Atividade 1: Previsão do número de compartilhamentos em redes sociais

Jonlenes Castro*
Anderson Rocha†

Abstract

O objetivo deste trabalho é explorar a técnica de regressão linear para fazer a previsão do número de compartilhamentos em redes sociais. A técnica é explorada de diversos aspectos visando obter o menor erro possível. Serão apresentados os preprocessamento, os passos, resultados e conclusões.

1. Introdução

Nas seções seguintes será apresentado o conjunto de dados para essa atividade, e todos os passos realizados para a obtenção do melhor modelo possível com a Regressão Linear (RL).

2. Atividades

Para a exploração da RL foram realizadas as seguintes atividades:

- Preparação e carregamento do *dataset*
- Implementação da RL
- *Feature scaling*
- Remoção de ruídos
- *Feature Selection*
- Alterações no *Target*

3. Análise e carregamento do *dataset*

O *dataset* que será utilizado neste trabalho para a exploração da RL é o *Online News Popularity* que foi coletado e disponibilizado por [1].

*Instituto de Computação, Universidade de Campinas (Unicamp). **Contact**: jonlenes.castro@ic.unicamp.br

†Instituto de Computação, Universidade de Campinas (Unicamp). **Contact**: anderson.rocha@ic.unicamp.br

O mesmo contém 31715 exemplos no arquivo de *train*. Desses exemplos 80% (25372 exemplos) foram utilizados para fazer o treinamento do algoritmo e 20% (6343 exemplos) foram utilizados para validar o treinamento.

No arquivo de *test*, tem-se um total de 7929 exemplos que foi utilizado para fazer os testes após se obter o melhor modelo.

A primeira etapa consiste na análise do *dataset* e dos seus *features*. O *dataset* foi carregado e as duas primeiras colunas foram removidas, pois as mesmas não são preditivas. Também foi separada a ultima coluna do arquivo de *train*, pois ela é o *target* dos exemplos.

Apos tirar essas 3 colunas dos exemplo, restaram 58 colunas que serão utilizadas na RL.

4. Implementação da RL

Finalizado o carregamento do *dataset* e a divisão dos *features*, o próximo passo consistiu em implementar uma classe de LR que computasse a mesma. Essa classe contém os métodos necessários para ser utilizados na RL (como, por exemplo, gradient_descent, normal_equation cost_function, fit, predict, rmse e r2_score).

Com essa classe, foi realizado o primeiro teste sem nenhuma alteração dos dados. A RL não convergiu e função custo aumentou na casa de 10^20 apos cada iteração. O gráfico deste experimento com alguns valores diferentes para alfa pode ser visto na Figura 5. Esse gráfico apresenta a relação entre quantidade de interações e o valor da função de custo.

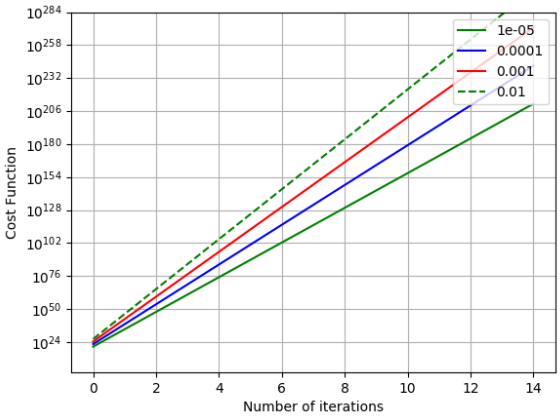

Figura 1. Função de custo no *dataset* original.

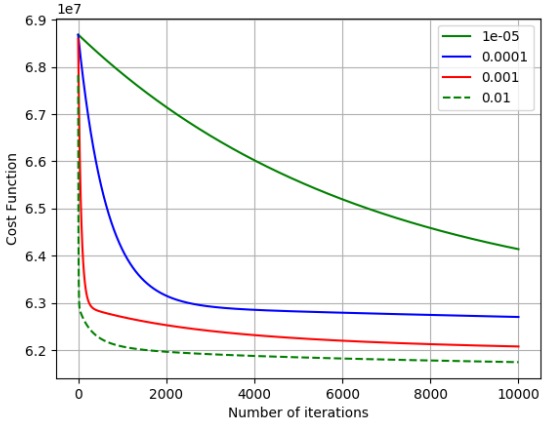

Figura 2. Função de custo após a *rescaling*.

5. *Feature scaling*

Como apresentação na Seção 4, no primeiro teste não houve convergência da RL utilizando o *dataset* sem alteração, então pode se iniciar a alteação do *dataset* fazendo um *feature scaling*.

O *feature scaling* é um método usado para padronizar o intervalo de variáveis independentes ou recursos de dados [2].

Existem diversos métodos para realizar o *feature scaling*, dentre eles, neste trabalho foram explorados o *Rescaling*, *Mean normalisation*, *Standardization* [3].

O *Rescaling* foi aplicado de duas formas diferentes: somente para os *features* que possui o valor mínimo menor do que -1 ou o máximo maior do que 1; e para todos os *features*. A aplicação em todos os *features* obteve melhor resultado.

Aplicando o *Rescaling* em todos os *features*, foi testados vários alfas para encontrar aquele que proporcionasse o menor custo para a RL. O resultado deste experimento pode ser observado na na Figura 2.

Refinando a análise do alfa, tem o melhor resultado da função de custo ($6 * 10^7$), com o valor de alfa igual a 0.25, como pode ser visto na Figura 3. Esse modelo treinado possui *Root Mean Square Error* (RMSE) para o conjunto de dados de validação de aproximadamente $1 * 10^4$.

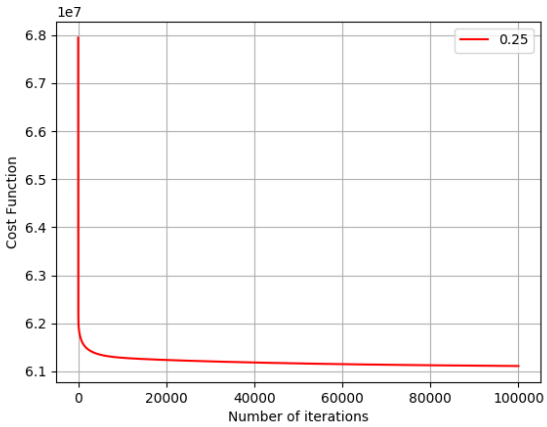

Figura 3. Melhor alfa encontrado para o modelo

O mesmos testes foram realizados para o *Mean normalisation* e *Standardization*. A comparação entres os métodos pode ser visto na Figura 4. Apesar de o *Standardization* ter apresentado uma pequena vantagem em relação aos outros, tanta a função de custo, quanto o RMSE continua na casa de 10^7 e 10^4, respectivamente.

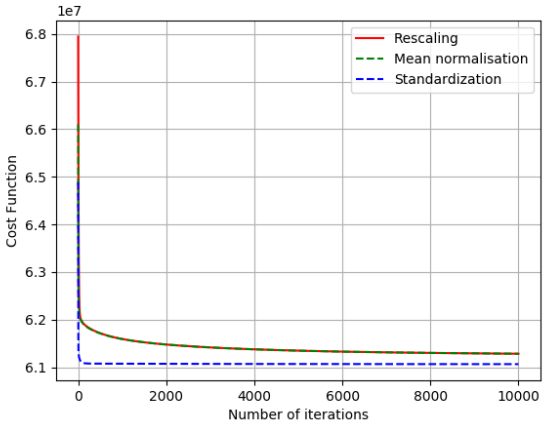

Figura 4. Comparação dos métodos de feature scaling.

6. Remoção de ruídos

Uma etapa importante para alguns algoritmos de *Machine Learning* (ML) é a limpeza dos dados, ou seja, a remoção de exemplos que não corresponde a realidade ou que representam um condição anormal [3]. Nem sempre é possível saber se um dado é ou não de fato um ruido.

Neste experimento, foram considerados possíveis ruídos valores que são extremamente grandes ou pequenos, quando comparado com os demais exemplos para um determinado *feature*.

Esses exemplos possivelmente ruidosos foram encontrados por *feature* e removidos. Após a remoção de cada conjunto de exemplos, foi verificado se o modelo ficou melhor ou pior. Todos aqueles exemplos que estavam atrapalhando o modelo foram removidos do conjunto de treinamento.

Ao fim desse processo, a função de custo caiu para $5 * 10^7$, o RMSE caiu para $8 * 10^3$ e a precisão aumentou para 3.84%. No total foram removidas aproximadamente 1000 exemplos do treinamento.

7. Feature Selection

O próximo passo consiste na exploração de *feature selection*. Este processo seleciona automaticamente os *features* que mais contribuem para a variável de previsão ou saída de interessado [4]. Foram explorados o *Univariate Selection* (UC), *Recursive Feature Elimination* (RFE) e *Principal Component Analysis* (PCA).

A UC pode ser utilizada para selecionar os recursos que têm a relação mais forte com a variável de saída [4]. Para realizar este processo a biblioteca *scikit-learn* [5] fornece a classe *SelectKBest* que pode ser usada com um conjunto de testes estatísticos para selecionar um número específico de recursos. Foi utilizado o teste estatístico qui-quadrado (chi^2) para recursos não negativos. O melhor modelo encontrado com esta técnica coincide com o modelo anterior (mesma quantidade de *features*, mesma precisão e mesmo valor na função de custo).

O RFE funciona removendo recursivamente os atributos e construindo um modelo nos atributos que permanecem. Ele usa a precisão do modelo para identificar quais atributos (e combinação de atributos) contribuem mais para prever o atributo de destino [4]. O melhor modelo encontrado com esse método consiste de um subconjunto com 31 *features*, que após treinamento obteve função de custo com o valor $5.5 * 10^7$, RMSE com valor de $7.8 * 10^3$ e precisão 4.34%.

O PCA usa álgebra linear para transformar o conjunto de dados em uma forma compactada [4]. O melhor modelo encontrado com esse método consiste de um subconjunto com 54 *features*, que após treinamento possui função de custo com o valor $5.5 * 10^7$, RMSE com valor de $7.8 * 10^3$ e precisão 4.24%.

O melhor modelo entre os 3, foi utilizando a eliminação RFE, como pode ser visto na Tabela abaixo.

	UC	RFE	PCA
Num. features	58	31	54
Função de custo	$5 * 10^7$	$5.5 * 10^7$	$5.5 * 10^7$
RMSE	$8 * 10^3$	$7.8 * 10^3$	$7.8 * 10^3$
Precisão	3.84%	4.34%	4.24%

8. Alterações no *Target*

Para tentar melhorar o modelo, o valor de *shares* foi transformado para uma escala logarítmica (log, log2, log10) antes de fazer o treinamento, conforme realizada em [6]. Isso melhorou bastante a função de custo, que ficou compreendida entre 0 e 1. No entanto, a fazer a operação reversa para voltar ao numero de *shares* originais (aplicando a exponenciação, $base^{\ log\_base(found\_shares)}$) o RMSE e a precisão pioraram, então este experimento foi descartado.

Uma outra abordagem utilizada foi a remoção de *outlier*. Este pode ser definido como sendo um valor que apresenta um grande afastamento dos demais da série [7].

Similarmente ao que foi feito com a remoção de ruido dos *features* (Seção 6), aqui foram removidos os *outliers* dos exemplo onde o valor é maior que sua média mais ou menos dois desvios padrão, pois como pode ser visto na Figura 5 muitos poucos exemplos possuem valores acima de 25 mil.

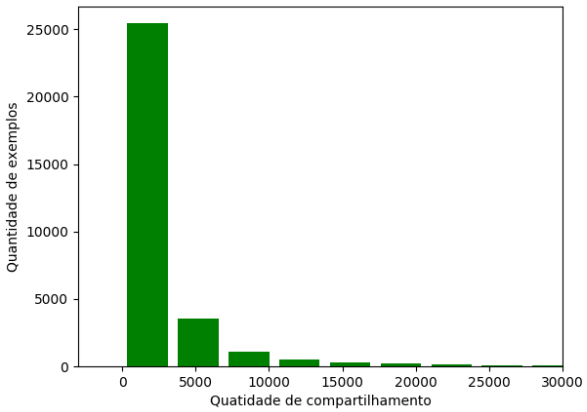

Figura 5. Histograma de 'shares'.

Após essa remoção, tem se um modelo um pouco melhor: função de custo igual a $5*10^6$, RMSE igual a $3.2*10^3$ e precisão igual a 6.46%

9. Normal Equation

Após se obter o melhor modelo possível utilizando o *Gradient Descent* (GD) para calcular os valores dos tetas, também foi utilizado o *Normal Equation* (NE) que calcula os tetas de forma algébrica.

Apesar de o NE não ser recomendado para grande quantidade de exemplos, foram utilizados todos os exemplos do conjunto de treinamento para realizar o calculo. Com os tetas encontrados, foi calculado o valor da função de custo, do RMSE e da precisão, onde foi encontrado aproximadamente o mesmos valores. Para este modelo treinado, o uso do GD conforme mencionado anteriormente é esquivante ao NE.

10. Regularização

A regularização foi implementada tanto no GD quanto no NE, porém não estava sendo utilizada no modelo (lambda igual a zero). Fazendo os testes para diferentes lambadas, o melhor modelo encontrado foi com o lambda igual a zero, ou seja, sem usar a regularização.

11. Conclusão e Trabalhos Futuros

A aplicação da RL para esse *dataset* não obteve um bom modelo, pois ao fazer o teste final (com o arquivo de *test*) a precisão foi de apenas 6%, aproximadamente. O RMSE foi de aproximadamente $3.5 * 10^3$, o que representa o erro grande dos valores preditos em comparação aos *targets*.

Ainda tem muito o que pode ser melhorado nesse modelo para a aplicação da RL, como exemplos de trabalhos futuros, aplicação de técnica de detecção de *outliers*, pois a remoção de ruído realizada na Seção 6 baseados em testes manuais e observações.

Outra possibilidade seria fazer a combinação dos melhores *features* (aqueles encontrados com o REF neste trabalho, por exemplo) e gerar mais *features* a parti de combinação destes melhores, pois, como não foi percebido praticamente nada *overfitting*, pode significar que o modelo não foi ajustado corretamente.

O modelo, apesar de grandes erros, consegui fazer uma predição, mas o resultado pode ser melhorado também utilizando outras tecnicas ou cominações de tecnicas.

Referências

[1] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer, 2015. 1

[2] Sebastian Raschka. About feature scaling and normalization, 2014. Disponível em: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html. 2

[3] J. Gama, K. Faceli, A.C. Lorena, and A.C.P.L.F. De Carvalho. *Inteligência artificial: uma abordagem de aprendizado de máquina*. Grupo Gen - LTC, 2011. 2, 3

[4] Jason Brownlee. Feature selection for machine learning in python, 2016. Disponível em: https://machinelearningmastery.com/feature-selection-machine-learning-python/. 3

[5] Time scikit. Machine learning in python, 2018. Disponível em: http://scikit-learn.org/. 3

[6] Ziyi Liu. Statistical models to predict popularity of news articles on social networks. 2017. 3

[7] Wlademir Ribeiro Prates and Joni Hoppen. O que são outliers e como tratá-los em uma análise de dados?, 2017. Disponível em: https://aquare.la/o-que-sao-outliers-e-como-trata-los-em-uma-analise-de-dados/. 3