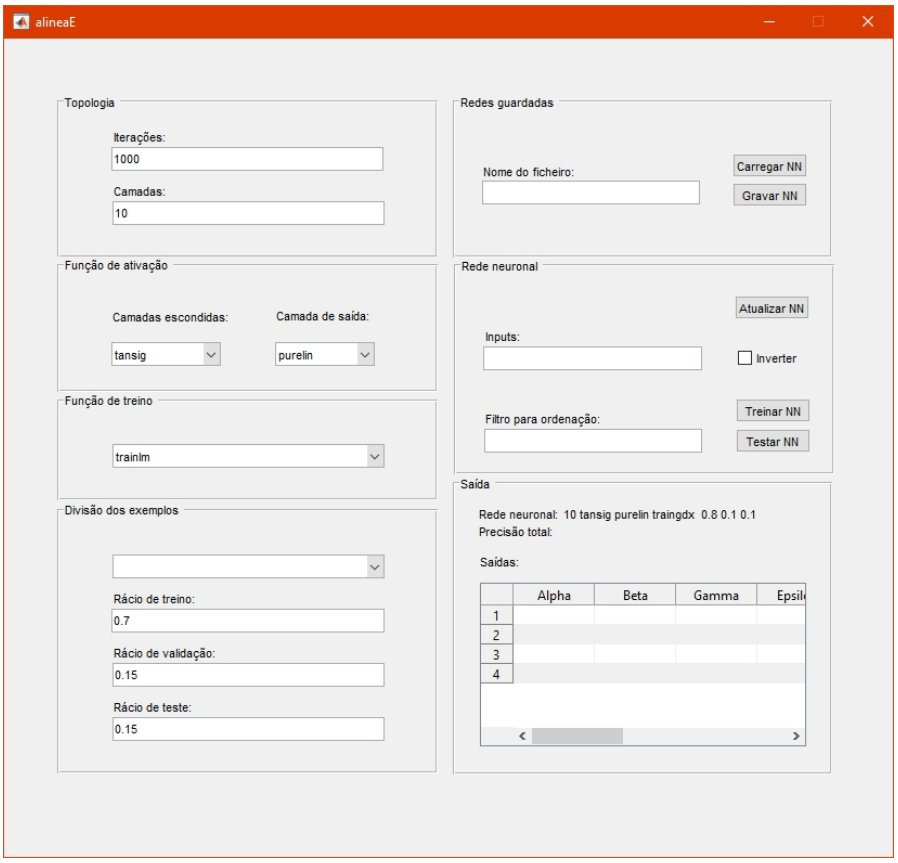
Como os resultados obtidos não foram os melhores foram realizados mais testes com outras redes neuronais e foi possível constatar que uma rede com precisão de 99% e precisão de teste de 100% ainda deu pior resultado com cerca de 20% (rede neuronal com 1 camada escondida de 10 neurónios, com função de ativação “tansig”, com função de ativação de saída “tansig”, função de treino “traingdx”, com divisão de exemplos “divideint = {0.8, 0.1, 0.1}”). Ainda foi possível testar uma rede com precisão total de 99% e 90% de precisão de testes obteve cerca de 50% de precisão total (rede neuronal com 1 camada escondida de 10 neurónios, com função de ativação “tansig”, com função de ativação de saída “tansig”, função de treino “trainlm”, com divisão de exemplos “dividerand = {0.8, 0.1, 0.1}”), como é possível ver na imagem abaixo representada, sendo esta a rede com melhores resultados das redes que foram testadas.

```
Carregada a rede neuronal com 99.000000 de precisão total e 90.000000 de precisão de teste.

Precisão total: 50.000000
```

7. Alínea e

A aplicação gráfica foi criada com o GUIDE do MatLab e abaixo é possível ver a interface gráfica com as várias tarefas pretendidas no enunciado.



Na rede neuronal é possível introduzir os inputs e um filtro para a ordenação, os inputs é a pasta de onde vêm as imagens para treino ou teste (dependendo do botão clicado "Treinar NN" ou "Testar NN") e o filtro para a ordenação serve para ordenar as imagens pois o MatLab carrega as imagens de forma desordenada. Um exemplo para carregar e ordenar as imagens da pasta 3 seria (sem as aspas):

Inputs: "Pasta3/*.jpg";

Filtro para ordenação: "letter_bnw_test_%d".

8. Conclusão

Com o desenvolvimento deste trabalho foi possível aprender sobre redes neuronais, e as diversas configurações das redes neuronais e ainda sobre Matlab.

Durante o desenvolvimento deste trabalho foram surgindo problemas e desafios que foram superados com a ajuda dos professores da disciplina, os apontamentos por eles disponibilizados e da Internet.