O Uso de Diferentes Modelos Preditivos para a Estimação do Resultado de Pedidos de Bolsas de Pesquisa

Matheus Gomes Cordeiro[1] e Abel Pinheiro de Figueiredo[2]

Universidade Federal do Ceará (UFC) - Departamento de Engenharia de Teleinformática (DETI)
60430-160, Fortaleza - Brasil

Abstract. O propósito deste trabalho foi buscar um modelo preditivo para assim entender o fenômeno da aprovação de pesquisas. Nesse sentido, com o poder de simular, aproximadamente, o processo de aceitação e recusa de bolsas de pesquisa é possível então entender quais fatores são mais importantes para a aprovação das mesmas. Logo, é relevante o estudo de qual modelo melhor se adéqua aos dados e de atemão esse modelo foi o não linear.

1 Introdução

Uma pesquisa acadêmica é de suma importância para diversas pessoas e instituições: para o aluno serve como porta de entrada para o cerne acadêmico, provendo-lhe não apenas uma renda monetária e um ganho de conhecimento pelo estudo na área, más também facilita o ingresso na pós-graduação. No caso da instituição, com o sucesso de uma pesquisa acadêmica, a mesma consegue uma maior relevância e uma maior credibilidade na comunidade científica e na sociedade local.

Nesse âmbito, é interessante destacar que o uso inadequado das finanạs destinadas á pesquisa, levando a um alongamento no prazo da dela e , as vezes, a resultados pífios, pode , gradualmente, minar os investimentos dos portadores de rescursos, semelhante ao que ocorreu na Austrália, por volta de 2010, onde a taxa de sucesso de pedidos de bolsa caíram para **20-25%**. Desse modo, seria de grande relevância ter a posse de uma capacidade discriminatória eficiente para que se possa analisar as características que propiciam ou contribuem para a aprovação de uma pesquisa científica, resultando em uma melhor gestão destes recursos e assim possibilitando um aumento no sucesso da aprovação de pesquisa.

Portanto, este trabalho se dispôs a realizar uma análise do dataset cedido pela Universidade de Melbourne [1] e poder assim produzir diferentes modelos que permitam predizer de forma eficiente o resultado de um pedido de financiamento pesquisa.

2 Metodologia

O dataset utilizado para as predições possui **8708** amostras com **1882** preditores, dos quais **1630** foram descartados por não terem muita relevância para os modelos, sobrando apenas **242** preditores, além disso, pela análise dos dados, as correlações se mostram baixas e a diferença entre as magnitudes de uma amostra para a outra é razoável. Todas as amostras são referentes pedidos de bolsa de pesquisa entre os anos de **2005** a **2008**, e dentro desse dataset existe uma coluna que mostra se o pedido foi bem sucedido ou não, com os valores **0** e **1**, essa coluna deve ser utilizada como referência no treinamento do modelo.

Foram testados dois métodos de classificação, dos quais um é linear (Regressão Logística) e o outro não linear (Rede Neural) e após os testes os dois foram comparados através da Matriz de Confusão e da acurácia do desempenho. Cada método é detalhado a seguir:

2.1 Classificação Logística

$$f(X) = X^T\omega + c \tag{1}$$

$$Probabilidade = \frac{1}{1 + e^{-f(X)}} \tag{2}$$

$$Custo = \frac{1}{2}\omega^T\omega + \sum_{i=1}^{n} log(e^{-y_i f(X_i)} + 1) \tag{3}$$

- O **f(X) (1)** representa uma regressão linear, onde **X** é a matriz de preditores (**m x n**:

m = Número de Amostras; n = Número de Preditores), ω um vetor de pesos β_i (ω = [β_1,β_2,...,β_n]) e **'c'** o intercept (β_0).

- A Probabilidade **(2)** de determinada amostra ser considerada um sucesso (bolsa aprovada) é determinada por uma função logística, que tendo como o expoente do número de *euler* o **f(X)**, acaba virando uma Regressão Logística, ou melhor um Perceptron.

- A função de Custo **(3)** é a *Logistic Loss* com uma Regularização do tipo L_2 - simbolizada por *(1/2)* $\omega^T\omega$. Nesse caso, a Regularização funciona assegurando que o modelo não irar se tornar muito específico e gerar um *overfitting*, pois ela limita o poder da variância durante o aprendizado. Na função de Custo **(3)** abordada, diferente da Entropia Cruzada onde se faz necessário a maximizaçã da mesma, deseja-se uma minimização, nesse sentido, é notável que a medida que $y_i f(X_i)$ aumenta a exponencial fica cada vez mais próxima de zero, levando o logarítimo a ficar mais perto de **log(1)** que é zero, porém aumentar $y_i f(X_i)$ também significa aproximar a Probabilidade **(2)** de 1. Tudo isso só vale para $y_i = 1$, que é quando a amostra é classificada como *sucesso*, entretanto ao otimizar o reconhecimento de uma amostra do tipo 1, é ensinado ao modelo a identificar amostras do tipo 0, já que: $Prob \rightarrow 1 \Rightarrow$ (1 - $Prob$) $\rightarrow 0$, como (1 - $Prob$) = $(1+Exp(f(X))^{-1}$ é bem consistente que quando $y_i f(X_i) \rightarrow \infty$, (1 - $Prob$) $\rightarrow 0$.

- Foi realizada uma Regressão Logística no contexto das características anteriores e se obteve um modelo com uma acurácia de **84%** (foi feito o uso de um dataset de teste). A Matriz de Confusão está na **Tabela 1**.

2.2 Rede Neural

$$\sigma(X) = \frac{1}{1 + e^{-(X^T\omega + c)}} \tag{4}$$

$$Custo = \sum_{i=1}^{n} [y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)] \tag{5}$$

$$\omega(t+1) \leftarrow \omega(t) - \eta \frac{\partial Custo}{\partial \omega_i} \tag{6}$$

- A função **(4)** é conhecida pelo nome de função de trãnsferência, e está acoplada aos neurônios da Rede Neural, possuindo a excepiconal tarefa de filtrar os sinais que chegam na entrada do neurônio, no qual esse sinal é representado pela soma ponderada dos valores presentes no vetor de entrada ($X^T\omega$) mais um valor de bias **"c"**. Esse vetor de entrada pode vir tanto de uma camada oculta quanto da camada de entrada, dependendo, somente, da camada onde o neurônio em questão está. O fato de $\sigma(X)$ ser uma sigmoide ajuda no processo de classificação, já que a mesma retorna uma probabilidade, como visto no tópico passado.

- A função de custo **(5)** é chamada de Entropia Cruzada, e ela possui uma performance bastante consistente no âmbito da classifição. A mesma funciona da seguinte forma: quando uma amostra é da classe **1** (sucesso), por exemplo, implica que $y_i = 1 \Rightarrow$ (1-y_i) = 0, nesse contexto, o custo vai ser dado por uma soma dos logarítimos das prdições de todas as amostras: $log(\hat{y}_1)+log(\hat{y}_2)+...+log(\hat{y}_i) = log(\prod_{i=1}^{n} \hat{y}_i) = log[L(\omega; X)]$, onde $L(\omega; X)$ é a verossimilhança de uma variável aleatória **X** em "**i**" observações, fazendo o uso de um conjunto de parâmetros $\omega = [\omega_1, \omega_2, ..., \omega_i]$ (pesos da rede), tal que $\hat{y}$ é a sua função de probabilidade. Nesse contexto, para otimizar o modelo, busca-se maximizar a função de custo pois $max[Custo] \Rightarrow max[log(L(\omega; X))] \Rightarrow max[L(\omega; X)]$, e isso significa maximizar o produto da probabilidade de ser **1** de todas as amostras que são, realmente, **1**. Já quando uma amostra é de classe 0 (sem sucesso) implica que $y_i = 0 \Rightarrow$ (1-y_i) = 1: o processo de otimização é semelhante a situação anterior e a única diferença é que nesse caso é maximizado o produto da probabilidade de ser **0** de todas as amostras que são, realmente, **0**. Desse modo, quando se maximiza a Entropia Cruzada (função de custo), são encontrados parâmetros que tanto retornam probabilidades próximas a 1 das amostras que são 1 quanto ratornam probabilidades próximas a 0 das amosras que são 0, reproduzindo assim o fenômeno que se deseja modelar , o qual é uma aproximação de um *Experimento de Bernoulli.*

- Os pesos da rede neural são atualizados de acordo com **(6)**, o qual mostra uma atribuição realizada dentro de um algoritmo, nela se vê o novo peso como sendo o antigo menos o produto da taxa de aprendizado "η" com a derivada da função de custo em relação ao peso. Os ajustes nos pesos se resumem a gradientes locais da função de custo, isso na camada de saída, depois são gerados erros locais que sáo repassados via *backpropagation* para o resto da rede. Entretanto, o termo associado com a taxa de aprendizado é semelhante a um vetor que direciona a busca no espaço de pesos, no qual se pretende encontrar uma configuração de pesos para rede que é relacionada ao menor erro. Nesse sentido, é executado o cálculo da derivada da Entropia Cruzada e um sinal negativo é gerado, isso implica que o peso tendera a aumentar no decorrer das iterações, porém, quando a busca está chegando próximo ao custo mínimo, aquela derivada vai se aproximando de 0 e portanto o ajuste nos pesos vai ficando muito pequeno, o que é bem consistente, já que quanto mais perto do alvo a busca chega menos necessário é o ajuste dos pesos.

- Foi construída uma Rede Neural segundo os critérios apresentados anteriormente. Essa Rede Neural possui uma taxa de aprendizado igual a **0.009**, além disso possui uma camada oculta com **126** neurônios, baseado na regra empírica da camada oculta possuir número de neurônios igual a $\lfloor \frac{N_{in}+N_{out}}{2} \rfloor$, onde N_{in} é o número de neurônios de entrada que é **252** e N_{out} é o número de neuroônios de saída que é **1**. A acurácia da rede foi de **86%** e a Matriz de Confusão está na **Tabela 2**.

	Sucesso	Sem Sucesso
Sucesso	283	46
Sem Sucesso	39	150

Table 1: Regressção Logística

3 Resultados

Nesse cenário, é notável que a Rede Neural obteve resultados um pouco melhores que a Regressão Logística, isso pode ser justificado pelo fato de que modelos não lineares conseguem traduzir com mais facilidade relações mais complexas entre os preditores. Outro ponto relevante a se destacar é que a foi utilizado o Gradiente de Descida Estocástica com *batch* igual a **50** e isso pode ter influenciado na acurácia.

Esse trabalho foi todo feito com a linguagem de programação **Python** [**2**] dentro da aplicação web **Jupyter Notebook** [**3**], com o uso da biblioteca de Aprendizado de Máquina **skit-learn** [**4**] e da biblioteca de Análise de Dados **pandas** [**5**], além disso todos os códigos se encontram em [**6**].

References

[1] **Predict Grant Applications, (2010), https://www.kaggle.com/c/unimelb**

[2] **Python Software Foundation. Python Language Reference, version 3.6. http://www.python.org.**

[3] **Kluyver, Thomas Ragan-Kelley, Benjamin Perez, Fernando Granger, Brian Bussonnier, Matthias Frederic, Jonathan Kelley, Kyle Hamrick, Jessica Grout, Jason Corlay, Sylvain Ivanov, Paul Avila, DamiÃ¡n Abdalla, Safia Willing, Carol development team [Unknown, Jupyter. (2016). Jupyter Notebooks â a publishing format for reproducible computational workflows. http://jupyter.org/index.html**

[4] **Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.https://scikit-learn.org/stable/index.html**

[5] **Mckinney, Wes. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics. Python High Performance Science Computer. https://pandas.pydata.org/**

[6] **Cordeiro M.G. Machine-Learning. Repositório GitHub: https://github.com/matheus123deimos/Machine-Learning.**

	Sucesso	Sem Sucesso
Sucesso	282	47
Sem Sucesso	30	159

Table 2: Rede Neural