Classificação de Remuneração com Base em Dados Sociais

Gabriel Saraiva Espeschit
Escola de Engenharia da UFMG
Belo Horizonte, Brasil
gabrielespeschit@ufmg.br

Lucas Guimarães Hofner
Escola de Engenharia da UFMG
Belo Horizonte, Brasil
lgh@ufmg.br

Victor Paulinelli Miranda
Escola de Engenharia da UFMG
Belo Horizonte, Brasil
victor-paulinelli@ufmg.br

***Resumo*—Nesse artigo investiga-se o desempenho de dois métodos de classificação do grupo de remuneração de uma pessoa com base nos dados sociais. Foram utilizadas as Máquinas de Aprendizado Extremo e o Classificador Bayesiano para fazer a classificação.**

***Palavras Chave*—redes neurais artificiais, reconhecimento de padrões, máquinas de aprendizado extremo, classificador bayesiano**

I. Introdução

Problemas de reconhecimento de padrões são problemas em que, para uma série de dados de entrada, deve-se associar esses dados a uma classe conhecida. Esse tipo de problemas também são conhecidos como problemas de classificação.

Nesse estudo, pretende-se investigar o desempenho de dois métodos capazes de resolver o problema de classificação. Será utilizado um banco de dados que contem os dados sociais de um indivíduo para relacionar com a faixa salarial do mesmo. Os métodos escolhidos foram: Classificador Baeysiano e Máquinas de Aprendizado Extremo (ELM). A bases de dados foi retirada do site da universidade da Califórnia.

A seguir, será feito uma breve revisão a respeito dos métodos implementados e das base de dados utilizadas.

II. Revisão da Literatura

Nessa seção iremos fazer uma breve explicação a respeito do funcionamento de cada metodologia adotada nesse estudo.

A. Classificador Bayesiano

O Classificador Bayesiano é baseado no Teorema de Bayes para classificador os dados. Esse teorema é representando pela equação abaixo:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{1}$$

Dessa forma, é obtida a probabilidade de um determinado dado de entrada ser de uma classe (chamada de probabilidade *a posteriori*) se calculando a probabilidade da classe em questão conter aquele dado (chamado de verossimilhança), também levando em conta a probabilidade daquela classe aparecer no conjunto de dados de entrada (probabilidade *a priori*).

O cálculo das probabilidades *a priori* é realizada simplesmente encontrando a frequência que uma classe aparece dentre todos os dados de entrada. Sendo assim, a probabilidade *a priori* é ser dada por

$$P(C_k) = \frac{n_k}{\sum_{i=1}^{N} n_i} \tag{2}$$

em que $P(C_k)$ é a probabilidade *a priori* da classe *k*, N é a quantidade total de dados e *n* é a quantidade de dados de uma classe.

A verossimilhança é calculada por meio do cálculo da função de distribuição de probabilidade (PDF) da classe em questão. Normalmente, é utilizado a distribuição normal para modelar a PDF para cada classe do problema. Uma função de probabilidade normal com n atributos, com correlação diferente de zero, é determinada por

$$P(x|C_i) = \frac{1}{(2\pi)^n |COV|} exp(-\frac{1}{2}(x-\mu)^T COV^{-1}(x-\mu)) \tag{3}$$

em que

$$COV = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{21}\sigma_2\sigma_1 & \sigma_2^2 & \cdots & \rho_{2n}\sigma_2\sigma_n \\ \vdots & \vdots & \cdots & \vdots \\ \rho_{n1}\sigma_n\sigma_1 & \rho_{n2}\sigma_n\sigma_2 & \cdots & \sigma_n^2 \end{bmatrix} \tag{4}$$

e μ é o vetor de médias de cada atributo, σ_n é o desvio padrão do n-ésimo atributo e ρ_{kn} é a correlação entre o k-ésimo atributo e o n-ésimo atributo. Assim, uma vez que foi modelado a PDF da função normal para cada classe (calculando as médias, desvios padrão e correlação dos atributos daquela classe), pode-se calcular a verossimilhança de um dado apenas avaliando a PDF naquele ponto.

Por fim, seria necessário calcular P(B) da equação (1). Porém essa probabilidade não pode ser calculada. Dessa forma, a solução normalmente utilizada é dividir o numerador dessa equação para uma classe I pelo numerador dessa equação para outra classe J.

$$K = \frac{P(x|C_I)P(C_I)}{P(x|C_J)P(C_J)} \tag{5}$$

Isso da um fator, K, que se maior que 1, indica que a probabilidade daquele dado ser da classe I é maior que da classe J. Caso seja menor, a probabilidade é maior que aquele dado seja da classe J. Dessa forma é feita a classificação.

B. Máquinas de Aprendizado Extremo

ELMs (*Extreme Learning Machines*) são redes neurais artificiais que fazem o mapeamento da variável de entrada por meio de uma função de mapeamento $\phi_h(\mathbf{x_i}, \mathbf{Z})$, tal que não haja nenhum tipo de restrição sob esse mapeamento e a matriz $\mathbf{Z}$ seja selecionada de forma aleatória. O número de funções (neurônios) $h(\mathbf{x}, \mathbf{z_i})$ deverá ser suficiente para garantir a separabilidade no espaço da camada intermediaria.

Para treinar uma ELM, o primeiro passo é criar uma matriz aleatória de pesos $\mathbf{Z}$. Em seguida, deve-se encontrar a matriz de mapeamento $\mathbf{H}$ por meio da operação $\mathbf{H} = \phi_h(\mathbf{xZ})$, em que ϕ_h é usualmente uma função sigmoidal como a tangente hiperbólica. Sendo assim, o problema passa a ser encontrar uma matriz de pesos $\mathbf{W}$, de dimensões pxm, que satisfaça (6).

$$\phi_o(\mathbf{HW}) = \mathbf{Y} \tag{6}$$

Para problemas de classificação, podemos fazer uma linearização de (6) e reduzir a equação à $\mathbf{HW} = \mathbf{Y}$, de tal modo que podemos obter a solução por meio da pseudo-inversa de $\mathbf{H}$, $\mathbf{H}^+$, como pode ser visto em (7).

$$\mathbf{W} = \mathbf{H}^+\mathbf{Y} \tag{7}$$

Desse modo, obtemos os pesos $\mathbf{W}$ que satisfaçam a equação (7) e criamos nosso modelo ELM.

III. Metodologia

A. Classificador Bayesiano

O classificador de Bayes foi construído utilizando a linguagem R. Inicialmente, importou-se o conjunto de dados de teste e de treino. Para cada conjunto, foram extraídos os dados dos adultos que ganham menos que 50 mil dólares anuais em uma matriz e os que ganham mais, em outra. Essas etapas estão explicitadas na figura 1. Em seguida, foram calculados as probabilidades *a priori* e os parâmetros para a construção da PDF de cada classe.

```
5
6  x <- read.csv('D://                    //adult_fac.data',sep=",")
7  x_test <- read.csv('D:                   '/adult_fac.test',sep=",")
8
9
10 ganhamMaisTreino <- x[x[,16]==1,c(2:15)]
11 ganhamMaisTeste <- x_test[x_test[,16]==1,c(2:15)]
12
13
14 ganhamMenosTreino <- x[x[,16]==0,c(2:15)]
15 ganhamMenosTeste <- x_test[x_test[,16]==0,c(2:15)]
16
17
```

Fig. 1. Código do carregamento e separação dos dados

Para o cálculo das probabilidades *a priori*, foi utilizada a equação (2). Assim, calculou-se a quantidades de linhas (amostras) do conjunto de treino da matriz de pessoas que ganham mais que 50k anuais e dividiu-se pela quantidade total de linhas do conjunto de treino. Essa é a probabilidade *a priori* de ser uma pessoa que ganhe mais que 50k anual. Da mesma forma, foi feito o cálculo para a probabilidade *a priori* de pessoas que ganham menos que 50k anuais. O código dessa implementação em R se encontra na figura 2

```
19 #Calcular as evidências
20 probganhamMais <- nrow(ganhamMaisTreino) / ( nrow(ganhamMaisTreino) + nrow(ganhamMenosTreino))
21 probganhamMenos <- nrow(ganhamMenosTreino) / ( nrow(ganhamMaisTreino) + nrow(ganhamMenosTreino))
22
```

Fig. 2. Código do cálculo das probabilidades a priori

Em seguida, para cada classe (pessoas que ganham menos e mais que 50k anuais), foram calculadas as médias de cada um dos 14 atributos e a matriz de covariância desses atributos (Figura 3). Para as médias, foi iterado em cada coluna e calculado o valor da média, construindo um vetor de médias. Para a matriz de covariância, foi utilizada a função *cov()*, que retorna justamente a matriz de covariância da matriz de entrada. Esses dois parâmetros serão utilizados a seguir para o cálculo da PDF de n variáveis, conforme a equação (3).

```
25 uganhamMenos <- vector()
26 uganhamMais <- vector()
27 #Calcular parâmetros para verossimilhança
28 for(i in 1:ncol(ganhamMenosTreino)) {
29   uganhamMenos <- append(uganhamMenos,mean(ganhamMenosTreino[,i]),)
30   uganhamMais <- append(uganhamMais,mean(ganhamMaisTreino[,i]),)
31 }
32 KganhamMais <- cov(ganhamMaisTreino)
33 KganhamMenos <- cov(ganhamMenosTreino)
34
```

Fig. 3. Código do cálculo do vetor de média e de covariância

Com isso, já temos o classificador treinado, pronto para testes.

B. Máquinas de Aprendizado Extremo

A criação da ELM utilizada nesse projeto foi feita seguindo as equações vistas em II-B. Sendo assim, foi criada duas funções em na linguagem de programação *Python* para fazer o treinamento da ELM e a validação dos resultados. A função de treinamento pode ser vista na figura 4. A de validação está na figura 5.

```
def ELM_train(X_data, Y_data, num_neuronios):
    p = num_neuronios
    X = X_data
    Y = Y_data
    n = X.shape[1]
    Z = np.random.uniform(low = -0.5, high = 0.5, size = (n+1, p))
    Xaug = np.append(X, np.ones((X.shape[0], 1)), 1)
    H = np.tanh(np.matmul(Xaug, Z))
    W = np.matmul(np.linalg.pinv(H), Y)
    return W, H, Z
```

Fig. 4. Função de treinamento da ELM

```
def ELM_y(X_data, W, Z):
    X = X_data
    Xaug_t = np.append(X, np.ones((X.shape[0], 1)), 1)
    H_t = np.tanh(np.matmul(Xaug_t, Z))
    Y_hat = np.matmul(H_t, W)
    return np.where(Y_hat > 0, 1, -1)
```

Fig. 5. Função de validação da ELM

Na função de treinamento se passa os dados de treino, os valores $\mathbf{Y}$ correspondentes e o número de neurônios que se

deseja ter no modelo. A função de treinamento nos retorna os parâmetros **W, H** e **Z** encontrados para aquela quantidade de neurônios. Os parâmetros **W** e **Z** são passados para função *ELM_y* com os dados de de treino e retorna o **Y** previsto para cada amostra.

IV. Descrição dos dados

Para este trabalho foi utilizado o banco de dados *Adults* disponível para acesso [2]. As amostras desse banco de dados representam uma pessoa cuja renda anual possa ser menor ou igual a 50 mil dólares ou maior do que este valor, e dessa forma configura um problema de classificação binário. A classificação deve ser realizada a partir de 14 atributos demográficos reais, extraídos por por Barry Becker do censo americano de 1994 e ilustrados na Tabela I.

Parâmetro	Num. Elementos	Descrição
age	Var. Contínua	Idade
workclass	8	Classe de Trabalho
fnlwgt	Var. Contínua	Representatividade estatísica da amostra
education	16	Escolaridade
education-num	Var. Contínua	Anos de escolaridade
marital-status	7	Estado Civil
ocupation	15	Profissão
relationship	6	Relacionamento
race	5	Etnia
sex	2	Gênero
capital-gain	Var. Contínua	Ganho de capital
capital-loss	Var. Contínua	Perda de capital
hours-per-week	Var. Contínua	Jornada de trabalho semanal
native-country	41	País de origem

TABELA I
DESCRIÇÃO DOS ATRIBUTOS DO DATASET "ADULTS"

O banco de dados, entretanto, não é inteiramente completo para todas as suas 48842 instâncias, o que pode causar erros ou resultados grosseiramente inadequados no classificador. Para isso, o próprio fornecedor do banco de dados sugere uma pré-filtragem que exclua menores de 16 anos, exija que a pessoa trabalhe ao menos 1 hora por semana, e garanta que o atributo *fnlwgt* seja maior que 1. Em todas as simulações apresentadas neste trabalho os dados foram filtrados como explicitado.

V. Experimentos

A. Classificador Bayesiano

O conjunto de dados de teste já estava previamente separado do conjunto de dados de treino. Dessa forma, não foi realizada a validação cruzada com N iterações. Além disso, para um dado conjunto de dados de treino e teste, o classificador sempre irá classificar da mesma forma, já que o cálculo dos parâmetros e das probabilidades seriam determinísticos nesse caso. Assim, o experimento realizado com o classificador bayesiano se deu simplesmente fazendo uma execução sobre o conjunto de dados de teste fornecido. Dessa forma, com o classificador já 'treinado', iterou-se sobre o conjunto de teste de cada classe. Para cada dado, foi calculado a verossimilhança, pela equação (3) para as duas classes, utilizando a função *dmvnorm()*. Utilizando esse valor e as probabilidades *a priori*, foi calculado o fator K para cada classe, conforme a equação (5). Caso seja menor que um, foi classificado como uma pessoa que ganha menos que 50k, do contrário, foi classificado como uma pessoa que ganha mais. Dessa forma, foi comparado com o valor esperado, de acordo sobre qual conjunto de teste de qual classe se estava iterando. Se era a classe de teste das pessoas que ganham mais que 50k e foi classificado dessa forma, é um acerto. Se não, se for classificado como uma pessoa que ganha menos, é um erro. Cada acerto foi computado em uma variável, que no final, dividido pela quantidade de amostras do conjunto de teste, foi calculada a acurácia do método. O teste dos dados para avaliação da acurácia do classificador se encontra na Figura 6.

```
acerto <- 0
for(i in 1:nrow(ganhamMaisTeste)){
  probXganhamMais <- dmvnorm(ganhamMaisTeste[i,],uganhamMais,KganhamMais)
  probXganhamMenos <- dmvnorm(ganhamMaisTeste[i,],uganhamMenos,KganhamMenos)
  K <- (probXganhamMais * probganhamMais) / (probXganhamMenos * probganhamMenos)
  if(K>=1) {
    acerto <- acerto + 1
  }
}

for(i in 1:nrow(ganhamMenosTeste)){
  probXganhamMais <- dmvnorm(ganhamMenosTeste[i,],uganhamMais,KganhamMais)
  probXganhamMenos <- dmvnorm(ganhamMenosTeste[i,],uganhamMenos,KganhamMenos)
  K <- (probXganhamMais * probganhamMais) / (probXganhamMenos * probganhamMenos)
  if(K<1) {
    acerto <- acerto + 1
  }
}

acuracia <- acerto * 100 / (nrow(ganhamMenosTeste) + nrow(ganhamMaisTeste))
```

Fig. 6. Código da avaliação do classificador

B. Máquina de aprendizado Extremo

Para a ELM os experimentos conduzidos consistiu em variar o número de neurônios da camada intermediária da rede. Sendo assim foram conduzidos experimentos em que foram criadas ELMs com 100, 200, 300, 500, 1000, 1500 e 2000 neurônios para avaliar qual seria o impacto disso na acurácia obtida. Para cada número de neurônios, foram criados 10 modelos cada um sendo validado conta o conjunto de teste. A acurácia média para cada número de neurônio foi salva. A figura 7 mostra o código responsável por fazer esses testes.

```
num_neuronios = [100, 200, 300, 500, 1000, 1500, 2000]
acc = []
for num in num_neuronios:
    acc_aux = []
    for i in range (10):
        W, H, Z = ELM_train(X_data_train, Y_data_train, num)
        Y_data_val = ELM_y(X_data_test, W, Z)
        acc_aux.append((Y_data_val == Y_data_test).mean())
    acc.append(np.mean(acc_aux))
```

Fig. 7. Experimentos conduzidos com ELM

VI. Resultados

A. Classificador Bayesiano

Para o classificador bayesiano, com uma única iteração, o resultado da acurácia obtida para o classificador se encontra na Figura 8.

acuracia	82.2615318469382

Fig. 8. Acurácia obtida com o classificador de Bayes

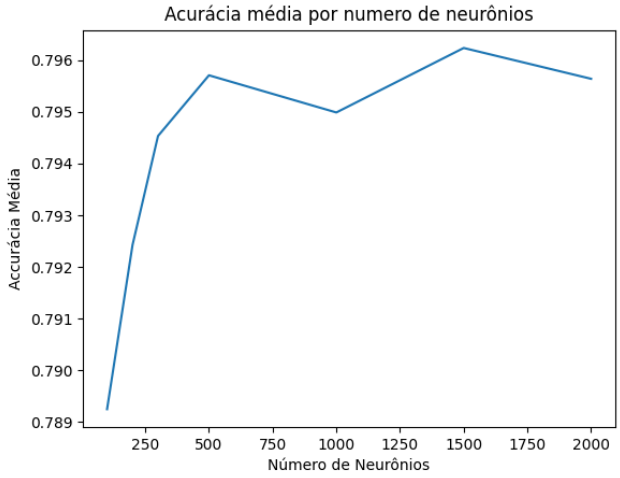

Fig. 9. Acurácia média obtida com as ELMs

B. Máquina de aprendizado Extremo

Para ELM as acurácias média para cada número de neurônios utilizados estão dispostas na figura 9.

Podemos verificar que a melhor acurácia foi obtida utilizando 1500 neurônios,em que uma média de 79,8% foi atingida.

VII. Conclusão

Podemo concluir que, as ELMs não foram capazes de superar os Classificadores Baeysianos para esse problema de classificação. Além de se tratar de um solução computacionalmente mais cara (no caso de 1500 neurônios), a acurácia média encontrada para as ELMs não superou em nenhum caso a acurácia encontrada pelo Classificador Bayesiano.

Isso é um atestado ao bom condicionamento do Naïve Bayes como um classificador, especialmente para problemas binários. ELM's, apesar de terem uma performance relativamente boa para classificação, são automaticamente tunadas com os dados de treino facilitando a ocorrência de *overfitting*.

Em estudos futuro, será interessante explorar como que esses dois métodos conseguem conversar melhor e criar um método que seja uma junção de ambos.

Referencias

[1] A. P. Braga, "Introdução à Engenharia de Dados Uma Perspectiva de Redes Neurais Artificiais e Reconhecimento de Padrões," Escola de Engenharia da UFMG, 2020.

[2] Dua, D. and Graff, C. "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml, Irvine, CA: University of California, School of Information and Computer Science, 2019