Conhecimento e Raciocínio

Trabalho Prático

2023/2024

João Almas | 2021138417 | P5

João Pinto | 2022143707 | P3

Índice

Escolha e Preparação do Dataset

O dataset escolhido foi Cirrose.

Conversão de atributos

Para a conversão dos atributos em valores numéricos ou booleanos foi criada a função MATLAB, “translate.m”.

Este script processa os dados do ficheiro "Train.csv", convertendo atributos categóricos em valores numéricos com base num mapeamento predefinido correspondente à informação do ficheiro “descrição_datasets.pdf”. O processo é realizado célula a célula, começando por verificar se o valor correspondente é categórico. Caso seja, é então verificado se corresponde a um atributo no mapeamento. Se sim, é substituído pelo valor correspondente; caso contrário, é atribuído NaN. Os dados são então organizados numa nova tabela a partir da qual será criado um ficheiro “Train_Treated.csv”, contendo o dataset traduzido e pronto para ser analisado com o cbr para preencher os valores em falta.

Identificação do atributo “target” e aplicação do Raciocínio Baseado em Casos (CBR)

Para preencher os atributos com valores em falta (NaN) no ficheiro "Train_translated.csv", apenas foi implementada a fase de recuperação (retrieve) do sistema de raciocínio baseado em casos (CBR) utilizando a função MATLAB "cbr.m".

Na função, é definido um limiar de similaridade, representado pela variável "similarity_threshold", que determina o grau de semelhança necessário para considerar dois casos como suficientemente similares.

O processo inicia-se com o carregamento do conjunto de dados da biblioteca de casos, que contém os valores previamente traduzidos. Em seguida, é realizado um processo de busca para cada caso que apresenta valor ausente na coluna "Stage". Essa busca compara o caso atual com os demais casos na biblioteca, procurando por casos similares.

No retrieve é usada distância local apenas nas colunas: Status, drug, sex, ascites e edema. Nas restantes colunas é usada a distância linear.

Caso sejam encontrados casos suficientemente similares, o valor de "Stage" do caso atual é preenchido com o valor correspondente ao caso mais similar encontrado. Se nenhum caso similar for encontrado, a função retorna um erro.

Após preencher todos os valores ausentes na coluna "Stage", os dados atualizados são exportados para um novo ficheiro Excel chamado "Treated_Train.csv".

Estudo e análise de Redes Neuronais

Este estudo foi tratado pelo grupo como uma forma de benchmark á funcionalidade da implementação desenvolvida, visto que, ao utilizar todos os exemplos para treino e com um dataset de tamanho reduzido, é expectável que a precisão global seja muito elevada.

Start	N.º de camadas	N.º de Neurónios	Funções de Ativação	Função de treino	Divisão dos exemplos	Precisão	Erro	Tempo de Execução (segundos)
Configuração default	1	10	tansig, purelin	trainlm	(1, 0, 0)	100	1,07E-15	0,0333
Alteração na função de treino								
Configuração N.º1	1	10	tansig, purelin	trainbfg	(1 , 0, 0)	100	1,96E-12	0,5967
Configuração N.º2	1	10	tansig, purelin	trainoss	(1 , 0, 0)	100	1,95E-19	0,2386
Configuração N.º3	1	10	tansig, purelin	trainc	(1 , 0, 0)	100	1,88E-04	7,9103
Configuração N.º4	1	10	tansig, purelin	traingdm	(1 , 0, 0)	90	6,54E-02	0,2704
Alteração na função de ativação								
Configuração N.º5	1	10	logsig, purelin	trainlm	(1 , 0, 0)	100	2,58E-18	0,0306
Configuração N.º6	1	10	purelin, purelin	trainlm	(1 , 0, 0)	100	3,93E-17	0,0276
Configuração N.º7	1	10	logsig, logsig	trainlm	(1 , 0, 0)	100	1,88E-01	0,069
Configuração N.º8	1	10	satlin, purelin	trainlm	(1 , 0, 0)	100	5,88E-23	0,0339

(os valores de erro para configurações com funções lineares não estão balizados, o que faz com que se torne inconclusivo)

Conclusões das alterações na função de treino:

- Configuração N.º 1: A precisão é alta (100%), o que quer dizer que tudo parece estar nos conformes, e que as redes têm a capacidade de aprender. O mesmo aplica-se para as configurações N.ºs 2 e 3.
- Configuração N.º 4: Aqui, a precisão cai para 90%, e o erro é relativamente alto (6.54E-02), indicando que a função traingdm possivelmente não será uma boa opção para treinar redes futuramente.

Conclusões das alterações na função de ativação:

- Configuração N.º 5, Nº 6 e Nº 8: Tal como as primeiras configurações apresentam altas precisões, independentemente da configuração.
- Configuração N.º 7: A precisão é alta (100%) porem, o erro é relativamente alto (0.188), o que é inesperado, visto que seria de esperar que com a função de sigmoide como a logsig tivesse melhores resultados num problema nos quais os targets são binário visto que foram codificados em one hot encoding.

Train

Train_Treated	N.º de camadas	N.º de Neurónios	Funções de Ativação	Função de treino	Divisão dos exemplos	Epochs	Learning Rate	Precisão global	Precisão de teste
Configuração Nº 1	1	10	tansig, purelin	trainscg	(0.75, 0, 15)	1000	0,01	74.509803	45.39215
Configuração Nº 2	1	10	tansig, purelin	traincgf	(0.75, 0, 15)	1000	0,01	69.852941	44.362745
Configuração Nº 3	1	10	tansig, purelin	traincgp	(0.75, 0, 15)	1000	0,01	74.019607	44.068627

Explicação:

Iniciaram-se as experiências no dataset train ao comparar diversas funções de treino (ficheiro result_PreDatasetChanges) sendo as três configurações acima as que tiveram melhor precisão. Estes resultados não são muito promissores, ainda assim estas redes tiveram isoladamente:

Train_Treated	Precisão global da melhor rede	Precisão da melhor rede na fase de teste
Configuração Nº 1	75.490196	61.764705
Configuração Nº 2	73.039215	60.294117
Configuração Nº 3	73.774509	58.823529

Concluido:

A diferença de funções de treino não mudou drasticamente os resultados das redes, pelo que é difícil concluir algo conciso. Visto que os resultados são de baixa precisão, não foram guardadas nenhumas redes.

Train_BetterRetrieve	N.º de camadas	N.º de Neurónios	Funções de Ativação	Função de treino	Divisão dos exemplos	Epochs	Learning Rate	Precisão global	Precisão de teste
Configuração Nº 1	1	10	tansig, purelin	trainlm	(0.75, 0, 0.15)	1000	0,01	75.245098	43.774509
Configuração Nº 2	1	10	tansig, purelin	traincgf	(0.75, 0, 0.15)	1000	0,01	72.794117	44.901960
Configuração Nº 3	1	10	tansig, purelin	traincgb	(0.75, 0, 0.15)	1000	0,01	73.039215	44.117647

Explicação:

Depois da performance anterior, pôs-se a proposta de o retrieve não estar bem calibrado e adequado para o dataset, então, depois de aprofundar o conhecimento do dataset e o problema em si, foi produzido um novo retrieve mais compreensivo, para melhorar a qualidade do dataset transformado. No retrieve foram mudadas as similaridades locais dos status e edema, e foram mudados os pesos dos atributos. Estas experiências usam as mesmas configurações das experiências acima (ficheiro result_BetterRetrieve).

Train_BetterRetrieve	Precisão global da melhor rede	Precisão da melhor rede na fase de teste
Configuração Nº 1	80.147058	57.352941
Configuração Nº 2	73.284313	55.882352
Configuração Nº 3	79.166666	54.411764

Concluido:

Tal como nas experiências anteriores, a mudança das funções de treino não demonstrou nenhuma melhoria de resultados, sendo que a experiência focal deste segmento também não foi de grandes ganhos, levando a crer que o problema não consta diretamente no retrieve. Foi também concluído que a função trainc demora demasiado tempo para resultados medíocres, pelo que foi decidido não a voltar a utilizar.

Train_Balanced	N.º de camadas	N.º de Neurónios	Funções de Ativação	Função de treino	Divisão dos exemplos	Epochs	Learning Rate	Precisão global	Precisão de teste
Configuração Nº 1	1	10	tansig, purelin	trainlm	(0.8, 0, 0.15)	1000	0,01	95.588235	40.30303
Configuração Nº 2	1	10	tansig, purelin	traingdx	(0.85, 0,0.15)	1000	0,01	91.176470	43
Configuração Nº 3	1	10	tansig, purelin	traingdx	(0.8, 0, 0.15)	1000	0,01	89.705882	43.63636
Adicionar alterações nas arquiteturas de rede e mais valores de divisão									
Configuração Nº 4	2	10 10	tansig, tansig, purelin	trainrp	(0.8, 0, 0.15)	1000	0,01	94.11764	44.242424
Configuração Nº 5	2	10 10	tansig, tansig, purelin	traincgp	(0.8, 0, 0.15)	1000	0,01	94.11764	43.333333
Configuração Nº 6	2	10 10	tansig, tansig, purelin	traincgf	(0.6, 0, 0.15)	1000	0,01	89.70588	45.238095

Explicação:

Com alguma análise ao dataset, veio-se a descobrir que este só tinha 17 casos com stage 1, o que definitivamente causa um desiquilibrio num dataset com 408 entradas, visto isto, tentou-se por undersampling, criar um dataset mais equilibrado, e fazer modificações nas funções de treino, como anteriormente e poucas variações nos valores de divisão aleatória, decidiu-se ainda, testar ao mesmo tempo outra arquitetura de rede e variar os valores de divisão ainda mais (ficheiro result_DatasetChanges).

Concluido:

Como resultado destas experiências, começaram-se a ver resultados melhores, pelo menos na precisão global o que quer dizer, sendo que a média de teste se mantém relativamente estagnada, o que significa que o equilíbrio do dataset foi uma boa abordagem, mas existem outros fatores que pioram os resultados dos testes.

Train_Balanced	N.º de camadas	N.º de Neurónios	Funções de Ativação	Função de treino	Divisão dos exemplos	Epochs	Learning Rate	Precisão global	Precisão de teste
Configuração Nº 1	1	10	tansig, purelin	trainscg	(0.8, 0, 0.1)	1000	0,01	97.058823	42.916666
Configuração Nº 2	1	10	tansig, purelin	trainscg	(0.7, 0, 0.1)	1000	0,01	92.647058	41.4814814
Configuração Nº 3	1	10	tansig, purelin	trainscg	(0.5, 0, 0.1)	1000	0,01	92.647058	39.0909090
Adicionar alterações na learning rate									
Configuração Nº 4	1	10	tansig, purelin	trainscg	(0.75, 0,0.25)	1000	0.005	89.705882	38.8235294
Configuração Nº 5	1	10	tansig, purelin	trainscg	(0.75, 0,0.25)	1000	0,0001	86.764705	40
Configuração Nº 6	1	10	tansig, purelin	trainscg	(0.75, 0,0.25)	1000	0,015	86.764705	37.0588235

Explicação:

Depois de analisar os resultados anteriores, pareceu estar a acontecer overfitting, visto que as precisões globais eram altas e as de teste baixas, pelo que, se escolheu a função de treino que originou a maior precisão de teste, e decidiu-se testar com diversas variações no train ratio, de modo a verificar se a sua diminuição causaria o aumento da precisão de teste. Depois disso, foi selecionada a média dos dois train ratio's e foram experimentados valores de learning rate diferentes, de modo a perceber se levaria a alguma melhoria nas precisões de teste. (ficheiro result_DatasetChanges2).

Concluido:

O que se concluiu a partir destes experimentos foi que altos train rate's levaram a melhores resultados na precisão global e no treino, o que leva a acreditar que o fenómeno que está a levar a resultados maus não se trata de overfitting. O teste do learning rate indica que melhores resultados foram obtidos pelos leaning rates mais baixos, ainda que , a diferença não seja significativa.

Explicação

Foram ainda testados os múltiplos datasets com funções que beneficiam redes que são treinadas para targets one hot encoded, sigmoid's que retornam valores binários, mas estas ficaram na maioria dos casos classificadas abaixo da purelin, utilizada para comparação. (ficheiros result_SigmoidTests, result_ManyDatasetsLogsigtest e ManyDatasetsLogsigtest)

Concluido

Concluiu-se que, ainda teoricamente, estas funções devessem apresentar melhores resultados, a função que foi mais utilizada até agora, purelin, produz melhores resultados, sendo as redes que foram guardadas de origem no ManyDatasetsLogsigtest2, todas a utilizar o purelin, mas sem grandes precisões no dataset de teste(máximo de 60).

Teste

Dataset do qual foi gerada a rede durante o seu treino	Resultado obtido
Train_Treated	50
Train_Normalized	30
Train_Balanced	60
Train_Oversampling	40
Train_noNaN	50

Conclusão

Os resultados obtidos refletem consistentemente um desempenho moderado, com nenhum resultado a ultrapassar o limite de 60. Esta falta de desempenho mais elevado era antecipada, dada a fraca qualidade do dataset Train.

Conclusões Finais

Apesar das diversas abordagens para tentar obter melhores resultados, não foi possível obter resultados promissores, pelo que acreditamos que o problema é o conjunto de dados em si, que parece ser de baixa qualidade, desequilibrado e inconclusivo.