Machine Learning: Logistic Regression

Tomás Jerónimo, M9988
Department of Computer Science
University of Beira Interior
Covilhã, Portugal
tomas.jeronimo@ubi.pt

***Resumo*—Neste documento apresenta-se o desenvolvimento do segundo trabalho prático da unidade curricular de Aprendizagem Automática e Mineração de Dados, que se enquadra no segundo ano de mestrado em Engenharia Informática, da Universidade da Beira Interior, no ano 2019/2020.**

I. Introdução

Este trabalho consiste na adaptação de um algoritmo de regressão logística para um conjunto de dados adequado para classificação binária. O resultado do processamento destes dados deve depois ser apresentado devendo-se, para isso apresentar a curva Característica de Operação do Receptor (curva ROC), a *Area Under The Curve* (AUC), e as seguintes medições: *accuracy*, *precision* e *recall*.

II. Conjunto de Dados

A. Descrição

O conjunto de dados [1] utilizado consiste num conjunto de amostras de estrelas de nêutrons candidatas a poderem ser classificadas como estrelas pulsares. Citando Feryal Özel [2] uma estrela de nêutron pode ser classificada como estrela pulsar se tiver a combinação correta de intensidade do campo magnético e frequência de rotação. O estudo destas estrelas é importante para, por exemplo, procurar planetas fora do nosso sistema solar ou medir distâncias cósmicas.

Este conjunto de dados apresenta oito variáveis, que representam diferentes características das amostras recolhidas, e uma última variável que representa a classe de cada amostra (se a amostra é ou não classificada como estrela pulsar).

B. Normalização de dados

Uma vez que o conjunto de dados se encontrava em excelente estado (sem células vazias, nulas ou colunas desnecessárias) procedeu-se imediatamente à normalização dos seus valores no intervalo [0, 1]. Para tal realizou-se a leitura do ficheiro *csv* e para cada coluna, à excepção da última (coluna da classe), achou-se o maior e menor valor.

Terminada a leitura, os dados foram armazenados num novo ficheiro, normalizados, aplicando-se a cada célula a seguinte fórmula:

$$\frac{x - min(x)}{max(x) - min(x)}$$

III. Pré-Processamento de Dados

Antes da implementação de qualquer método realizou-se uma partição do conjunto de dados em dois subconjuntos: subconjunto de treino e subconjunto de testes. A divisão foi feita usando a função *train_test_split()* da biblioteca *sklearl*, e foi feita usando as seguintes proporções: 66% das instâncias para o conjunto de treino e 33% para o conjunto de testes.

Para cada subconjunto utilizaram-se duas listas para armazenar os dados correspondentes: na primeira (x) armazenaram-se as colunas das características das estrelas e na segunda lista (y) armazenou-se a coluna das classes.

IV. Descida do Gradiente

A. Adaptação do algoritmo

Na adaptação do algoritmo de descida do gradiente foram alterados os parâmetros de entrada, substituindo-se pelas listas do subconjunto de treino.

Para o passo de aprendizagem e total de iterações, ao fim de alguns testes, achou-se que os melhores valores para cada uma destas variáveis seriam 0,01 para o passo de aprendizagem e 10.000 para o total de iterações. Na figura 1 apresenta-se a evolução do custo para estes valores, que resultou num valor final de 0,156.

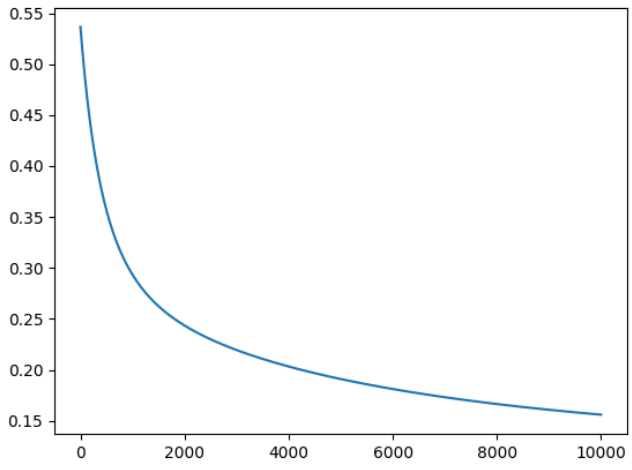

Fig. 1: Evolução do custo ao longo das épocas para o método Descida do Gradiente

Por fim, utilizou-se os *thetas* finais e a lista *x* do subconjunto de testes como parâmetros da função *h()* (função sigmóide) para obter as previsões do modelo para cada instância do subconjunto de testes. O resultado desta função foi devolvido numa lista que se utilizou depois nos cálculos apresentados na próxima subsecção.

B. Curva ROC e restantes métricas

Como referido na subsecção anterior, utilizou-se a lista de previsões, juntamente como a lista *y* do subconjunto de testes para criar uma lista de tuplos com o seguinte formato:

(resultado previsto, resultado real)

Esta lista foi depois ordenada pelo primeiro valor de cada tuplo.

De seguida, iniciou-se o processo de iteração do *threshold*. Para isso utilizou-se uma lista auxiliar para representar os dados à esquerda do *threshold*, mantendo-se os dados à direita na lista de tuplos criada anteriormente. No fim de cada iteração retirou-se o primeiro tuplo da lista da direita e anexou-se à lista da esquerda, repetindo-se o processo até a listas da direita ficar vazia.

Em cada iteração é calculado o *True Positive Rate* (TPR) e o *False Positive Rate* (FPR). Para tal encontra-se o número de *True Positives* (TP), *False Positives* (FP), *True Negatives* (TN) e *False Negatives* (FN) e aplicam-se estes valores nas seguintes fórmulas:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Este par de valores irá ser a coordenada de um dos pontos que irão compor a curva ROC.

Também durante cada uma das iterações é calculada a distância do ponto (FPR,TPR) ao ponto (0,1), que corresponde ao ponto ótimo de qualquer curva ROC. No ponto em que a distância for menor, são guardados os valores de TP, FN, FP, TN.

Terminadas as iterações do *threshold* constrói-se a curva ROC, usando os pontos recolhidos. A curva resultante apresenta-se na figura 2.

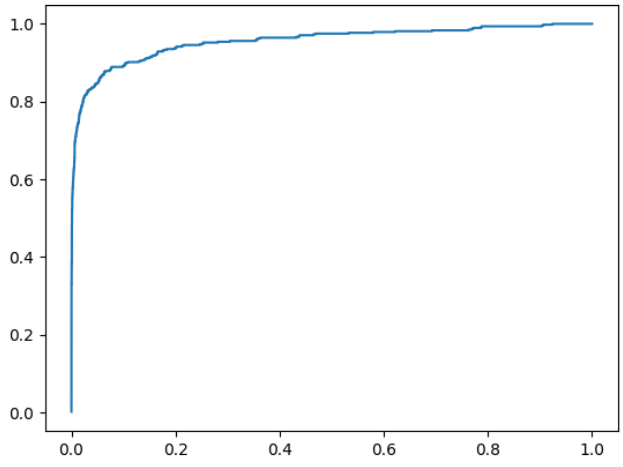

Fig. 2: Curva ROC do método Descida de Gradiente

Por fim, calcula-se a AUC utilizado a função *auc()* da biblioteca *metrics*. Para esta curva o resultado foi 0,95554.

Os resultados da *accuracy*, *precision* e *recall* foram calculados usando os valores guardados do threshold com menor distância ao ponto ótimo e os resultados apresentam-se na seguinte tabela:

Métrica	**Resultado**
Accuracy	0,08092
Precision	0,50535
Recall	0,88912

TABLE I: Resultados da *Accuracy*, *Precision* e *Recall* para o método Descida do Gradiente

V. TensorFlow

A. Adaptação do algoritmo

Tal como no método anterior substituiu-se os parâmetros anteriores pelas listas do subconjunto de treino e utilizou-se os mesmos valores para o passo de aprendizagem e para o número de iterações. A evolução do custo, ao longo das iterações no processo de treino é apresentada na figura 3 e resultou num custo final de 0,131.

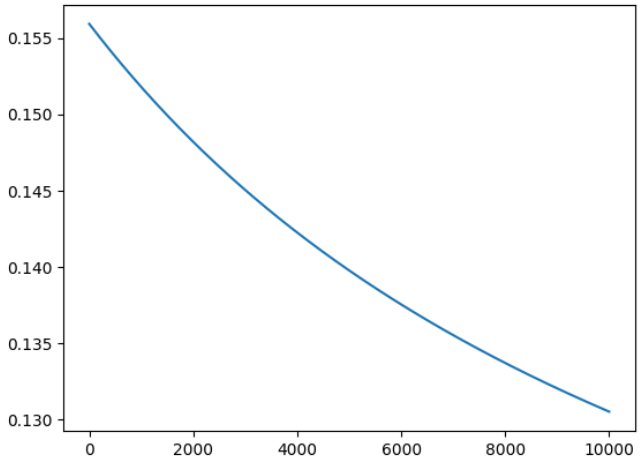

Fig. 3: Evolução do custo ao longo das épocas para o método TensorFlow

Mais uma vez obteve-se uma lista com previsões para cada instância do subconjunto de teste.

B. Curva ROC e restantes métricas

Este processo é exatamente igual a realizado no método da descida do gradiente, utilizando-se, no entanto, as previsões resultantes do método TensorFlow.

A curva ROC resultante apresenta-se na figura 4

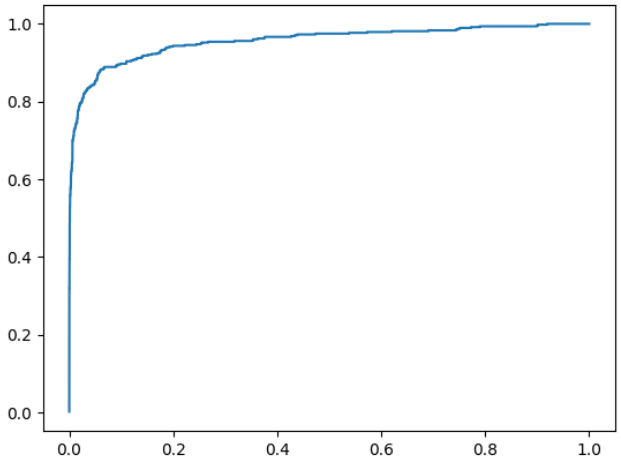

Fig. 4: Curva ROC do método TensorFlow

O valor da AUC resultante desta curva foi de 0,95648 e os resultados das restantes métricas são apresentados na tabela seguinte:

Métrica	**Resultado**
Accuracy	0,08092
Precision	0,53797
Recall	0,88912

TABLE II: Resultados da *Accuracy*, *Precision* e *Recall* para o método TensorFlow

VI. Conclusões

Este projeto tinha como objetivo a adaptação de dois algoritmos de regressão logística para um mesmo conjunto de dados de classificação binária, a implementação de uma curva ROC e o cálculo das métricas AUC, Accuracy, Precision e Recall para cada um destes métodos.

Da procura por um conjunto de dados de classificação binária, resultou a escolha de um *dataset* em muito boas condições onde o pré-processamento dos dados foi mínimo.

Ambos os algoritmos foram adaptados a este conjunto de dados e foi possível calcular todas as métricas necessárias.

Da comparação de resultados entre os dois métodos aplicados, pode-se concluir que os resultados obtidos forma muito semelhantes, tendo o método TensorFlow conseguido obter um custo um pouco menor. Em relação ás restantes métricas os resultados obtidos foram semelhantes.

Referências

[1] Dataset [Online] https://https://www.kaggle.com/pavanraj159/predicting-a-pulsar-star
[2] What are pulsars? [Online] https://www.space.com/32661-pulsars.html