Licenciatura em Engenharia Informática

Relatório de Trabalho Prático

[Conhecimento e Raciocínio]
[Trabalho Prático]

Daniel Tinoco - 2021132552
Bruno Martins - 2022147149

11 de Maio de 2024

Instituto Superior de Engenharia
Politécnico de Coimbra

Conteúdo

1. Introdução

Este trabalho foi realizado no âmbito da Unidade Curricular de Conhecimento e Raciocínio no ano letivo de 2023/24. Tem como objetivo o desenvolvimento de uma rede neuronal do tipo feedforward capaz de prever, com base na informação fornecida nos Datasets, se um determinado indivíduo teve ou não um AVC. Foi desenvolvido com base nos programas abordados nas aulas práticas sobre matéria similar.

2. Preparação do Dataset

O Dataset escolhido foi, como inferido, o 2º, "Stroke", este tem como atributos os seguintes:

- age – A idade do individuo
- hypertension – Se tem ou não hipertensão
- heart_disease – Se tem ou não uma doença cardiológica
- ever_married – Se já foi, ou é, casado.
- Residence_type – Se vive numa residência rural ou urbana
- avg_glucose_level – O nível médio de glucose
- bmi – Body Mass Index

Para além disso tem mais duas colunas, o id, e a coluna 'stroke', esta última é uma coluna binária que é o 'target' do nosso dataset. A coluna serve apenas como identificador, como não influencia em nada o resultado (podendo apenas induzir a rede em erro) optámos por não a incluir durante as próximas fases (incluindo esta)

Os Datasets START e TEST já estavam prontos para uso; no entanto, o Dataset TRAIN, elemento central do projeto, estava incompleto. Como explicado no enunciado, o primeiro passo para a sua preparação é converter todos os valores do tipo string para valores numéricos. Para tal, criámos um script em MATLAB (datastorter.m) que faz exatamente isso. A funcionalidade dele é relativamente simples:

- Primeiro, temos os valores booleanos, ou seja, aqueles que, como “ever_married”, só têm duas opções: “Yes” ou “No”. Estes são fáceis de converter; basta compará-los ao valor que queremos representar por ‘1’, neste caso, “Yes”, e guardar o resultado.
- Em segundo lugar, temos o “smoking_status”. Neste caso, usamos um Map para associar cada elemento à sua chave correspondente, ou seja, um “Unknown” será convertido para ‘3’.

Com o Dataset pronto a ser utilizado, preenchemos o target usando a parte “retrieve” do Case Based Reasoning (CBR) para tentar prever se alguém teve ou não um AVC, baseando-nos no resto dos dados. Para isso, dividimos temporariamente o Dataset em dois: os casos conhecidos de AVC e os desconhecidos.

- O primeiro passo é definir os weight_factors, ou seja, os pesos. Decidimos, com base nos nossos conhecimentos e alguma pesquisa, dar maior importância à idade, hipertensão e ao estado de fumador, seguidos de doenças cardíacas, nível de glicose e BMI. Fatores de menor importância incluem o género, o estado civil e o tipo de residência.
- Para cada um destes, usamos a distância linear para obter a sua similaridade local, exceto para o estado de fumador, que usa a distância linear, exceto quando um dos casos a comparar tem o valor ‘3’ (ou seja, “Unknown”).
- Após calcular a similaridade global, se esta for maior ou igual ao threshold, o caso é guardado. Se não houver nenhum caso que supere o threshold, este é reduzido em 0.01, e a fase de retrieve é reiniciada. Este processo repete-se até haver pelo menos um caso.
- Se tivermos pelo menos um caso suficientemente similar, usamos a moda dos casos similares para prever se o caso atual teve ou não um AVC. Por exemplo, se tivermos três casos em que ocorreu um AVC e dois em que não ocorreu, o programa assumirá que o paciente teve um AVC.
- O caso é então adicionado ao Dataset para ser utilizado na previsão do próximo resultado.

3. Ficheiro START.csv

O ficheiro START.csv é um ficheiro simples com apenas dez exemplos. Devido ao seu tamanho e às restrições dadas no enunciado, que só permite a alteração das funções de ativação e de treino, as experiências executadas foram limitadas. Cada experiência foi executada cinquenta vezes, e os resultados apresentados no relatório são a média dessas iterações.

Nestas experiências, verificámos algumas curiosidades sobre as funções de treino e as combinações de funções de ativação

Começando primeiro pela experiência “Default”, verificamos que facilmente chegou aos 100% de precisão, e correu as 50 iterações em apenas 3.4 segundos.

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Precisão Global	Erro	Duração (50 Iterações)
Default	1	10	tansig, purelin	trainlm	100	0	3.4s

Figura 1 - Experiência Default do ficheiro START.csv

3.1 Variação das funções de treino

Após a realização da experiência Default decidimos variar a função de treino, concluímos usamos então as seguintes funções: ‘traingdx’, ‘traingd’, ‘trainc’, ‘trainbfg’ e ‘trainscg’;

Neste Dataset, em termos de precisão, pouca ou nenhuma diferença existe, com todas as redes chegando a uma média de 50, com a exceção da ‘traingd’, que ficou com uma média de 99.4, diferença anotámos. A maior diferença foi encontrada no tempo de execução, especialmente nas funções ‘traingd’ e ‘trainc’, que demoraram 14 e 314 segundos respetivamente, destas é de destacar a ‘trainc’, que demorou mais de 5 minutos num Dataset pequeno.

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Precisão Global	Erro	Duração (50 Iterações)
Variar Funcao de Treino							
1.1	1	10	tansig, purelin	traingdx	100	0	5.9s
1.2	1	10	tansig, purelin	traingd	99.4	0.006	14.4s
1.3	1	10	tansig, purelin	trainc	100	0	314.3s
1.4	1	10	tansig, purelin	trainbfg	100	0	6.2s
1.5	1	10	tansig, purelin	trainscg	100	0	3.5s

Figura 2 - Experiência com variação nas funções de treino do ficheiro START.csv

3.2 Variação das funções de ativação

Por fim, verificámos as funções de ativação, nesta fase conseguimos tirar algumas conclusões importantes, nomeadamente que as funções hardlim e logsig não são ideais para a camada de output, conseguindo métricas de apenas 50%, já as funções tansig e purelin são muito mais viáveis para esta última camada.

No entanto é importante destacar que as funções logsig e hardlim aparentam ser viáveis para funções de camadas escondidas obtendo métricas de 100% e 95% respetivamente.

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Precisão Global	Erro	Duração (50 Iterações)
Variar Funções de Ativação							
2.1	1	10	tansig,hardlim	trainlm	50	0.5	2.7s
2.2	1	10	purelin,tansig	trainlm	97.2	0.028	3.4s
2.3	1	10	logsig,tansig	trainlm	99	0.01	3.1s
2.4	1	10	tansig,logsig	trainlm	50	0.5	3.4s
2.5	1	10	purelin,purelin	trainlm	100	0	2.7s
2.6	1	10	hardlim,purelin	trainlm	95	0.05	3.2s
2.7	1	10	logsig,purelin	trainlm	100	0	5.5s

Figura 3 - Experiência com variação nas funções de ativação do ficheiro START.csv

4. Ficheiro TRAIN.csv

O ficheiro TRAIN.csv apresenta um desafio significativo para a rede neuronal devido ao seu tamanho considerável. Esta complexidade refletiu-se nos resultados obtidos, onde a precisão global raramente excedeu 80%, situando-se frequentemente entre 75% e 80%. Contudo, as redes implementadas na secção 4.5 deste estudo destacaram-se, ultrapassando frequentemente os 80% de precisão, com algumas alcançando até os 90%.

Todas as experiências foram realizadas em 50 iterações, os resultados médios de ambas as precisões e os das melhores redes foram anotados.

É importante realçar que os resultados das melhores redes pouco demonstram, para além de uma rede que tenha sido especialmente boa, não deixam tirar conclusões claras e foram anotados puramente para decidir quais redes passar à próxima fase do trabalho prático.

Começando pela configuração Default, esta já contem a segmentação e obteve resultados que, quando comparados ao resto, são surpreendentes

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Divisão de Exemplos	Media Precisão Global	Media Precisão De Teste	Duração (50 Iterações)	Melhor Rede (Precisao Global) [Global, Teste]	Melhor Rede (Precisao Teste) [Global, Teste]
Default										
Default	1	10	tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	77.423	74.6087	9.2901	[81.475410, 72.826087]	[77.704918, 82.608696]

Figura 4 - Experiência Default do ficheiro TrainFiltered.csv

4.1 Variação nas configurações das camadas

Começando pelas configurações de camadas, podemos notar lições que podem ser contraintuitivas, apercebemo-nos que, ter mais camadas nem sempre é bom, as redes que mais têm, embora ter resultados relativamente bons no que toca à precisão global, verificamos que a precisão de teste estas ficaram muito aquém do esperado. Este é o caso das redes 1.7 e 1.8.

Aumentar os neurónios por camada, no entanto, aparentou demonstrar resultados melhores, como é visível nos exemplos 1.5, 1.9, 1.11 e especialmente 1.10.

Por fim, para saciar a curiosidade corremos uma rede com apenas 1 camada cujo possuía apenas um neurónio, esperávamos resultados horríveis, mas ficámos positivamente surpreendidos ao verificar que, embora se tenha verificado um decréscimo, este não foi de todo acentuado, com a média global apenas perdendo 1%, e a de teste subindo 1% em relação à default.

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Divisão de Exemplos	Media Precisão Global	Media Precisão De Teste	Duração (50 Iterações)	Melhor Rede (Precisao Global) [Global, Teste]	Melhor Rede (Precisao Teste) [Global, Teste]
Configuração das camadas										
1.1	2	5, 5	tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	77.518	73.5652	5.7263	[82.622951, 75.000000]	[78.196721, 82.608696]
1.2	2	10,10	tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	77.4066	72.5435	6.3472	[81.147541, 70.652174]	[75.737705, 81.521739]
1.3	3	5, 10, 5	tansig,tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	77.3049	74.2391	7.9512	[82.786885, 73.913043]	[77.213115, 84.782609]
1.4	3	10, 10, 10	tansig, tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	77.3246	77.3246	9.5609	[82.786885, 70.652174]	[82.295082, 81.521739]
1.5	2	20,20	tansig,tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	78.9705	70.6304	14.1122	[85.245902, 75.000000]	[81.967213, 81.521739]
1.6	5	10,10,10,10,10	tansig, tansig,tansig,tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	77.6262	72.3261	19.7245	[84.754098, 68.478261]	[76.229508, 81.521739]
1.7	10	10(x10)	tansig (x10), purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	75.4885	70	58.8589	[83.114754, 61.956522]	[79.016393, 79.347826]
1.8	10	10(x20)	tansig (x20), purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	73.0459	68.1522	364.6313	[83.934426, 63.043478]	[83.934426, 63.043478]
1.9	3	10, 20, 30	tansig, tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	78.4721	71.7391	31.3059	[83.934426, 79.347826]	[80.655738, 81.521739]
1.10	2	50,50	tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	82.0984	69.4348	451.193	[88.196721, 70.652174]	[84.590164, 79.347826]
1.11	3	30, 20, 10	tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	78.4066	71.7174	35.6431	[84.426230, 68.478261]	[82.786885, 80.434783]
1.12	1	1	tansig, tansig, purelin	trainlm	dividerand = [0.7, 0.15, 0.15]	76.377	75.6739	3.9385	[78.196721, 77.173913]	[76.229508, 84.782609]

Figura 5 - Experiência com variação nas configurações das camadas do ficheiro TrainFiltered.csv

4.2 Variação nas funções de treino

Passando às funções de treino, podemos tomar conclusões claras, quer quanto à sua qualidade, quanto à sua eficiência temporal. Começando pela qualidade verificámos que, no que toca à precisão global apenas uma conseguiu superar a Default, 'trainlm', senda esta, a 'trainbr' que demonstrou resultados significativamente melhor do que o resto, no entanto, no que toca a precisão de teste, a 'trainscg' demonstrou resultados promissores, no outro lado do espectro temos a 'trainc', que demonstrou resultados visivelmente piores do que o resto.

No que toca à eficiência temporal, a maioria teve durações parecidas, com a exceção de 3 casos notáveis, do mais rápido para o menor temos a 'trains' e 'trainr', que demorarm quase 3 minutos, 'trainr' que demorou quase 8 minutos, e, por fim, temos outra vez a 'trainc', que demorou quase 6 horas no total.

Tirando à parte a 'trainc' podemos concluir que esta função simplesmente não é adequada para este tipo de problema, uma vez que demora muito e demonstra resultados indesejáveis.

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Divisão de Exemplos	Media Precisão Global	Media Precisão De Teste	Duração (go Iterações)	Melhor Rede (Precisao Global) [Global, Teste]	Melhor Rede (Precisao Teste) [Global, Teste]
				Funcão de treino						
2.1	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	70.3475	69.1087	65.0417	[75.409836, 77.173913]	[75.409836, 77.173913]
2.2	1	10	tansig, purelin	trainbfg	dividerand = {0.7, 0.15, 0.15}	76.1311	73.8261	12.1713	[78.852459, 71.739130]	[76.885246, 85.869565]
2.3	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}	76.6098	73.6304	8.5004	[80.000000, 79.347826]	[76.557377, 81.521739]
2.4	1	10	tansig, purelin	trainoss	dividerand = {0.7, 0.15, 0.15}	76.1475	73.7826	10.7085	[78.688525, 68.478261]	[77.377049, 84.782609]
2.5	1	10	tansig, purelin	trains	dividerand = {0.7, 0.15, 0.15}	73.7836	74.5217	149.269	[77.868852, 73.913043]	[75.245902, 81.521739]
2.6	1	10	tansig, purelin	traingdx	dividerand = {0.7, 0.15, 0.15}	75.1869	73.5652	14.115	[79.508197, 79.347826]	[77.868852, 83.695652]
2.7	1	10	tansig, purelin	traincgp	dividerand = {0.7, 0.15, 0.15}	76.4361	74.913	11.3413	[78.688525, 76.086957]	[77.213115, 83.695652]
2.8	1	10	tansig, purelin	trainc	dividerand = {0.7, 0.15, 0.15}	66.8689	65.6739	19,451.00	[70.327869, 72.826087]	[65.409836, 77.173913]
2.9	1	10	tansig, purelin	traincgf	dividerand = {0.7, 0.15, 0.15}	76.1377	73.5	11.6011	[78.688525, 69.565217]	[78.360656, 82.608696]
2.10	1	10	tansig, purelin	traingda	dividerand = {0.7, 0.15, 0.15}	76.0098	73.8696	13.8905	[78.852459, 81.521739]	[76.557377, 84.782609]
2.11	1	10	tansig, purelin	trainbr	dividerand = {0.7, 0.15, 0.15}	79.741	74.5652	164.0429	[83.770492, 73.913043]	[79.672131, 85.869565]
2.12	1	10	tansig, purelin	trainr	dividerand = {0.7, 0.15, 0.15}	76.059	73.1522	449.11	[79.180328, 76.086957]	[75.901639, 81.521739]

Figura 6 - Experiência com variação das funções de treino do ficheiro TrainFiltered.csv

4.3 Variação nas funções de ativação

Passando agora às funções de ativação, como pretendíamos apenas compará-las com a Default, optámos por não adicionar ou remover camadas.

Verificamos, mais uma vez que as funções ‘hardlim’, ‘logsig’ e ‘softmax’ não são adequadas à camada de output, junta-se a estas a função ‘hardlims’, estas 4 obtiveram resultados muito abaixo do resto, para além disso as três primeiras apresentaram resultados muito duvidosos, que indicam que todas as iterações tiveram precisão global de exatamente 40%, podendo indicar isto, que estas funções não são adequadas a este tipo de problema.

De resto as combinações testadas não aparentam superar a Default, uma vez que, tirando os resultados que foram praticamente iguais, outras, como a ‘compet’, ‘purelin’ apenas o pioraram bastante.

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Divisão de Exemplos	Media Precisão Global	Media Precisão De Teste	Duração (50 Iterações)	Methor Rede (Precisao Global) [Global, Teste]	Methor Rede (Precisao Teste) [Global, Teste]
Funcão de ativação										
3.1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.2525	74.2826	9.0781	[80.163934, 71.739130]	[80.163934, 71.739130]
3.2	1	10	tansig, logsig	trainlm	dividerand = {0.7, 0.15, 0.15}	40	40.1739	9.6417	[40.000000, 39.130435]	[40.000000, 51.086957]
3.3	1	10	tansig, hardlims	trainlm	dividerand = {0.7, 0.15, 0.15}	50.4852	49.413	6.8739	[65.409836, 56.521739]	[62.622951, 71.739130]
3.4	1	10	tansig, softmax	trainlm	dividerand = {0.7, 0.15, 0.15}	40	39.7826	8.9257	[40.000000, 36.956522]	[40.000000, 51.086957]
3.5	1	10	tansig, hardlim	trainlm	dividerand = {0.7, 0.15, 0.15}	40	40.3478	6.8722	[40.000000, 36.956522]	[40.000000, 52.173913]
3.6	1	10	radbasn,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	76.9574	72.5435	8.8658	[80.491803, 77.173913]	[80.000000, 82.608696]
3.7	1	10	poslin,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.8623	73.087	9.2224	[81.147541, 75.000000]	[81.147541, 75.000000]
3.8	1	10	radbas, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.2918	72.087	9.6818	[81.147541, 69.565217]	[77.049180, 84.782609]
3.9	1	10	purelin,tansig	trainlm	dividerand = {0.7, 0.15, 0.15}	76.2951	75.6304	8.2089	[77.704918, 75.000000]	[76.557377, 86.956522]
3.10	1	10	hardlim, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	67.2262	65.4565	7.0495	[72.295082, 69.565217]	[69.344262, 72.826087]
3.11	1	10	tansig,tansig	trainlm	dividerand = {0.7, 0.15, 0.15}	77.4557	72.8261	9.3806	[83.606557, 69.565217]	[79.344262, 80.434783]
3.12	1	10	tribas,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.1574	72.4565	6.4041	[82.786885, 75.000000]	[79.344262, 81.521739]
3.13	1	10	compet,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	64.4852	63.5652	3.6058	[69.672131, 63.043478]	[65.409836, 76.086957]
3.14	1	10	purelin,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	76.5705	75.7391	12.9104	[77.704918, 73.913043]	[77.540984, 83.695652]
3.15	1	10	satlin,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.3344	73.6957	5.946	[82.459016, 67.391304]	[78.032787, 83.695652]
3.16	1	10	netinv,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	61.9016	58.587	7.2373	[67.868852, 65.217391]	[60.491803, 70.652174
3.17	1	10	softmax,purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	77.0689	73.2391	5.1351	[83.278689, 69.565217]	[77.213115, 82.608696]

Figura 7 - Experiência com variação das funções de ativação do ficheiro TrainFiltered.csv

4.4 Variação na segmentação

No que toca à segmentação usámos funções, ‘dividerand’, ‘divideint’ e sem segmentação.

Estas produziram resultados variados influenciados também pelas ratios aplicadas.

Notámos que ao ter menos de treino do que de validação / teste, os resultados foram afetados negativamente, este resultado foi o que esperávamos e a experiência serviu apenas de confirmação a esta teoria.

Notámos também que, usar divideint ou dividerand não aparenta afetar muito o resultado, no entanto, em casos com percentagens de treino mais elevada, a primeira (divideint) demonstrou resultados melhores, o inverso foi verificado quando as percentagens para treino eram mais reduzidas, mesmo assim, estes são resultados relativamente semelhantes, o que não nos permitiram obter conclusões.

No que toca à sem segmentação não podemos obter uma precisão de Teste, já que todos os exemplos são usados no treino, esta demonstrou resultados extremamente bons no que toca à precisão Global, no entanto, como visto anteriormente, estes resultados podem indicar que a rede apenas está demasiado habituada ao Dataset, e não saberia se adaptar a um novo, como o exemplo do Test.csv

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Divisão de Exemplos	Media Precisão Global	Media Precisão De Teste	Duração (50 Iterações)	Melhor Rede (Precisao Global) [Global, Teste]	Melhor Rede (Precisao Teste) [Global, Teste]
					Divisão de Exemplos					
4.1	1	10	tansig, purelin	trainlm	dividerand = [0.20, 0.40, 0.40]	72.9607	70.8934	4.7014	[77.213115, 76.229508]	[75.245902, 79.508197]
4.2	1	10	tansig, purelin	trainlm	dividerand = [0.9, 0.05, 0.05]	77.3148	72.7742	4.5681	[83.606557, 74.193548]	[79.508197, 96.774194]
4.3	1	10	tansig, purelin	trainlm	dividerand =[0.75, 0.125, 0.125]	77.8033	73.7895	3.8774	[82.459016, 72.368421]	[77.377049, 86.842105]
4.4	1	10	tansig, purelin	trainlm	dividerand = [0.4, 0.3, 0.3]	75.8164	71.8579	5.0418	[78.852459, 74.316940]	[78.852459, 79.781421]
4.5	1	10	tansig, purelin	trainlm	divideint = [0.4, 0.3, 0.3]	76.1934	74.5355	4.9697	[78.852459, 77.049180]	[76.065574, 79.234973]
4.6	1	10	tansig, purelin	trainlm	divideint = [0.6, 0.2, 0.2]	75.9705	77.2787	5.0281	[81.147541, 79.508197]	[76.721311, 85.245902]
4.7	1	10	tansig, purelin	trainlm	divideint = [0.75, 0.125, 0.125]	77.7541	74.8421	4.0443	[82.786885, 72.368421]	[81.147541, 80.263158]
4.8	1	10	tansig, purelin	trainlm	divideint = [0.9, 0.05, 0.05]	78.6492	76.3226	3.9857	[84.918033, 74.193548]	[81.475410, 83.870968]
4.9	1	10	tansig, purelin	trainlm	divideint = [0.7, 0.15, 0.15]	78.1967	74.2174	3.9894	[81.639344, 72.826087]	[77.377049, 78.260870]
4.10	1	10	tansig, purelin	trainlm	Sem Segmentação	88.0918		84.3084	[90.983607, NaN]	
4.11	1	10	tansig, purelin	trainlm	dividerand = [0.6, 0.2, 0.2]	77.1508	73.5246	3.9828	[81.147541, 78.688525]	[80.000000, 82.786885]

Figura 8 - Experiência com variação na segmentação do ficheiro TrainFiltered.csv

4.5 Variação de todas as configurações

Por fim, com o objetivo de obter as melhores redes, decidimos variar, podemos afirmar que obtemos resultados relativamente bons no que toca à precisão global, com a exceção do 5.8.
Usámos as configurações melhores que vimos nos pontos anteriores, assim como outras combinações que ficámos curiosos.

Verificamos um decréscimo na precisão de teste em quase todas as redes, o que poderá vir a afetar os seus desempenhos na próxima fase do Trabalho Prático.

Configuração	Nº de Camadas Escondidas	Configuração Neuronios	Funções de Ativação	Função de Treino	Divisão de Exemplos	Media Precisão Global	Media Precisão De Teste	Duração (so Iterações)	Melhor Rede (Precisao Global) [Global, Teste]	Melhor Rede (Precisao Teste) [Global, Teste]
Combinações Multiplas [Tentar Obter os Melhores Resultados]										
5.1	2	50,50	poslin,poslin,purelin	traincgf	dividerand = {0.9, 0.05, 0.05}	79.518	70.6452	39.4228	[87.213115, 74.193548]	[79.836066, 87.096774]
5.2	2	50,50	poslin,poslin,purelin	traincgf	divideint = {0.9, 0.05, 0.05}	79.1213	74.3226	41.4832	[85.409836, 77.419355]	[81.475410, 83.870968]
5.3	2	20, 20	tansig,tansig,tansig	trainlm	divideint = {0.7, 0.15, 0.15}	80.7508	73	18.2199	[87.377049, 69.565217]	[77.049180, 80.434783]
5.4	4	10,10,10,10	logsig (4x), purelin	trainbr	divideint = {0.9, 0.05, 0.05}	96.7115	72.2581	1,727.60	[99.180328, 83.870968]	[99.180328, 83.870968]
5.5	4	10,10,10,10	logsig (4x), purelin	trainbr	divideint = {0.7, 0.15, 0.15}	93.7443	72.3043	836.1515	[97.049180, 80.434783]	[97.049180, 80.434783]
5.6	2	20,20	logsig, logsig, purelin	trainbr	divideint = {0.7, 0.15, 0.15}	84.3967	75.1957	2.72E+03	[96.557377, 80.434783]	[85.573770, 80.434783]
5.7	2	20,20	tansig, tansig, purelin	trainbr	divideint = {0.7, 0.15, 0.15}	82.7475	68.1087	1.28E+03	[97.049180, 80.434783]	[97.049180, 80.434783]
5.8	3	10, 20, 30	logsig, logsig, logsig, purelin	trainbr	divideint = {0.7, 0.15, 0.15}	66.3016	61.7391	842.1757	[96.393443, 76.086957]	[85.409836, 78.260870]
5.9	3	10, 20, 30	poslin, tansig, logsig, purelin	trainlm	divideint = {0.7, 0.15, 0.15}	80.1213	71.8043	35.0516	[87.049180, 70.652174]	[78.524590, 79.347826]
5.10	3	10, 20, 30	poslin, poslin, poslin, purelin	trainbr	divideint = {0.7, 0.15, 0.15}	79.8328	73.9565	2,526.30	[82.295082, 72.826087]	[80.491803, 77.173913]

Figura 9 - Experiência com variação em todas as configurações do ficheiro TrainFiltered.csv

5. Ficheiro TEST.csv

No que toca à seleção de redes, usámos dois critérios, primeiro os valores de precisão global / teste, e segundo a variação destes valores. As redes devem ter valores bons e consistentes, no entanto, optámos por incluir redes cujos valores eram muito bons, como por exemplo a rede global 5.4, que obteve 99% de precisão global, para anotar os seus resultados.

Os resultados foram bastante surpreendentes, uma vez que, ao testar as redes Default, a melhor rede Default no que toca a precisão de teste acertou em 80% dos casos, o melhor de todos as redes selecionadas.

De resto são poucas as conclusões que conseguimos tirar, podemos dizer que, embora redes terem exatamente as mesmas precisões globais e de teste no ficheiro Train.csv, no ficheiro Test.csv estas podem não dar o mesmo resultado, como foi o caso das redes 5.5 (Teste) e 5.7 (Global), que,

embora tenham dado exatamente as mesmas precisões no train.csv, obtiveram resultados diferentes aqui, tendo a 5.7 acertado em 70% dos casos do test.csv, e a 5.5 acertado em apenas 60%. Podemos também afirmar que redes com grande precisão global ou de teste resultam em melhores resultados. Sinceramente, ficámos relativamente surpreendidos.

Adicionalmente, uma pequena variação entre as precisões não assegura necessariamente bons resultados em outros conjuntos de dados, como demonstra a rede 1.5 de Teste, que, apesar de ter apresentado precisões quase idênticas anteriormente, alcançou somente 60% de acertos neste ficheiro.

Suspeitamos que estes resultados possam estar relacionados com a hipótese de o ficheiro Test.csv conter valores para os quais as redes neuronais testadas não estavam devidamente preparadas, por serem substancialmente diferentes dos casos utilizados durante o treino, embora esta suposição não possa ser confirmada com absoluta certeza.

Concluímos, portanto, que, ainda que algumas redes demonstrem um bom desempenho num determinado dataset, podem não atender às expectativas quando aplicadas a datasets distintos.

Configuração	Media Precisão Global	Erro
Default		
Default - Global	70	0.3
Default - Teste	80	0.2
Redes		
5.4 - Global	70	0.3
5.5 - Teste	60	0.4
5.7 - Global	70	0.3
4.10 - Global	70	0.3
1.10 - Teste	70	0.3
5.2 - Teste	60	0.4
1.5 - Teste	60	0.4

Figura 10 - Resultados do Ficheiro Test.csv

6. Aplicação Gráfica

Optámos por aceitar o desafio proposto pela alínea bónus 3.4 d). Para este fim, desenvolvemos uma aplicação que permite ao utilizador criar uma rede neuronal, treiná-la, guardá-la/carregá-la, testá-la, para alem disso permite introduzir os valores dos atributos de apenas um caso e verificar qual a classificação dada pela rede. A aplicação oferece grande liberdade ao utilizador na customização desta rede.

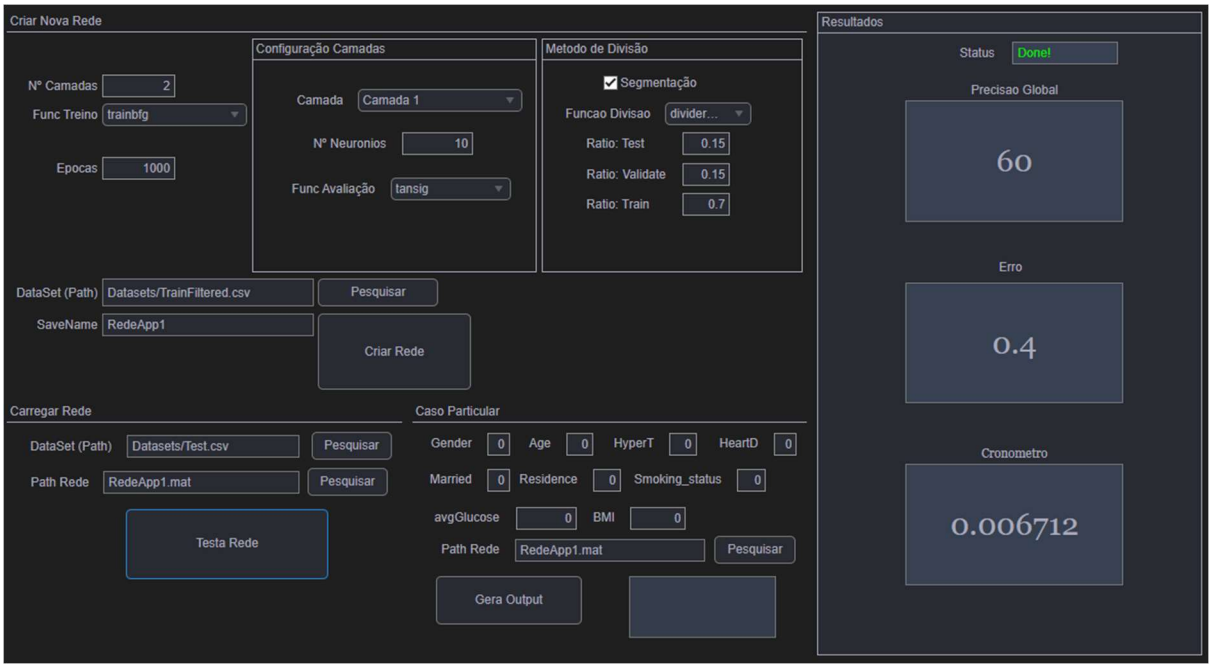

Figura 11 - Interface da aplicação

Começando pelas funções de criação de rede, estas permitem configurar o treino desta, assim como a sua estrutura, o painel 'Configuração Camadas' permite ao utilizador escolher a camada a editar e alterar a sua quantidade de neurónios e a sua função de ativação, isto aplica-se à camada de output, embora nesta não ser possível alterar a quantidade de neurónios.

Os botões "pesquisar" em todos os painéis permitem ao utilizador escolher, através do explorador de ficheiros, o Path para o Dataset / rede neuronal.

Na parte de carregar Rede esta é apenas para o teste, o painel de Precisão Teste (lado direito) é convertido para exibir o erro, uma vez que no teste não existe segmentação do Dataset.

O campo SaveName pode ser deixado vazio, neste caso ele não será guardado.

As funções de treino, avaliação e segmentação (divisão) disponíveis foram baseadas no PDF disponibilizado na Ficha 8 das aulas práticas, funções que causavam problemas foram, no entanto, removidas, já que provavelmente não eram adequadas ao tipo de problema apresentado.

7. Conclusão

Este trabalho prático permitiu-nos aprofundar os nossos conhecimentos sobre Redes Neuronais e CBR. Verificámos que é preciso ter cautela com os parâmetros aplicados no treino e que existem vantagens e desvantagens em certas funções, assim como nem todas as combinações são viáveis. Também aprendemos sobre a importância de um Dataset bem preparado para o CBR e para o desempenho da rede.

Por fim, sentimos que compreendemos um pouco mais sobre o fascinante mundo da Inteligência Artificial, que atualmente recebe grande atenção em todo o mundo e será de suma importância para o nosso futuro, tanto no campo da Informática quanto no dia a dia.

8. Bibliografia

- Moodle ISEC Disciplina de CR – Consultado múltiplas vezes durante a realização do trabalho.
- Documentação MATLAB – Consultada periodicamente para melhor entendimento e esclarecimento de dúvidas e problemas que apareceram ao longo do desenvolvimento do Trabalho

9. Ficheiros em Anexo

- Cbr.m e retrieve.m - Codigo do CBR do trabalho prático.
- DataSorter.m – Script para a preparação do Dataset pre-CBR

- FeedfowardSTART.m / FeedFowardTRAIN.m / FeedfowardTEST.m – Ficheiros de código com as funções respetivas às redes neuronais
- Experiencia.m – Ficheiro auxiliar criado para executar o FeedFowardTRAIN.m de forma mais fácil e intuitiva.
- appGrafica.mlapp – Aplicação gráfica
- Resultados.xlsx – Ficheiro Excel que contem todos os resultados das experiências realizadas
- Datasets (Pasta) – Pasta que contem os Datasets utilizados.
- Redes (Pasta) – Pasta que contem todas as “melhores redes” mencionadas na folha de trainFiltered do ficheiro Excel.
- RedesTest (Pasta) – Pasta que contem copias das redes usadas na folha de test do ficheiro Excel.