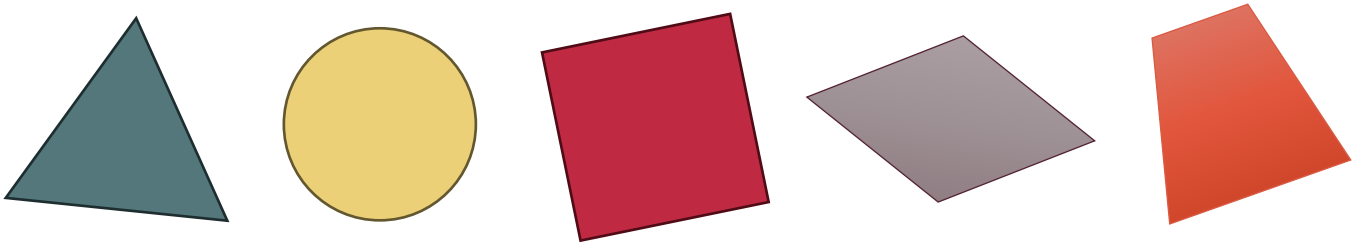
Classificar Formas Geométricas, uma análise de configurações e desempenho.

“Redes Neuronais – Feedforward”

Conhecimento e Raciocínio 2024-2025



- António Pedroso 2021132042
- Bruno Correia 2015015880

I. Objetivo de Estudo Proposto

- Foi-nos proposto, no âmbito da unidade curricular de Conhecimento e Raciocínio do ano letivo 2024/2025, o desenvolvimento de redes neuronais feedforward – da configuração dos seus parâmetros e hiperparâmetros e análise do seu desempenho na classificação de um dataset de imagens de figuras geométricas variadas.
- Neste relatório vão ser descritos os passos e justificadas as opções tomadas na resolução das alíneas do enunciado, destacando as caraterísticas principais de cada rede e implicações das mesmas nos resultados obtidos.
- Todo o código descrito e explicado no relatório e trabalho prático foi desenvolvido utilizando o ambiente de programação MATLAB (2024).

Dataset:

- Os dados usados para treinar, validar e testar o desempenho das redes desenvolvidas, além dos poucos criados através de desenho digital, são provenientes da pasta fornecida e são compostos por 6 classes diferentes relativas a figuras geométricas descritas abaixo:

    1. Classe ***'circle'*** **–** figura geométrica representativa de um círculo;
    2. Classe ***'kite'*** **-** figura geométrica representativa de um quadrilátero;
    3. Classe ***'parallelogram'*** **-** figura geométrica representativa de um paralelograma;
    4. Classe ***'square'*** **-** figura geométrica representativa de um quadrado;
    5. Classe ***'trapezoid'*** **-** figura geométrica representativa de um trapézio

As figuras estão proporcionalmente distribuídas por pastas com o nome da classe que as identifica, dentro de outras 3 pastas ***start, test*** *e* ***train*** com respetivamente 5, 10 e 50 exemplos diferentes de cada, que perfazem um *dataset* total de 390 imagens (224*224px).

II. Descrição das Metodologias Implementadas

1. Pré-processamento dos Dados

- As imagens do dataset foram **transformadas** com o intuito de reduzir o tempo computacional significativamente, sem perder as principais caraterísticas que as distinguem.
- Foi aplicada a função **rbg2gray** para reduzir 3 vezes a dimensão dos dados, de RBG para escalas de cinza, na condição if size(img, 3) == 3, que apenas executa a função caso a imagem tenha 3 canais (RGB), visto que as figuras estão a preto e branco.
- Foi aplicada outra função de transformação – **imresize**, redimensionando novamente as imagens, desta vez para [28 x 28] pixéis.
- Finalmente aplicou-se a função **imbinarize**, convertendo os valores de intensidade (0-255) em 0s e 1s – focando no contraste entre fundo e objeto.

2. Estruturação dos Dados

- Estrutura de dados **shapes**: é um array de strings que contém a lista com os nomes das classes, usada no pré-processamento de dados;

- Estrutura de dados **labels**: array que armazena os rótulos de todas as imagens, indicando a classe a que cada uma pertence – é o output esperado (alvo) durante o treino da rede;

- Estrutura de dados **binaryImages**: matriz que armazena todas as imagens como vetores de 784 elementos binários (28x28) onde cada coluna representa uma imagem e cada linha um pixel binarizado.

3. Processo de Treino

De seguida descrevem-se os processos de treino das diferentes redes, associados à alínea do enunciado e explicados detalhadamente:

a) ***net = feedforwardnet(x);*** (x: nº neurónios/camada oculta)

Uma rede *feedforward*, ou *multilayer perceptron* (**MLP**), é um tipo de rede neuronal artificial composta por neurónios organizados em camadas (entrada: 784, oculta(s): x, saída: 6) onde há um ajuste dos vetores de pesos, **w,** durante o treino para minimizar uma função de perda que calcula a diferença entre previsões (output da rede) e alvos (labels) – sendo usada para problemas de regressão e de **classificação**.

Primeiramente treinámos a rede com a totalidade das amostras da pasta *start* até chegar a uma taxa de acurácia de 100% - utilizámos os parâmetros **default** do MATLAB para redes neuronais, fazendo variar unicamente o número de camadas ocultas e o seu número de neurónios associados.

➡ Parâmetros por *default* em redes neuronais no MATLAB:

i. **tansig:** tangente sigmoide – comprime os valores entre [-1, 1] introduzindo não linearidade à rede, na camada oculta.

$$f'(x) = 1 - f(x)^2$$

ii. **purelin:** produz uma saída linear contínua, sem transformação.

$$f(x) = x \quad , \quad f'(x)=1$$

iii. **trainlm:** combina Método de Gauss-Newton (para convergência rápida) e Gradiente Descendente (para estabilidade, quando Gauss-Newton falha), ajustando iterativamente os pesos dos neurónios para minimizar a função de

perda, produzindo um valor não negativo que diminui com a precisão da rede.

$$\Delta w = -(J^T J + \mu I)^{-1} J^T e \qquad (\Delta W: \text{variação dos pesos})$$

iv. **mse**: função de perda que calcula o erro quadrático médio entre os **outputs** e os **alvos** da rede, onde **N** é o número de amostras – amplamente usada em regressão e, por default, no MATLAB para redes neuronais.

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (t_i - y_i)^2$$

v. **train split:** divisão das amostras entre treino, validação e teste.
net.divideFcn: 'dividerand':
net.divideParam.trainRatio: 1.00 (70% para treino p/default),
net.divideParam.valRatio: 0.00 (15% para validação p/default),
net.divideParam.testRatio: 0.00 (15% para teste p/default).

vi. **goal:** 0 (erro alvo mínimo p/default) – parâmetro de convergência: o treino continua a correr até que a função de perda atinja 0 (o que na prática acontece raramente, sendo frequentemente ajustado para valores positivos pertos de 0).

Com estes valores padrão do MATLAB para redes neuronais e no contexto da alínea a) do enunciado, fomos treinar as nossas redes com a **totalidade das amostras** da pasta *train,* obtendo precisões globais de 100% para todas as topologias testadas, variando apenas o número de Epochs que a rede demorou a chegar ao *goal* definido – resultado espectável uma vez que se usaram todos os exemplos no treino (a rede não classificou nada que não tivesse visto antes).

Desempenhos das redes na alínea a):

NumNeurons	MeanAccuracy	Epochs	Run
10	100	8	1
20	100	6	1
50	100	4	1
[10 10]	100	15	1
[20 10]	100	12	1
[50 20]	100	5	1
10	100	9	2
20	100	5	2
50	100	4	2
[10 10]	100	18	2
[20 10]	100	11	2
[50 20]	100	5	2

(...)

b) Nesta alínea do enunciado expandimos o nosso conhecimento de redes neuronais, explorando uma série novos de parâmetros e hiperparâmetros que iremos explicar seguidamente:

i. **poslin:** mais conhecida com ReLU (rectified linear unit) é uma função de ativação linear que preserva valores positivos, colocando os negativos a 0.
f(x) = max(0, x)

ii. **logsig:** sigmoide logística – função de ativação aplicada nas camadas ocultas - comprime a entrada para o intervalo [0, 1] introduzindo uma não-linearidade suave (pouco eficaz em redes profundas).

$$f(x) = 1 / (1 + e^{(-x)})$$

iii. **hardlim:** função de ativação binária. Backpropagation requer derivadas da função de ativação para ajustar os pesos – a derivada de hardlim é 0 em todos os pontos o que impossibilita o treino da rede.
f(x) = 1 se x >= 0, f(x) = 0 se x < 0

iv. **purelin:** tentativa de usar como função de ativação – passa a entrada diretamente como saída sem transformação, não introduz não-linearidade o que nos fez obter resultados muito baixos semelhantes a iii.
f(x) = x

v. **softmax:** converte os valores da camada final em probabilidades normalizadas que somam 1, ideal para classificação multiclasse – permitiu-nos obter valores altos de acurácia usando esta função de saída.

$$f(xi) = e^{x} * i / \sum_{j=1}^{K} e^{x} * j$$, K = nº classes

vi. **crossentropy:** a função de perda entropia cruzada mede a divergência entre a distribuição de probabilidades prevista pela rede e os alvos reais – otimizada para classificação multiclasse e emparelha bem com **softmax.**

$$CE = -1/N * \sum_{i=1}^{N} \sum_{c=1}^{K} t*i,c * \log(y*i,c).$$, K = nº classes

vii. **trainscg:**(Scaled Conjugate Gradient) função de treino que utiliza gradientes conjugados para ajustar os pesos de forma eficiente, sendo **mais leve** que **trainlm** é ideal para datasets médios, como no nosso caso (390 amostras) com convergências rápidas que permitem efetuar as iterações em menos tempo com resultados satisfatórios.

$w_{k+1} = w_k + \alpha*k*p*k$, p*k é a direção de busca, a*k é o passo ajustado

viii. **trainrp**:(Resilient Backpropagation) ajusta os pesos com base no sinal do gradiente, ignorando a magnitude – o que o torna robusto a ruídos e eficiente em

redes com menos camadas, mas pode ser lento em convergência.

$\Delta w_{ij} = -sign(\partial E / \partial w_{ij}) * \Delta w_{ij}$, onde Δwij é o passo adaptativo

ix. **trains:**(Sequential Order Incremental Training) treina a rede numa ordem sequencial, sendo útil na adaptação a padrões locais mas pode exigir mais iterações.

$w_{k+1} = w_k - \eta (\partial E / \partial w_k)$, onde η é a taxa de aprendizagem

x. **traingd:**(Gradient Descent) ajusta os pesos com base no gradiente descendente simples, sendo o método mais básico utilizado – lento.

$w_{k+1} = w_k - \eta (\partial E / \partial w_k)$

Desempenhos das melhores redes na alínea b):

Topology	Activation	EndTrainFcn	PerformFcn	TrainFcn	TrainRa	ValRati	TestRati	MeanTest	MeanGlo	MeanEpoc	Run
[50 30]	logsig	softmax	mse	trainscg	0,8	0,1	0,1	84,44	97,11	55,67	4
[50 30]	logsig	softmax	crossentropy	trainscg	0,7	0,15	0,15	82,22	93,97	53,70	5
[50 30]	logsig	softmax	mse	trainscg	0,8	0,1	0,1	81,00	92,67	48,50	5
[50 30]	logsig	softmax	crossentropy	trainrp	0,8	0,1	0,1	79,33	93,33	34,60	5
[40 30]	tansig	purelin	mse	trainscg	0,7	0,15	0,15	79,26	93,56	68,67	2
50	tansig	tansig	mse	trainlm	0,8	0,1	0,1	78,67			1
[50 30]	logsig	softmax	mse	trainscg	0,7	0,15	0,15	77,78	90,22	56,00	4
50	tansig	tansig	mse	trainscg	0,8	0,1	0,1	77,33			1
[50 30]	logsig	softmax	mse	trainrp	0,7	0,15	0,15	76,30	90,44	36,00	4
[20 10]	tansig	purelin	mse	trainscg	0,7	0,15	0,15	76,30	92,67	64,00	2

Não é de espantar que as melhores redes treinadas têm todas a mesma topologia **[50 30]** pois 2 camadas ocultas representam um bom equilíbrio entre capacidade de generalização e redução de overfitting, assim como o número de neurónios que é suficiente para a complexidade das amostras testadas. Outra caraterística que as une é a aplicação da **sigmoide logística** nas camadas ocultas que lhes confere uma não-linearidade suave que é ideal para redes pouco profundas. Confirmou-se também a hipótese acima referida de que a função de saída **softmax** iria emparelhar bem com a função de perda **crossentropy** devido ao facto de que ambas são indicadas para problemas de classificação multiclasse (6) onde a softmax gera probabilidades normalizadas e a crossentropy penaliza as discrepâncias entre essas previsões e os outputs alvo. Outra especificidade que une as nossas 3 melhores redes é a função de treino **trainscg** que é ideal para topologias médias, como [50 30], acelerando a convergência tendo em consideração as melhores direções de busca por pesos que conferem melhores desempenhos na rede. Para efeitos de diversidade nas redes e para conseguir extrair conclusões mais ricas, optámos por escolher a quarta melhor, em que foi aplicada a função de treino **trainrp –** apesar de nos ter conferido uma performance inferior de teste, devido ao facto de que utiliza somente o sinal do gradiente para atualizar os pesos, consideramos que é satisfatória em comparação com as centenas de outras redes testadas. Algo inesperado e talvez inexplicável é o facto da combinação softmax + **mse** ter sido a nossa melhor rede, já que teoricamente o mse é mais adequado para regressão pois calcula o erro quadrático médio entre previsões e alvos, minimizando diferenças numéricas. Talvez a binarização das imagens possa ter reduzido a

variabilidade dos erros e tornado o mse relativamente melhor, e o trainscg pode ter ajustado os pesos de forma a compensar essa limitação.

c) Resultados dos testes das 3 melhores redes (treinadas somente com imagens da pasta **train**) com as amostras da pasta **test:**

NetID	GlobalAccuracy
1	88,33
2	83,33
3	80,00

Verifica-se que as redes estão com uma boa capacidade de generalização, mas ainda há bastante margem para melhorias. Resultados satisfatórios mas espera-se ainda um aumento de performance.

Desempenho das redes após treino com os dados da pasta **test:**

Ne	GlobalAcc	TestAccu	StartSetAcc	TrainSetA	TestSetAc	Epochs	Run
1	99,50	97,78	80,00	93,67	98,33	12,30	2
2	96,33	96,67	83,33	92,00	96,67	11,80	4
1	98,00	95,56	90,00	91,67	100,00	11,50	1
2	96,33	95,56	76,67	94,00	96,67	9,90	1
2	95,00	95,56	83,33	94,67	95,00	11,80	2
1	96,50	95,56	93,33	96,67	95,00	8,30	4
1	94,33	95,56	93,33	96,33	88,33	7,10	5
2	96,50	95,56	83,33	94,33	96,67	11,40	5
1	95,83	92,22	93,33	96,67	93,33	10,30	3
3	80,00	86,67	83,33	91,33	80,00	6,00	3
2	91,33	85,56	76,67	92,00	96,67	9,00	3
3	80,00	83,33	83,33	91,33	80,00	6,00	1
3	80,00	78,89	83,33	91,33	80,00	6,00	4
3	80,00	76,67	83,33	91,33	80,00	6,00	2
3	63,67	66,67	83,33	91,33	80,00	8,50	5

Verifica-se o espectável: as redes aprenderam com os treinos de um conjunto maior de imagens, o que lhes vai conferir uma capacidade ainda maior de generalização.

Desempenho das redes após treino com a totalidade do dataset (pastas **train, start** e **test,** 390 amostras)**:**

Ne	GlobalAcc	TestAccur	StartSetAcc	TrainSetAc	TestSetAc	Epochs	Run
1	98,67	98,64	100,00	99,00	98,33	10,30	4
1	97,69	97,80	100,00	98,00	95,00	6,60	5
1	98,38	97,63	100,00	98,67	95,00	7,60	1
2	95,64	97,46	83,33	96,67	93,33	9,90	2
1	97,62	96,78	100,00	99,00	95,00	8,50	2
2	96,05	96,44	83,33	98,00	96,67	10,50	1
2	95,92	95,42	83,33	97,33	95,00	8,40	3
2	96,26	95,25	86,67	97,67	96,67	11,10	4
1	96,72	95,08	90,00	98,00	93,33	7,70	3
2	95,82	94,75	83,33	97,00	95,00	9,30	5
3	88,97	91,02	83,33	91,33	80,00	6,00	1
3	88,97	90,34	83,33	91,33	80,00	6,00	3
3	88,97	90,17	83,33	91,33	80,00	6,00	2
3	88,97	89,32	83,33	91,33	80,00	6,00	4
3	88,97	88,14	83,33	91,33	80,00	6,00	5

Todas as redes aumentaram no seu desempenho – novamente solidifica a hipótese de que aumentar o dataset de treino vai resultar num aumento proporcional na capacidade de classificação de imagens de figuras geométricas como é o nosso caso. Algumas das redes inclusivamente atingiram uma precisão global de 100% o que, contrariamente ao que possa parecer, é algo indesejado pois reflete um **overfitting** aos dados onde as redes estão a classificar no teste amostras que já viram no treino.

d) Resultados da performance das redes na classificação de imagens desenhadas (custom images):

Ne	Accuracy	circle	kite	parallelog	square	trapezoid	triang	Run
1	66,67	80,00	20,00	60,00	100,00	40,00	100,00	2
2	60,00	80,00	0,00	80,00	100,00	0,00	100,00	2
1	53,33	40,00	0,00	20,00	100,00	80,00	80,00	1
2	50,00	80,00	0,00	20,00	80,00	60,00	60,00	1
3	40,00	20,00	20,00	20,00	40,00	40,00	100,00	1
3	40,00	40,00	20,00	40,00	20,00	40,00	80,00	2
1	33,33	40,00	20,00	20,00	40,00	20,00	60,00	3
2	20,00	40,00	20,00	20,00	0,00	0,00	40,00	3
3	16,67	40,00	20,00	20,00	0,00	0,00	20,00	3

Verifica-se uma redução muito significativa na acurácia das redes – isto deve-se principalmente a dois fatores:

1) a qualidade dos desenhos – as redes tentaram classificar imagens não só nunca antes vistas mas ligeiramente diferentes do que estão treinadas para identificar;
2) confirma-se o **overfitting** aos dados do treino – as redes estão com uma baixa capacidade de generalização.

5. Matrizes de Confusão

1-CiiR4)	circle	kite	parallelogram	square	trapezoid	triangle
circle	65	0	0	0	0	0
kite	0	65	0	0	0	0
parallelogram	0	1	61	1	1	1
square	0	0	0	65	0	0
trapezoid	1	0	4	0	59	1
triangle	0	2	2	0	1	60

1-CiiiR4)	circle	kite	parallelogram	square	trapezoid	triangle
circle	65	0	0	0	0	0
kite	0	65	0	0	0	0
parallelogram	0	0	63	1	0	1
square	0	0	0	65	0	0
trapezoid	0	0	2	0	63	0
triangle	0	0	0	0	0	65

3-CiiR1)	circle	kite	parallelogram	square	trapezoid	triangle
circle	61	0	0	1	3	0
kite	0	63	0	0	0	2
parallelogram	0	0	52	5	7	1
square	1	0	0	62	2	0
trapezoid	0	0	2	0	63	0
triangle	1	0	1	1	1	61

3-CiiiR1)	circle	kite	parallelogram	square	trapezoid	triangle
circle	65	0	0	0	0	0
kite	0	65	0	0	0	0
parallelogram	0	0	62	1	1	1
square	0	0	0	65	0	0
trapezoid	1	0	3	0	61	0
triangle	0	0	0	0	0	65

Podemos observar as matrizes de confusão relativas às redes 1 e 3 após o treino com a pasta de teste (alínea c) ii.) e treino com a totalidade do dataset (alínea c) iii.) – comprova-se que a capacidade das redes classificarem os exemplos das pastas vai aumentando em par com o número de amostras treinadas.

6. Classes com melhor Predição

Das centenas de redes que foram desenvolvidas ao longo deste trabalho prático, consegue-se depreender que as classes **circle, triangle, square** e **kite** são as que somam predições verdadeiras positivas em maior quantidade – são as mais fáceis de identificar. Por outro lado as classes **parallelogram** e **trapezoid** foram as que atingiram valores mais baixos (cerca de 96% nas matrizes de confusão finais relativas às melhores redes) por terem caraterísticas mais complexas.

III. Conclusões

O desenvolvimento deste trabalho prático permitiu-nos testar os nossos conhecimentos teóricos adquiridos ao longo do semestre. Conseguimos progredir com as melhores redes ao longo do mesmo e as hipóteses inicialmente lançadas foram-se verificando, pelo menos na sua maioria. É com satisfação que concluímos este relatório pois é o reflexo do trabalho árduo não apenas de escrita de código, mas de tratamento e análise de dados.

IV. Bibliografia

Forums MATLAB;

Duckduckgo;

Moodle – Ficheiros de apoio às aulas.