Computação Natural

MEI (4º Ano)

Trabalho Prático Nº 1)

Relatório de Desenvolvimento

Hugo da Giuiao	José Ramos	Nuno Reis
(pg41073)	(a73855)	(a77310)

14 de Abril de 2020

Resumo

Este trabalho prático tem como principal objectivo o aprofundamento e aplicação dos conteúdos e competências adquiridas nas aulas práticas e teóricas da unidade curricular de *Computação Natural*. Além de procurar desenvolver o conhecimento relacionado com o funcionamento de *Rede Neuronal Artificial* e a capacidade de as implementar com linguagem *Python*.
Neste caso em particular o objecto de estudo trata-se de um conjunto de dados recolhidos de exames de massas em *mamografia* incitando a exploração d casos de ocorrência de tumor para se poder prever com maior eficácia e rapidez o diagnostico da massa.
No presente relatório expomos os resultados obtidos, bem como a especificação desenvolvida para alcançar objectivos requeridos no enunciado deste trabalho.

Conteúdo

Capítulo 1

Introdução

Área: Computação Natural

1.1 Caso de estudo

No enunciado prático foi nos pedido a implementação de um modelo de previsão baseado em *Redes Neuronais Artificiais* para classificar, num exame de massas mamário, se estas são benignas ou malignas. O *dataset* fornecido para estudo apresenta um conjunto de dados reais, obtidos em casos clínicos, que nos dão a informação e valores de vários factores de predição usados pelos responsáveis medico aquando a asserção da classificação do tumor. Para a implementação deste trabalho optamos pelo uso da linguagem *Python*, bem como a ferramenta *TensorFlow*, aconselhada pelo docente.

1.2 Descrição informal do problema

Os requisitos fundamentais deste projeto são:

- Desenvolvimento de modelo de Inteligência Artificial que recebendo os dados clínicos seja capaz de classificar o respetivo tumor como benigno ou maligno.
- Aplicação de técnicas de *Rede Neuronal Artificial* no modelo.
- Recorrer a *Algoritmo Genético* para garantir variação e melhorar eficiência de cada iteração de treino da rede neuronal.
- Apresentar uma accuracy que satisfaça o objectivo do projeto.

1.3 Estrutura do Relatório

Este relatório encontra-se organizado em cinco capítulos.

O primeiro capitulo expõe o conceito do projeto e a organização do presente relatório.
No capítulo 2, é feita uma descrição do problema proposto, e uma breve análise do *dataset* fornecido.
No capítulo 3 é abordado o processo de desenvolvimento incluindo decisões tomadas na criação do modelo de *Rede Neuronal Artificial*.
No capítulo 4 expomos técnicas adicionais a que recorremos para garantir variação dos neurónios, bem os resultados finais obtidos
Por fim, no capítulo 5 temos a síntese de todo o trabalho realizado, assim como uma análise crítica dos resultados obtidos, de problemas encontrados e do possível trabalho futuro a desenvolver.

Capítulo 2

Análise e Processamento de dados

2.1 Análise de Features

O processo de criação do modelo de previsão começa com a visualização dos dados contidos no dataset fornecido. Para tal recorremos à bibioteca pandas, carregando a informação para um objeto dataframe.

Bi-Rads - "Breast Imaging Reporting and Data System" 1-5 (ordinal);
Os médicos recorrem a um sistema padrão para descrever os dados e resultados de uma mamografia. Ao classificar os resultados nessas categorias, os médicos podem descrever o que encontram em uma mamografia usando as mesmas palavras e termos. Isso facilita muito a comunicação sobre esses resultados e o acompanhamento após os testes.
Contudo esta feature não será usada para treino uma vez que já apresenta um carácter preditivo.

Age - Idade do paciente contabilizada em anos (inteiro);
A possibilidade de ter cancro da mama aumenta com o aumento da idade; uma mulher com mais de 60 anos apresenta maior risco, e o cancro da mama é menos comum antes da menopausa. Contudo a idade não e um factor exclusivo, uma vez que factores genéticos tamb'em influenciam a incidência.

Shape - A forma do corpo é categorizada em quatro grupos:
Forma da massa: redonda = 1, oval = 2, lobular = 3, irregular = 4 (nominal);

Margin - A margem do corpo é categorizada em cinco grupos:
Margem da massa: circunscrita = 1, micro-lobada = 2, obscurecida = 3, mal definida = 4, cravada = 5 (nominal);

Density - A densidade do corpo é categorizada em quatro grupos:
Densidade: densidade de massa alta = 1 iso = 2 baixa = 3 contendo gordura = 4 (ordinal);
Os relatórios de mamografia também incluem uma avaliação da densidade das mamas. Seios densos têm menos tecido adiposo e mais tecido não adiposo em comparação com os que não são densos.

Severity - A gravidade da massa é categorizada em dois grupos:
Gravidade: benigna = 0, maligna = 1 (binominal);

2.2 Tratamento dos dados

Quando ocorreu o carregamento dos dados, todos os valores em falta foram substituídos pelo caractere '?'. Com recurso ao um *heatmap* conseguimos averiguar a sua distribuição, descobrindo que esta era aleatória. Com este conhecimento decidimos remover todos este valores do *dataset*, recorrendo à função *dropna*.

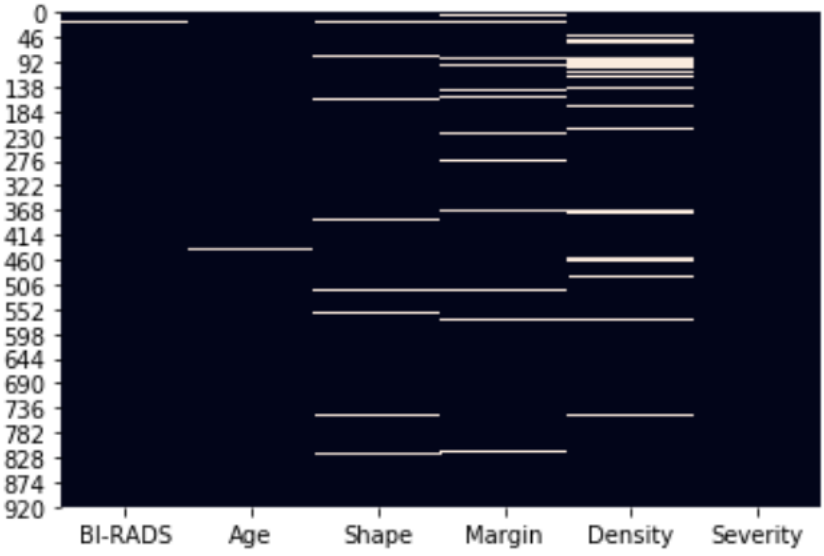

Figura 2.1: Heatmap de missing values

De seguida o grupo procedeu à separação dos dados em features e labels, sendo a ultima a *Severity*. Foi ainda necessário a redefinição dos índices uma vez que algumas instâncias tiveram de ser eliminadas quando se lidou com os valores em falta.

Por fim, para garantir uma distribuição de importância equitativa entre as features do *dataset* decidimos trabalhar sobre a gama de valores possíveis dos dados de input, normalizando todas as features. Este tratamento serviu para garantir que não houvessem valores maiores a se tornarem dominantes perante inputs menores durante o treino da *Rede Neuronal Artificial*. Para tal, foi usado o *StandardScaler* para normalizar o conjunto de features.

Capítulo 3

Concepção da *Rede Neuronal Artificial*

3.1 Criação do Modelo

Com recurso á biblioteca de *TensorFlow* o grupo decidiu recorrer ao classificador *DNNClassifier* para nos basearmos a estruturar a criação do modelo.

```
classifier = tf.estimator.DNNClassifier(hidden_units=h_u,
                                        n_classes=2,
                                        feature_columns=feat_cols,
                                        activation_fn=a_f,
                                        dropout=0.5,
                                        optimizer=tf.train.AdamOptimizer(
                                           learning_rate=network.network[0]
                                        ))
```

Figura 3.1: *DNNClassifier*

Analisando os parâmetros do nosso *classifier*, numa primeira implementação do modelo criou-se uma RNA. Cujos parâmetros são retirados de uma estrutura que gera valores aleatorios, que se encontram dentro de intervalos adequados a cada parametro. O argumento *feature_columns* trata-se de um array com os nomes das features num formato reconhecido pelo *TensorFlow*. A função de activação escolhida foi uma aleatoria, guardada na variavel a_f.
Para estratégia de optimização, recorremos ao optimizador *AdamOptimizer* que, em pesquisa efectuada apresentou melhor capacidade em modelos semelhantes, como se observou por pesquisas do grupo. Ao argumento *learning_rate* foi atribuído um valor aleatório a cada *network* treinada . Por fim, utilizou-se um *dropout* de 0.5, de maneira a adicionar um factor de probabilidade de activação ou desactivação aos neurónios da rede tal que fosse possível o aparecimento de caminhos alternativos ao longo da rede, evitando o overfitting do modelo.

3.2 Treino e Avaliação

Com o intuito de garantir que o modelo desenvolvido se apresenta preparado para qualquer tipo de combinação de dados que lhe seja entregue para avaliação o grupo optou por recorrer a uma técnica de *cross-validation* com dez *folds*. Através da biblioteca Kfold do sklearn, geramos automaticamente os índices de divisões do *dataset* tendo sido necessário iterar pelo objecto do Kfold dez vezes e definir os dados de treino e teste, para proceder à validação do modelo.
Em adição foi decidido que o factor através do qual mediríamos a eficiência do modelo criado seria a *accuracy* media dos k-folds.
No final da criação do modelo de *Rede Neuronal Artificial* serão apresentados as cinco melhores configurações, mostrando a *accuracy* de cada uma.
Como objectivo o grupo pretende alcançar um valor adequado ao problema. Sendo este um problema e carácter de saúde, podendo influenciar a vida de pessoas, o nosso objetivo final seria os 100% de *accuracy*, contudo devido a dimensão do projeto e do dataset fornecido pretendemos alcançar pelp menos os 80%.

Capítulo 4

Optimização de parâmetros

O primeiro modelo que implementamos foi capaz de cumprir o objectivo funcional de calcular o valor de Severity embora não tivesse sido capaz de apresentar valores de *accuracy* que desejávamos alcançar. Com isto em mente decidimos tentar alcançar uma *accuracy* maior, para tal procuramos optimizar os hiperparâmetros. A estratégia principal do processo de optimização foi implementar conceitos de algoritmos genéticos na nossa *Rede Neuronal Artificial* no qual, cada *network* representa um setup de configuração de um modelo de *RNA* e cada *pool* representa os hiperparâmetro dessa configuração. Cada *pool* esta definida com quatro elementos, o primeiro corresponde à variável learning_rate, o segundo número de nodos por cada camada intermédia, o terceiro ao número de camadas intermédias e o último é a função de ativação. A função de fitness será finalmente representada através da *accuracy* obtida.

4.1 Funções Principais

Em baixo apresentamos, algumas funções a que recorremos para a implementação do nosso *Algoritmo Genético*:

- *create_population* : Função é responsável por gerar a população inicial. Cria um conjunto de dez com pools aleatórias, tomando em atenção um intervalo de valores adequado a cada hiperparâmetro.
- *update_classifier_parameters*: Função recebe como argumento uma determinada *network* mapeando os valores de pool dela na criação de um modelo.
- *fitness* : Função que retorna o valor da *accuracy* de uma dada *network*;
- *breed*: Esta função recebe como argumento a estrutura que contem a informação das *network* de uma dada geração. O processo passa por gerar novas *network* baseando-se em características de outros duas *network*, uma pai e uma mãe seleccionadas da geração anterior.
- *mutate*: Função que permite introduzir diversidade. Através disto, foi-nos possível prevenir algo que se apresentava a ocorrer no treino da nossa *RNA* e que estaria a causar abrandamento da evolução. Esta função altera aleatoriamente, dentro dos parâmetros estabelecidos, uma das características da *pool* da *network*.
- *evolve*: Esta é a função principal do *Algoritmo Genético*. Através dela actuamos sobre uma população de networks com o *Algoritmo Genético*, preparando os parâmetros desta através de selected cross-breeding e mutation. O resultado será uma nova população que apresenta os melhores dados da anterior, sendo estes trabalhados com recurso ás funções *breed* e *mutate*, tal que permita garantir resultados diferentes, e em principio, melhores que da geração anterior.

4.2 Algoritmo de Treino

O processo de treino da do modelo *Rede Neuronal Artificial* inicia-se com criação de uma população através da função nn_param_choices que decide de uma lista de possibilidades aleatoriamente os parâmetros de uma da rede.
O algoritmo corre iterativamente durante as gerações decididas previamente na respetiva variável.

Para cada *network* de uma geração:

Como metade da população da nova geração são os *network pool* da geração anterior, é possível acelerar o processo verificando se o *network* em questão já foi avaliado anteriormente e nesse caso é apenas necessário obter o resultado do modelo criado na geração anterior;

Caso isso não se verifique, estamos perante uma nova *network* e terá de ser avaliado através da função classify_create_folds que para um dado *network* actualiza o modelo, treina e valida com base na construção do modelo anteriormente mencionada, retornando a *accuracy* do modelo;

Guarda-se a melhor prestação de cada geração para analisar posterior- mente a evolução;

Selecciona-se os melhores cromossomas para "pais"da nova geração, recorrendo ao uso da função *grade*;

Aplica-se o processo de *crossover*, através da função *breed*;

Aplica-se o processo de mutação, através da função *mutation*;

Gera-se a nova população, agregando os "pais"e os "filhos"através da função *evolve*;

Finalmente, obtém-se o valor máximo de *fitness* da última geração e a respectiva configuração das *network*.

4.3 Resultados

Quando corremos o programa desenvolvido é-nos possível verificar uma evolução da *accuracy* ao longo das gerações. Isto deve-se claramente ao efeito do *Algoritmo Genético* implementado. Inicialmente podemos ver que a *accuracy* alcançada esta na ordem dos 76% e ao longo das gerações esta vai melhorando ate que alcançamos 80,7%. No âmbito do dataset e dos dados fornecidos, consideramos este um bom resultado. Em baixo apresentamos o resultado dos melhores 5 resultados, de configuração de hiperparâmetros calculada com *Algoritmo Genético* e com a configuração de 20 gerações de treino e 5 layers base para a nossa *RNA*.

```
Network accuracy: 80.72%
Network accuracy: 79.52%
Network accuracy: 79.52%
Network accuracy: 78.31%
Network accuracy: 78.31%
```

Figura 4.1: *Resultados*

O grupo tentou ainda implementar outras configurações para a *Rede Neuronal Artificial*, contudo não conseguiu obter resultados de *accuracy* melhores que os apresentados em cima. Experimentamos com números de layers diferente, gerações e outros parâmetros em termos de intervalos aquando se recorre a função mutate do *Algoritmo Genético*.
Por fim, observando com atenção a evolução de valores obtidos em vários testes da rede que uma configuração com maior número de gerações de treino não traria grandes benefícios, e aumentaria muito o custo de treino desta rede. Em certas ocasiões era possível verificar que, a partir da décima geração não havia melhoria dos resultados de *accuracy*.

Capítulo 5

Conclusão

Através da utilização dos recursos anteriormente mencionados, fomos capazes de aprimorar não só os nossos conhecimentos de desenvolvimento de *Redes Neuronais Artificiais*, adquiridos na presente unidade curricular, bem como as restantes ferramentas e técnicas como , que implementamos com o intuito de melhorar a eficiência do modelo criado.
A proposta de uso de algoritmos genéticos foi inicialmente complicada de implementar. Acima de tudo foi preciso investigação para conseguirmos entender como organizar o processo para implementar num modelo de forma eficiente ao desafio em causa. Tal que a função que se liga ao classifier foi explorada com mais atenção. Com o devido esforço grupo conseguiu ainda compreender as situações em que se deve aplicar os diversos parâmetros relacionados bem como foi possível ter noção da carga computacional exigida com o aumento da complexidade de problemas.
Com o trabalho concluído sentimo-nos satisfeitos, consideramos que o desafio se mostrou interessante, uma vez que mesmo tendo sido fornecidas todas as bases para a resolução do problema inicial com o anseio por melhorar e aprimorar o resultado final surgiram novos desafios que, com esforço e organização conseguiram ser superados.