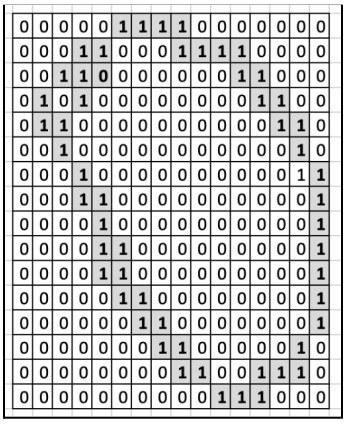
Aprendizagem Computacional

Relatório do Trabalho Prático 2a



OCR - Optical Character Recognition

2021/2022

Mestrado em Engenharia Informática

PL3	Pedro Rodrigues	2018283166	pedror@student.dei.uc.pt
PL3	Miguel Rabuge	2018293728	rabuge@student.dei.uc.pt

Índice

1. OCR - Optical Character Recognition

Neste primeiro capítulo, tivemos como objetivo gerar dados e desenhar redes neuronais para classificar dígitos (0-9). Fazemos uma explicação detalhada sobre o *dataset*, a arquitetura das redes neuronais, os resultados obtidos e as conclusões que retiramos dos mesmos, respondendo às perguntas colocadas no enunciado.

1.1. Dataset

Em termos de dados, foram gerados cerca de 1000 dígitos (20 *mpaper* sets), com igual distribuição entre os mesmos (100 de cada), por ambos os elementos do grupo.

A performance de qualquer técnica de *Machine Learning* depende, em grande parte, da quantidade, bem como da qualidade, dos dados que este utiliza.
Deste modo, é correto afirmar que mais e melhores dados possibilitam o nosso modelo a ter melhores performances. A quantidade é fácil de medir, porém a qualidade é algo mais abstrato. Assim, definimos como qualidade a naturalidade dos dígitos, ou seja, quão mais parecidos estes forem com aqueles que escreveríamos naturalmente numa folha de papel, não procurando a perfeição, mais qualidade terão. É esta naturalidade que nos permite treinar redes neuronais mais robustas, dado que não conhecem apenas uma representação perfeita, de um determinado digito, mas várias imperfeitas, como se pode verificar abaixo:

Figura 1 - P1.mat

1.2. Arquitetura das Redes Neuronais

Neste subcapítulo detalhamos as implementações dos 3 modelos abaixo:

1.2.1. Filtro + Classificador (1 Camada)

Neste primeiro modelo, foi utilizado como filtro uma memória associativa que procura "corrigir" os dados de entrada para dados tendencialmente mais perfeitos:

Figura 2 - Dígitos Perfeitos

Deste modo, como demonstração, treinando a memória associativa com 500 dígitos de entrada e utilizando 10 dígitos não pertencentes aos 500, podemos observar a saída do filtro:

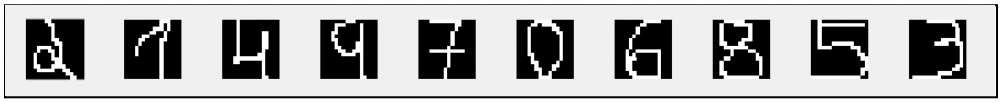


Figura 3 - "Dígitos desordenados"

Figura 4 - Resultado do filtro associativo para os digitos desordenados

Assim, o trabalho de classificação pela rede neuronal será em teoria mais simples, dado que não tem de classificar dígitos naturais, mas sim aproximações desses dígitos a dígitos perfeitos, dado que são estes últimos os que são passados de entrada ao classificador.

O classificador utilizado nesta implementação é descrito abaixo.

1.2.2. Classificador (1 Camada)

Para este classificador foi utilizada uma rede neuronal com uma camada de 10 neurónios. É definida a entrada como um dígito (imagem 16x16) transformado num vetor de tamanho 16*16 = 256. No final, o resultado da rede é um vetor de tamanho 10:

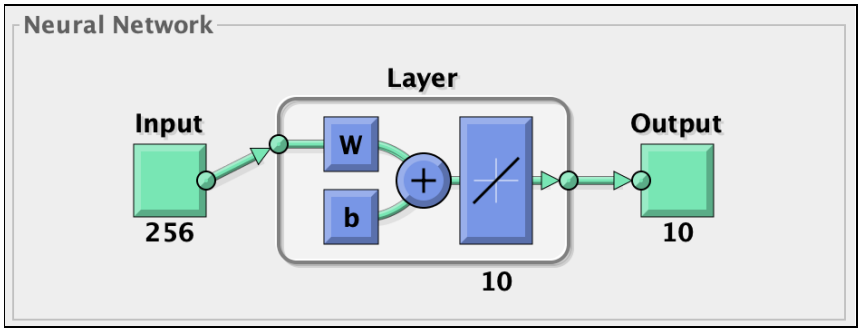


Figura 5 - Rede Neuronal (1 Camada)

Nota: A função de ativação apresentada na figura 5 é Linear (purelin), porém foram implementadas também a Sigmoidal (logsig) e a de heaviside (hardlim)

A rede neuronal é treinada tendo como *Target* a matriz identidade (10x10) concatenada de forma a ser do tamanho do input, dado que este se encontra ordenado tal que: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, …

Para o treino, é utilizado o método do gradiente caso a função de ativação seja linear ou sigmoidal, e a regra do perceptrão para a função de heaviside.

1.2.3. Classificador (2 Camadas)

De forma semelhante ao classificador acima, o classificador com duas camadas funciona de forma semelhante, porém com mais uma camada escondida que tem 50 neurónios e tendo sempre como função de ativação da *output layer* a função de heaviside.

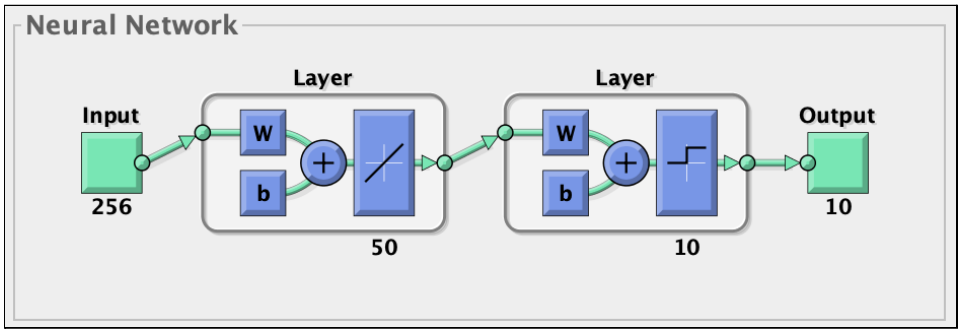


Figura 6 - Rede Neuronal (2 Camadas)

Nota:

- É possível alterar a função de ativação da *hidden layer* entre Linear (purelin), Sigmoidal (logsig) ou de Heaviside (hardlim)

1.3. Resultados

Os resultados apresentados têm como métrica de comparação a *accuracy*. Os resultados aqui demonstrados são obtidos correndo o script *train_all.m* para *treinar as redes neuronais, e correndo o accuracy.m* para calcular os valores.

O dataset de treino corresponde aos N elementos de input, enquanto que o dataset de teste corresponde aos seguintes 50 elementos:

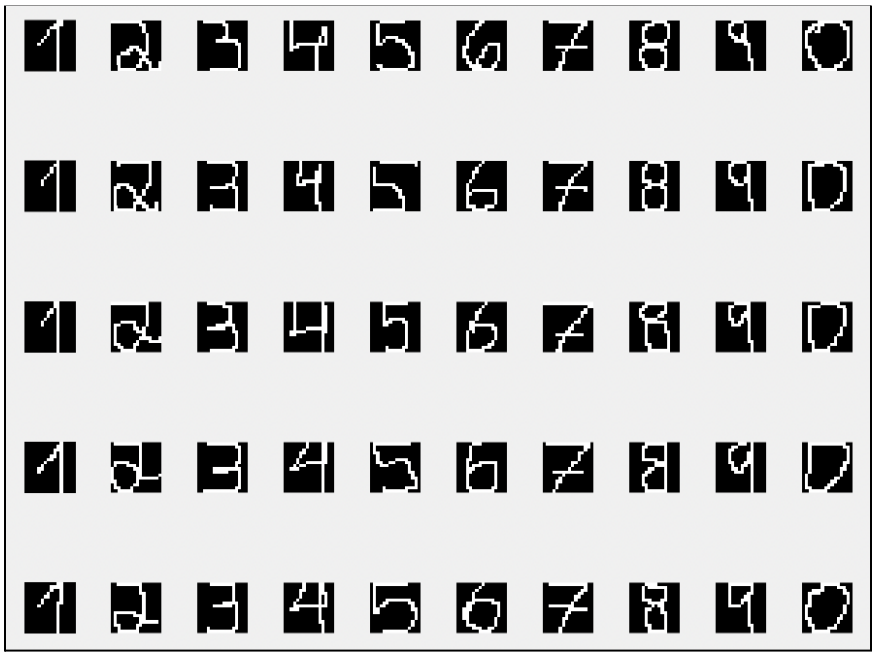


Figura 7 - AccuracyTest.mat

#Input	Classificador	Ativação	Treino	Teste
200	Filtro + 1 Layer	Linear	1.00	0.38
		Heaviside	1.00	0.24
		Sigmoidal	1.00	0.40
	1 Layer	Linear	0.925	0.78
		Heaviside	0.885	0.74
		Sigmoidal	0.89	0.82
	2 Layers	Linear	0.875	0.50
		Heaviside	0.10	0.10
		Sigmoidal	0.665	0.40

#Input	Classificador	Ativação	Treino	Teste
500	Filtro + 1 Layer	Linear	1.00	0.80
		Heaviside	0.994	0.72
		Sigmoidal	1.00	0.80
	1 Layer	Linear	0.954	0.90
		Heaviside	0.924	0.88
		Sigmoidal	0.88	0.82
	2 Layers	Linear	0.798	0.66
		Heaviside	0.10	0.10
		Sigmoidal	0.594	0.50
1000	Filtro + 1 Layer	Linear	0.995	0.90
		Heaviside	0.996	0.80
		Sigmoidal	0.999	0.92
	1 Layer	Linear	0.97	0.96
		Heaviside	0.955	0.92
		Sigmoidal	0.811	0.84
	2 Layers	Linear	0.747	0.7
		Heaviside	0.10	0.10
		Sigmoidal	0.647	0.62

Tabela 1 - Resultados Experimentais de *Accuracy*

1.4. Conclusões

Por fim, considerando os resultados experimentais, respondemos às questões colocadas no enunciado:

- Como é que os dados influenciam a performance do sistema de classificação?

    ➢ Como afirmado no subcapítulo 1.1. (Dataset) e posteriormente verificado experimentalmente no subcapítulo 1.3. (Resultados), a quantidade de dados tem um direto impacto na forma como o sistema generaliza para novos dígitos. Note-se que as redes treinadas com 1000 dígitos de *input* generalizam melhor do que redes treinadas com 200 ou 500 dígitos, como se pode verificar na última coluna - Teste - dos resultados.

- Que arquitetura apresenta melhores resultados?

    ➢ De acordo com a nossa implementação, a arquitetura que aparenta apresentar melhores resultados é a rede com 1 camada, sem filtro, como podemos verificar na Tabela 1, seguido pelo classificador de 1 camada, com filtro, apesar deste último não generalizar tão bem.

- Qual a vantagem, se alguma, de uma *softmax layer*?

    ➢ A softmax layer permite à rede neuronal retornar como output uma distribuição de probabilidades em vez de números que variam entre -inf e +inf. Esta representação do output permite não utilizar heurísticas para escolher o resultado mais provável, como colocar o maior número a 1 e os restantes a 0. Com a softmax, escolhe-lhe simplesmente o mais provável. Por outro lado, permite também ao utilizador interpretar o resultado da rede como as probabilidades que a rede dá a cada uma das classes do output, ou seja, a certeza/incerteza probabilística que a rede tem na classificação.

- Qual é a melhor função de ativação?

    ➢ De acordo com os resultados experimentais, a função de ativação que apresenta melhor performance ao longo dos vários sistemas de classificação, com mais ou com menos dados, é a função de ativação linear.

- O sistema de classificação consegue alcançar os principais objetivos (classificação de dígitos)? Qual a percentagem de dígitos bem classificados?

    ➢ Podemos afirmar que os sistemas de classificação baseados numa rede com 1 camada (com e sem filtro) conseguem classificar dígitos, com elevado grau de certeza, entre os 80% e 99% de *accuracy*.

    Relativamente aos sistemas de classificação com uma rede de 2 camadas, utilizando como função de ativação a função linear ou logarítmica, sim, se lhe fornecermos dados suficientes, atingindo apenas cerca de 70% de accuracy com 1000 dígitos. Por outro lado, a função de heaviside não, devido a uma possível falha na implementação.

- Como é a capacidade de generalização? O sistema de classificação é robusto o suficiente? Qual a percentagem de novos dígitos bem classificados?

    ➢ A capacidade de generalização é muito boa, nomeadamente na rede com 1 camada, sem filtro, onde a *accuracy*, tanto de treino como de teste, são ambas altas e muito semelhantes, de volta dos 90% com 1000 dígitos, sendo portanto um sistema robusto.

    A rede com 1 camada, com filtro, aparenta ter tendência para dar *overfit* nos dados de treino, pelo que depois não generaliza tão bem para os de teste, que desconhece, como se pode observar para 200 e 500 dígitos, onde desce de mais de 90% para cerca de 35% e 70%, respetivamente, não sendo portanto muito robusto

    A rede com 2 camadas aparenta generalizar bem, na medida em que os resultados da *accuracy* no dataset de teste são semelhantes aos de treino, porém não classifica com grande taxa de precisão os dígitos, ficando-se pelos 65% de teste com 1000 dígitos.