Regressão Linear Com Multiplas Variaveis

Beariz Camargo Câmara[1]

[1]Inteligência Artificial - Universidade Federal de Mato Grosso do Sul (UFMS-CPPP)

1. Regressão Linear com Multiplas Variaveis

A análise de regressão logística é uma técnica analoga a regressão linear, ela é utilizada quando é necessário categorizar alguma variavel por classes, ou sej, quando precisarmos classificar algo. Essa técnica proporciona uma previsão que esta entre 0 e 1, o que nos proporciona analisar qual a probabilidade de algo pertencer a determinada classe.

2. Descrição do Trabalho

A tarefa consistiu na implementação da regressão logística para classificar em um conjunto de fotos quais possuiam um gato e o quais não possuiam.

3. Código

```
import numpy as np
import matplotlib.pyplot as plt
from reportlab.pdfgen import canvas
import pandas as pd
import cv2
import glob
from utils import salva_imagem_com_predicao

#arquivos_de_gatos = "./data/train/cat/*.png"
#arquivos_nao_gatos = "./data/train/noncat/*.png"

arquivos_de_gatos = "./data/train/testando/cat1/*.png"
arquivos_nao_gatos = "./data/train/testando/noncat1/*.png"


X = []
Y = []


def inicial_for_validation(theta, alpha, it):
    X1 = []
    Y1 = []

    arquivos_de_gatos1 = "./data/train/testando/cat2/*.png"
    arquivos_nao_gatos1 = "./data/train/testando/noncat2/*.png"
    for arquivo_de_gato1 in glob.glob(arquivos_de_gatos1):
        imagem = cv2.imread(arquivo_de_gato1)
        imagem = np.reshape(imagem, (64*64*3))
        X1.append(imagem)
        Y1.append(1)
```

```
31
32      for arquivo_nao_gato1 in glob.glob(arquivos_nao_gatos1):
33          imagem = cv2.imread(arquivo_nao_gato1)
34          imagem = np.reshape(imagem, (64*64*3))
35          X1.append(imagem)
36          Y1.append(0)
37
38      X1 = np.asarray(X1)
39      Y1 = np.asarray(Y1)
40      X1 = X1 / 255
41
42      X1 = np.insert(X1, obj=0, values=1, axis=1)
43      Y1 = np.expand_dims(Y1, axis=1)
44
45      print("Fotos com gatos ={}, fotos sem gatos={}".format(
46          np.sum(Y == 1), np.sum(Y != 1)))
47
48      J = np.zeros((it, 1))
49
50      acuracia = np.zeros((it, 1))
51
52      m = len(Y)
53
54      H_theta = np.ones((X.shape[0], 1))
55      for i in range(it):
56          # equa  o da reta
57          Z = np.dot(X, theta)
58
59          # fun  o de ativa  o
60          H_theta = sigmoid(Z)
61
62          # Loss
63          J[i] = 1/m * np.sum(- Y * np.log(H_theta) -
64                              (1-Y) * np.log(1 - H_theta))
65
66          e = H_theta - Y
67
68      predicao = H_theta >= 0.5
69      predicao = np.around(H_theta)
70      acuracia = np.sum(Y == predicao)/len(Y)
71      for i in range(0, len(X)):
72          tmp = X1[i, 1:]*255
73          tmp = np.reshape(tmp, (64, 64, 3))
74          tmp = np.uint8(tmp)
75          salva_imagem_com_predicao(tmp,
76                                    "resultados2/i{}.png".format(
77                                        i),
78                                    H_theta[i, 0])
79
```

```
80
81  for arquivo_de_gato in glob.glob(arquivos_de_gatos):
82      imagem = cv2.imread(arquivo_de_gato)
83      imagem = np.reshape(imagem, (64*64*3))
84      X.append(imagem)
85      Y.append(1)
86
87  for arquivo_nao_gato in glob.glob(arquivos_nao_gatos):
88      imagem = cv2.imread(arquivo_nao_gato)
89      imagem = np.reshape(imagem, (64*64*3))
90      X.append(imagem)
91      Y.append(0)
92
93
94  X = np.asarray(X)
95  Y = np.asarray(Y)
96
97
98  # normalizing x
99  X = X / 255
100
101 X = np.insert(X, obj=0, values=1, axis=1)
102 Y = np.expand_dims(Y, axis=1)
103
104
105 print("Fotos com gatos ={}, fotos sem gatos={}".format(
106     np.sum(Y == 1), np.sum(Y != 1)))
107
108 # criando theta com sendo uma coluna com o numero de linha de X
109 theta = np.zeros((X.shape[1], 1))+0.00001
110
111 alpha = 0.0009991
112 it = 100000
113
114 J = np.zeros((it, 1))
115
116 acuracia = np.zeros((it, 1))
117
118 m = len(Y)
119
120 H_theta = np.ones((X.shape[0], 1))
121
122
123 def sigmoid(Z):
124     return 1/(1+np.exp(-Z))
125
126
127 # treino
128 for i in range(it):
```

```
    # equa  o da reta
    Z = np.dot(X, theta)

    # fun  o de ativa  o
    H_theta = sigmoid(Z)

    # Loss
    J[i] = 1/m * np.sum(- Y * np.log(H_theta) - (1-Y) * np.log(1
        - H_theta))

    e = H_theta - Y

    # atualiza  o do theta
    theta = theta - (alpha * (1/m)*(np.dot(X.T, e)))

    predicao = H_theta >= 0.5
    predicao = np.around(H_theta)
    acuracia = np.sum(Y == predicao)/len(Y)


for i in range(0, len(X)):
    tmp = X[i, 1:]*255
    tmp = np.reshape(tmp, (64, 64, 3))
    tmp = np.uint8(tmp)
    salva_imagem_com_predicao(tmp,
                              "resultados/imagem_{}.png".format(
                                 i),
                              H_theta[i, 0])

plt.title("Loss X Updates")
plt.xlabel('iteracoes')
plt.ylabel('J')
plt.plot(J)

plt.show()

inicial_for_validation(theta, alpha, it)
print("fim")
```

4. Comparação dos Resultados Obtidos

4.1. Primeira tentativa

Como primeiro teste foi utilizado o, alpha, $a = 0.0000991$ e o, número de iterações, $it = 100$, esses valores foram escolhidos de maneira aleatória.Logo abaixo ao gráfico estão as imagens geradas no treino e na validação, sendo a com menor resultado na imagem com gato e de maior resultado com a imagem sem gatos.

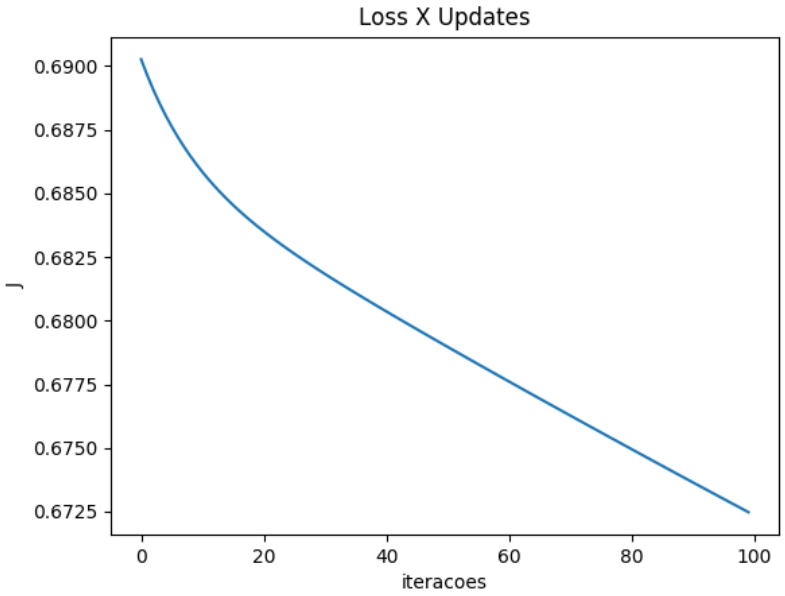

Figure 1. Gráfico 1-Treinamento

4.2. Segunda tentativa

No segundo teste foi mantido o alpha em $a = 0.0000991$ e houve o aumento do$it = 100000$. E pode-se notar de acordo com o gráfico 2 que a convergência foi maior ao aumentar o número de iterações, porém ainda está longe de ser satisfatória.

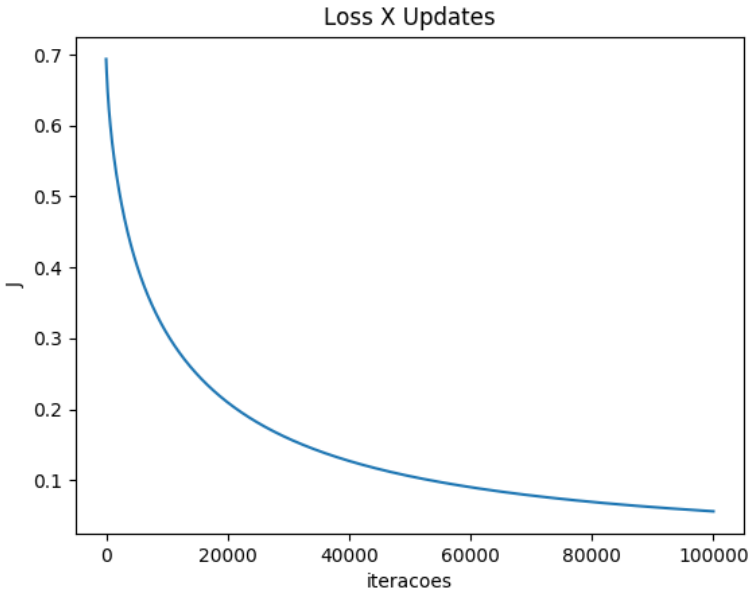

Figure 2. Gráfico 2- Treinamento

4.3. Terceira Tentativa

Na terceira tentativa houve a diminuição do alpha para $a = 0.0000091$ com a finalidade de que ao diminuir o alpha a convergência seria mais rápida,já que anteriormente houve o auteração apenas no número de iterações.Nessta tentativa manteve-se o número de iterações $it = 100000$, como é possivel perceber no gráfico.

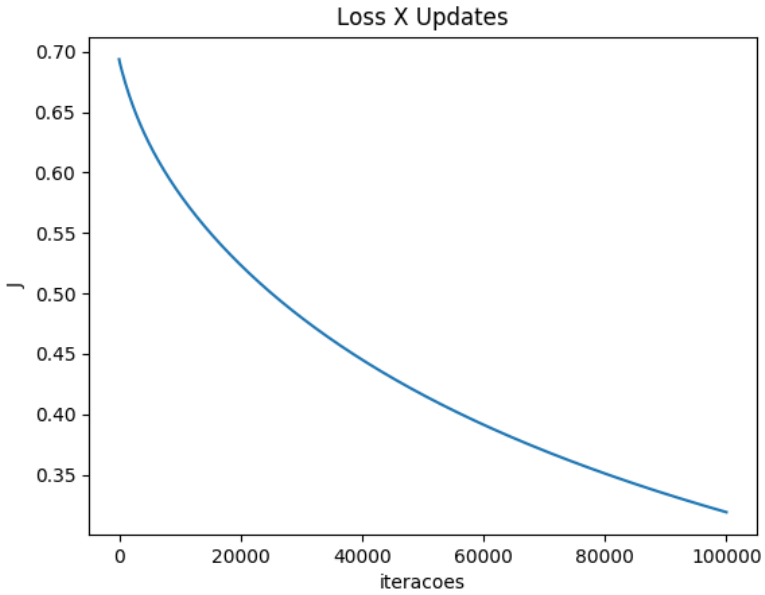

Figure 3. Gráfico 3- Treinamento

4.4. Tentativa Final

Na tentativa final podemos observar que houve a convergência total do nosso custo. Para isso utilizamos o $a = 0.0009991$ e o $it = 100000$. E é interessante observar que na validação o resultado foi tão satisfatório quanto no treino ao analizar as imagens geradas.

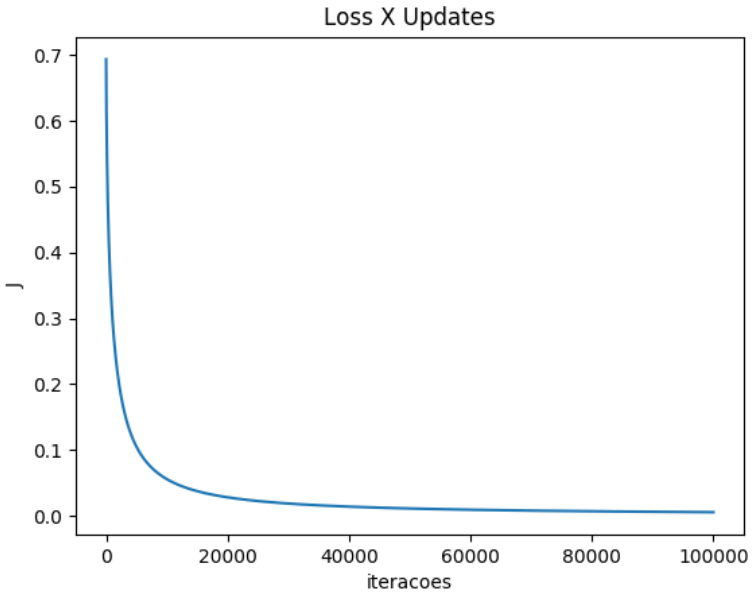

Figure 4. Gráfico 4- Treinamento