Relatório Intercalar

Redes Neuronais para a identificação de Pulsares

Mestrado Integrado em Engenharia Informática e Computação

IART - Inteligência Artificial

Turma 3MIEIC02, Grupo E1_3 :
André Miguel Ferreira da Cruz - 201503776
Edgar Filipe Amorim Gomes Carneiro - 201503784
João Filipe Lopes de Carvalho - 201504875

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

8 de Abril de 2018

Conteúdo

1 Objetivo

Este trabalho tem como objetivo a aplicação de Redes Neuronais artificiais na identificação de Pulsares. Neste sentido, começamos por expor e explicitar o tema, com uma pequena introdução à tarefa em questão, e às dificuldades à sua resolução (secção 1). De seguida, analisamos o *dataset* em uso, descrevendo características e possíveis tendências (*bias*) indesejadas (secção 2.1.1). No mesmo sentido, explicamos o pré-processamento necessário (secção 2.1.2), bem como o modelo de aprendizagem a aplicar (secção 2.1.3) e a sua arquitetura (2.1.4). Finalmente, expomos o trabalho efetuado até ao momento (secção 2.2), os resultados esperados, e as métricas usadas para avaliação (secção 2.3). Para concluir, extraímos algumas conclusões dos resultados obtidos, e traçamos metas para trabalho futuro.

Os pulsares são um tipo de estrela de neutrões raro que, devido ao seu intenso campo magnético, transforma energia cinética (rotacional) em energia eletromagnética. Além disso, o seu campo magnético é suficientemente forte para arrancar partículas da sua superfície (na sua maioria eletrões), que são depois aceleradas na *magnetosfera* e emitidas sob a forma de um feixe estreito e contínuo. Este feixe de radiação, juntamente com a contínua rotação do pulsar, gera um sinal característico, análogo ao de um farol em rotação, de intensidade suficiente para ser detetado a milhões de anos-luz de distância. Desta forma, a deteção de um pulsar corresponde à procura e reconhecimento dos seus sinais periódicos.

Cada pulsar produz um padrão próprio, distinto dos restantes corpos celestes, e que varia ligeiramente entre diferentes pulsares. Assim, podemos detetar um sinal realizando uma análise de alguns parâmetros, como por exemplo o tempo médio que um pulsar demora a reemitir o seu feixe para a terra. À primeira vista, concluir se um sinal é realmente proveniente de um pulsar pode parecer pouco complexo. No entanto, devido a fortes interferências de frequências de rádio e de sinais ruidosos que dificultam a receção de sinais genuínos, a grande maioria das supostas deteções são falsas.

Devido às dificuldades referidas anteriormente, e ao grande volume de dados em causa (induzido maioritariamente por falsas deteções), esta tarefa torna-se extenuante e dispendiosa para humanos, tornando-a uma muito boa candidata para a aplicação de técnicas de *Machine Learning*. Os algoritmos de *machine learning* funcionam através da construção de um modelo a partir de inputs amostrais, com a finalidade de fazer previsões ou decisões orientadas pelos dados e respetivas estatísticas, ao invés de seguir estruturas de decisão estáticas.

Assim sendo, neste trabalho iremos usar algoritmos de aprendizagem supervisionada, nomeadamente redes neuronais artificias, como forma de resolução do problema de classificação apresentado.

2 Descrição

Esta secção comporta a descrição dos dados usados, a explicitação da etapa de pré-processamento dos dados, bem como uma explicação mais detalhada do modelo de aprendizagem a aplicar. Por fim, é descrito o trabalho efetuado até ao momento, os resultados obtidos, as métricas usadas para a avaliação, bem como futuro trabalho no sentido de melhorar o desempenho do modelo.

2.1 Especificação

2.1.1 Descrição e análise do dataset.

Nesta secção, é feita uma análise do *dataset* em uso, especificando o seu conteúdo e determinando as suas características e possíveis tendências (*bias*) indesejadas.

Na base de dados fornecida, cada candidato é caraterizado por oito variáveis continuas e uma classe. As oito variáveis contínuas distinguem-se em dois grupos, sendo as primeiras quatro referentes a estatísticas relativas ao *folded profile* de um pulsar, e as restantes quatro relativas às características da curva *DM-SNR* (*Delta Modulation with Signal-to-Noise Ratio*). O primeiro grupo consiste num *array* de variáveis contínuas que descrevem uma versão pós-análise do pulsar. E as variáveis do segundo grupo são obtidas através da análise da curva *DM-SNR*. Esta consiste na modulação de um sinal analógico para

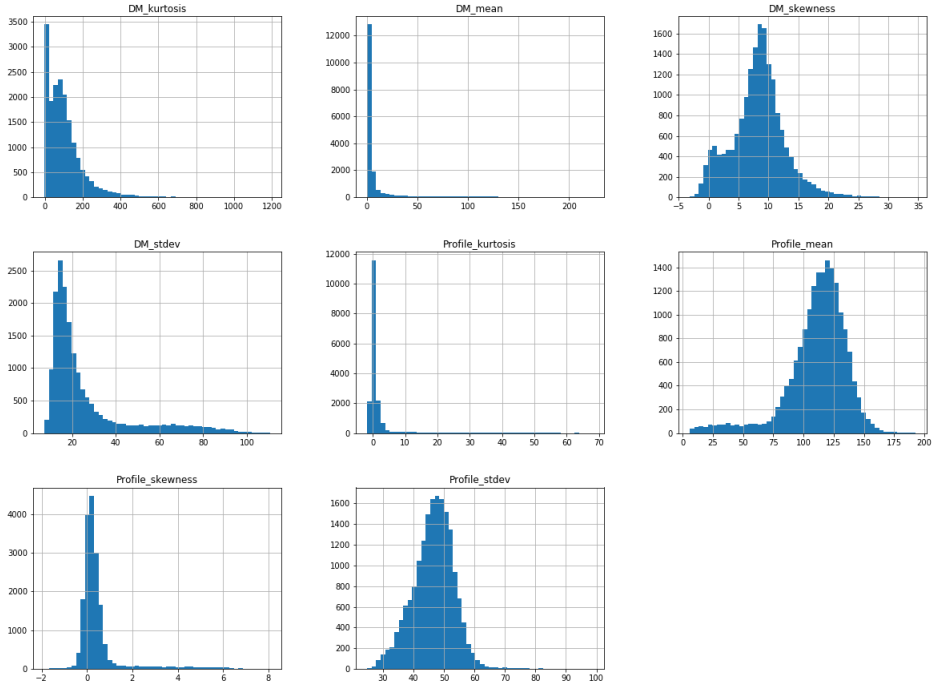

Figura 1: Distribuições relativas às 8 variáveis contínuas. Por ordem, da esquerda para a direita, e de cima para baixo, temos: Curtose da curva *DM-SNR*, Média da curva *DM-SNR*, Assimetria da curva *DM-SNR*, Desvio Padrão da curva *DM-SNR*, Curtose do *folded profile*, Média do *folded profile*, Assimetria do *folded profile* e Desvio padrão do *folded profile*.

digital ou o seu reverso. Por sua vez, a classe fornecida como último parâmetro tem um binário, indicando se o candidato era de facto um pulsar ou não. Em relação a cada um dos grupos são obtidas quatro variáveis contínuas: valor médio, desvio padrão, curtose e assimetria.

A base de dados fornecida contém um total de 17,898 entradas, sendo que destas, 16,259 tratam-se de dados espúrios causados pelo ruído (classe negativa) e as restantes 1,639 tratam-se de dados relativos a sinais de pulsares reais (classe positiva). Para melhor visualização dos dados fornecidos, a Figura 1 apresenta as distribuições das 8 variáveis contínuas de *input*. Essas mesmas distribuições encontram-se pormenorizadas na Figura 2, sob a forma de tabela.

Assim sendo, para esta tarefa de classificação binária os valores das oito variáveis relativas ao sinal são usados na análise de cada candidato, de forma a determinar se se trata ou não de um pulsar (informação fornecida na 9ª coluna do *dataset*).

2.1.2 Pré-processamento dos dados.

Nesta secção, é descrito qual o pré-processamento aplicado ao conjunto de dados, explicando também as vantagens que daí advém.

Relativamente à etapa de pré-processamento dos dados, temos duas considerações importantes: por um lado, todas as colunas de *input* contêm valores de vírgula flutuante, de distribuição contínua; por outro

	Profile_mean	Profile_stdev	Profile_skewness	Profile_kurtosis	DM_mean	DM_stdev	DM_skewness	DM_kurtosis
count	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000
mean	111.079968	46.549532	0.477857	1.770279	12.614400	26.326515	8.303556	104.857709
std	25.652935	6.843189	1.064040	6.167913	29.472897	19.470572	4.506092	106.514540
min	5.812500	24.772042	-1.876011	-1.791886	0.213211	7.370432	-3.139270	-1.976976
25%	100.929688	42.376018	0.027098	-0.188572	1.923077	14.437332	5.781506	34.960504
50%	115.078125	46.947479	0.223240	0.198710	2.801839	18.461316	8.433515	83.064556
75%	127.085938	51.023202	0.473325	0.927783	5.464256	28.428104	10.702959	139.309330
max	192.617188	98.778911	8.069522	68.101622	223.392141	110.642211	34.539844	1191.000837

Figura 2: Estatísticas descritivas sumárias, relativas às 8 variáveis de *input*.

lado, a coluna de *output*, corresponde a uma de duas classes (classe positiva ou negativa, caso seja ou não um pulsar).

Assim, para facilitar a aprendizagem por parte do algoritmo e contribuir para convergir mais rapidamente, fazemos normalização dos dados de input (*feature scaling*), para uma escala no alcance [-1,1]. Por sua vez, relativamente à classe de *output*, extraímos os dados para uma matriz unidimensional em que cada elemento tem um valor binário: 1 caso seja referente a um pulsar, e 0 em caso contrário.

2.1.3 Modelo de aprendizagem a aplicar: redes neuronais.

Neste trabalho iremos usar uma rede neuronal artificial, *Multilayer Perceptron* (MLP), que segue uma técnica de *aprendizagem supervisionada* denominada *backpropagation*. Assim, torna-se importante esclarecer cada um destes conceitos.

Algoritmos que seguem o modelo da aprendizagem supervisionada são algoritmos nos quais os possíveis *outputs* do algoritmo já são conhecidos e nos quais os dados usados para treinar o algoritmo já se encontram mapeados à resposta correta. Pode-se verificar que este modelo é aplicável ao problema dos pulsares na medida em que a totalidade das entradas na base de dados se encontram mapeados a um resultado, como na Secção 2.1.1.

Por sua vez, *backpropagation* é um método usado para o cálculo do gradiente de funções que regem as redes neuronais, sendo frequentemente usado no algoritmo de *gradient descent*. Este é um algoritmo iterativo de otimização de primeira ordem, que permite encontrar o mínimo local de uma função.

Dada uma rede neuronal artificial e uma função de erro (ou função de perda), o algoritmo de *backpropagation*, também conhecido por algoritmo da propagação de erros para trás (*backward propagation of errors*), calcula o gradiente da função de erro considerando sempre o peso das arestas na rede neuronal. Este cálculo é propagado para trás na rede neuronal, com o gradiente da última camada de pesos a ser calculado primeiro e o gradiente da primeira camada de pesos em último lugar (daí o seu nome *backpropagation*). Este fluxo invertido da informação de erro permite uma computação eficiente do gradiente em cada camada, ao invés da abordagem ingénua de calcular o gradiente de cada camada separadamente.

O algoritmo desenvolve-se em duas fases. Numa primeira fase, propaga-se através da rede de forma a chegar aos valores *output*. De seguida, calcula o custo da propagação para cada output, sendo que esta estará intrinsecamente relacionada com a função de erro usada no algoritmo do gradiente descendente. Por fim, propaga-se em sentido contrário de forma a gerar um valor de erro para cada um dos neurónios. Na segunda fase, os pesos das arestas são constantemente atualizados conforme o resultado desempenhado por cada um dos candidatos. Na atualização das arestas é usado uma percentagem previamente definida como fator de atualização dos pesos. Quanto mais elevado for este fator, mais rápido será o treino da rede neural. Por outro lado, quanto mais baixo for este fator, mais preciso será o treino.

2.1.4 Arquitetura das redes neuronais.

As redes neuronais, surgiram com base na perceção biológica das interligações presentes no cérebro humano. Assim, tal como o cérebro humano é constituído por neurónios, os quais recebem impulsos nervosos como entrada e saída de dados, também as redes neuronais são constituídas por camadas de neurónios, ativados com estímulos específicos. Um neurónio tem a capacidade de estimular os seus neurónios vizinhos, quando este é propriamente estimulado (função de ativação). Posto isto, podemos definir matematicamente um neurónio como sendo:

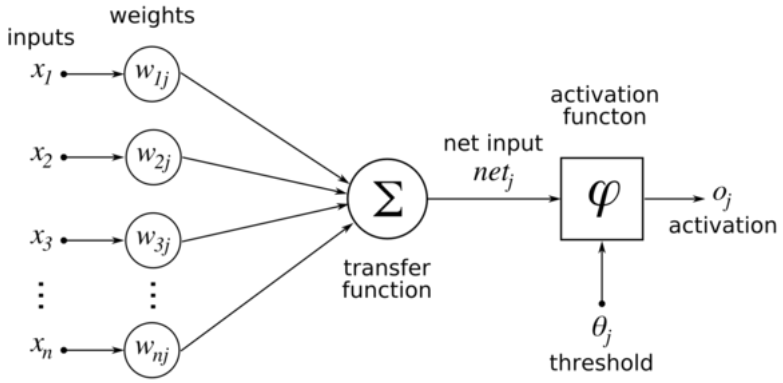

Figura 3: Modelo natemático de um neurónio.

$$y = f_{ativação}(\sum_i w_i x_i - \theta)$$

Figura 4: Representação matemática da saída de um neurónio.

Como o nome indica, é uma rede, por isso a sua consistência detêm vários neurónios conectados na forma de um grafo acíclico. As redes neuronais mais comuns, *feedforward*, são redes em que o *output* de certos neurónios é o *input* de outros. Desta forma é possível separar a rede em diferentes camadas:

1. Uma ***Input Layer***, primeira camada na rede neuronal, responsável por receber os dados, determinar o tamanho do *dataset* e apresentar os padrões de reconhecimento. São apenas constituídas por neurónios de *input*.

2. Uma ou mais ***Hidden Layer***, é nesta(s) camada(s) que é feita maior parte da aprendizagem dos pesos do modelo. Camadas seguidas deverão corresponder a *features* cada vez mais abstratas, permitindo alcançar um modelo generalizável e de bom desempenho.

3. Uma ***Output Layer***, última camada da rede, responsável por apresentar os dados, esta camada é constituída com um número de neurónios geralmente igual ao número de classes de *output* (ou apenas 1 se se tratar de classificação binária). É uma camada que não dispõe de função de ativação, e os cujos são exclusivamente neurónios de *output*.

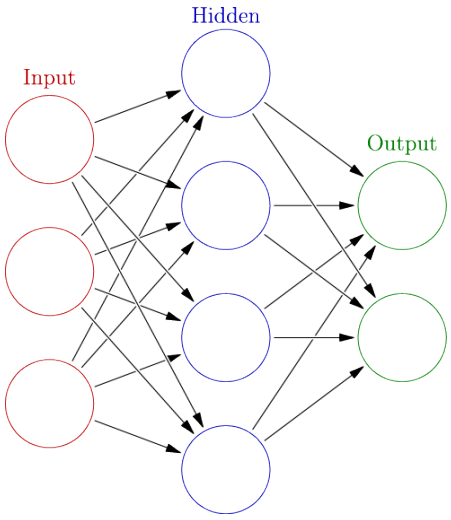

Figura 5: Representação Abstrata da Rede Neural

2.1.5 Configuração da rede neuronal.

Partindo agora para o nosso caso em concreto, é sempre muito complicado prever a configuração/arquitetura da rede ideal para um dado problema, isto é, qual o número ideal de *hidden layers*, qual o número de neurónios das camadas, qual o tipo de funções de ativação usadas, entre outras variações. Por isso é algo que requer alguns testes com diferentes composições para se obter os melhores resultados, coadjuvados por métodos de *Grid Search*, testando um conjunto de combinações promissoras e melhorando iterativamente.

Neste sentido, foram desenvolvidos testes ao número de *hidden layers*, variando entre 1 a 2 o número destas *layers*, e variando também o número de neurónios em cada uma entre 8, 12, 16, ou 32. Como função de ativação, foi usada a função "relu" para as camadas intermédias (*hidden*). A camada de *input* terá obrigatoriamente 8 entradas (uma por variável), e a camada de *output* tem apenas 1 neurónio, ativado pela função "sigmoid", de modo a balisar o *output* entre 0 e 1 (indicativo do grau de certeza de o candidato se tratar ou não de um pulsar). Os resultados destas configurações são apresentados na Secção 2.2.

2.1.6 Forma de Avaliação

Para nos assegurarmos que o modelo cumpre os requisitos, iremos dividir o *dataset* em duas partes, com 80% e 20%, usando a primeira para treino do modelo, e a segunda para avaliação do seu desempenho.

Assim, e tendo em conta a tarefa em questão, será muito útil ter em atenção o *recall* da classe positiva, pois queremos maximizar a identificação de pulsares. Neste sentido, as métricas usadas para avaliação dos modelos serão a precisão global do modelo (*accuracy*), o *recall* da classe positiva, bem como o *F1 score*. O *F1 score* consiste numa média harmónica da precisão e do *recall* para cada uma das classes em causa.

Adicionalmente, todos os testes terão que ter validação cruzada em k partes (*k-fold cross-validation*), de modo a diminuir os efeitos da aleatoriedade no treino dos modelos.

2.2 Trabalho Efetuado

Como referido na secção anterior, foram realizados vários testes, com diferentes configurações da rede neuronal. Os resultados, medidos no *dataset* de teste, estão apresentados na Tabela 1. Como podemos ver, atingimos resultados muito promissores, sendo que o melhor modelo tem duas *hidden layers*, com 32 e 16 neurónios. Todos os testes foram realizados com *3-fold cross-validation*, sendo facilmente reproduzíveis.

First Hidden Layer	Second Hidden Layer	Accuracy
32	32	98.02
32	16	98.07
32	12	98.16
32	8	98.01
16	12	97.93
16	8	97.74
8	8	97.76
32	-	96.41
16	-	96.08

Tabela 1: Resultados obtidos com diferentes configurações da rede neuronal.

Para a realização/codificação dos modelos de redes neuronais, bem como para a sua avaliação, foi usada a linguagem de programação *Python*, bem como as *frameworks* Keras, Tensorflow e Scikit-learn. Para o tratamento e visualização de dados foram usadas as *frameworks NumPy*, *Pandas* e *MatPlotLib*.

2.3 Trabalho Futuro

Apesar de termos atingido resultados muito promissores, falta ainda uma avaliação mais profunda aos modelos gerados, particularmente avaliando o *F1 score* de cada um, como referido na Secção 2.1.6.

Para além disso, achamos possível melhorar estes resultados com algumas variações na configuração da rede, nomeadamente:

- Uso de camadas Convolucionais unidimensionais;
- Uso de regularização L1 e/ou L2;
- Experimentação com diferentes funções de ativação;
- Realização de ***undersampling***, visto os exemplos da classe negativa serem 10 vezes mais comuns que os da classe positiva, retirando alguns dos exemplos negativos do *dataset* de treino.

3 Conclusões

Primeiramente, acreditamos que o nosso conhecimento sobre redes neuronais foi consideravelmente aprofundado, mostrando-se assim o grupo satisfeito com a escolha de tema realizada. De facto, Redes Neuronais artificiais representam um abordagem a problemas bastante atual, contribuindo assim ainda mais para o interesse do grupo no trabalho.

É também evidente que Redes Neuronais artificiais se revelam uma ferramenta extremamente útil, apropriada para resolver problemas semelhantes ao apresentado, resultando numa abordagem ao problema bastante diferente de tudo aquilo com o que nós, como alunos do 3º ano de Informática, tínhamos tido a oportunidade de trabalhar.

Em suma, os resultados obtidos revelaram-se promissores, sendo a precisão do melhor modelo apresentado superior a 98%. Ainda assim, o grupo considera ser possível melhorar o desempenho da rede neuronal artificial implementada, através da experimentação com diferentes camadas e configurações.

4 Recursos

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. "TensorFlow: A System for Large-Scale Machine Learning."In OSDI, vol. 16, pp. 265-283. 2016.

Chollet, François. "Keras."KerasCN Documentation (2017): 55.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn: Machine learning in Python."Journal of machine learning research 12, no. Oct (2011): 2825-2830. Harvard

Lyon, Robert James. "Why are pulsars hard to find?."PhD diss., University of Manchester, 2016.

R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. Knowles, Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach, Monthly Notices of the Royal Astronomical Society 459 (1), 1104-1123, DOI: 10.1093/mnras/stw656

Ruck, Dennis W., Steven K. Rogers, Matthew Kabrisky, Mark E. Oxley, and Bruce W. Suter. "The multilayer perceptron as an approximation to a Bayes optimal discriminant function."IEEE Transactions on Neural Networks 1, no. 4 (1990): 296-298.