

EPUSP - Escola Politécnica da Universidade de São Paulo
PCS3438 - Inteligência Artificial
Machine Learning
2019

Lista de exercícios



Prof. Dr. Eduardo Raul Hruschka

Rafael Seiji Uezu Higa

NUSP: 9836878

Douglas Carvalho Ramos

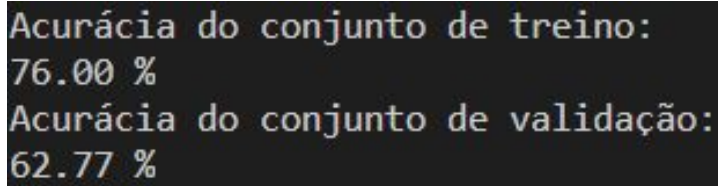
NUSP: 9853300

Igor Manuel Teixeira Ortega

NUSP: 9763071

Questão 1 - Naive Bayes

Resultado:



```
Acurácia do conjunto de treino:
76.00 %
Acurácia do conjunto de validação:
62.77 %
```

Código:

```
import numpy as np
import math

dataset = np.genfromtxt("class01.csv", delimiter=",")

trainingNumber = 350
featureNumber = np.shape(dataset)[1]-1 # = 100

trainingSet = dataset[1:trainingNumber+1,:] #350 exemplos de treino
testSet = dataset[trainingNumber+1:,:] #o restante: 750 exemplos

#Matrizes auxiliares para executar os cálculos
featureGroups = np.zeros((2, 10, featureNumber))
gaussianDenominator = np.zeros((10, featureNumber)) #10 x 100
gaussianExponent = np.zeros((10, featureNumber))
auxMat = np.zeros((10, 100))
labelCounter = np.zeros(10)
probbes = np.zeros(10)

for i in range(0, 10): #Para cada classe...
    #ver se o dado valor pertence à classe...
    classFilter = dataset[1:trainingNumber + 1, 100] == i

    #somar quantos valores pertenciam à dada classe (computo da frequencia da classe)
    labelCounter[i] = sum(dataset[1:trainingNumber + 1, 100] == i)

    for j in range(0, featureNumber): #calculo de media e desvio padrao para essa classe
        featureGroups[0, i, j] = np.mean(trainingSet[classFilter, j])
        featureGroups[1, i, j] = np.std(trainingSet[classFilter, j])

labelCounter /= trainingNumber

for counter in range(0, 2):
    accTest = 0

    #counter = 0 analisa estatisticas do training set, e = 1, as do validation set
    #notar que os valores p/ calcular gaussianas já foram calculados acima ao varrer o dataset
```

```

if counter == 0:
    exampleAmount = 350
else:
    exampleAmount = np.shape(testSet)[0]

for k in range(0,exampleAmount): #varrendo cada exemplo e calculando gaussiana
    prob = np.ones( (10,100) )

    #gaussiana: denominador da formula
    gaussianDenominator[:,:] = math.sqrt(2 * math.pi) * featureGroups[1,[:, :]]

    if counter == 0:
        #matriz auxiliiliar para calculo somente
        auxMat[:,] = np.copy(trainingSet[k,:100].T)

    else:
        auxMat[:,] = np.copy(testSet[k,:100].T)

    #gaussiana: expoente do "e"
    gaussianExponent[:,:] = (-1) * np.power(auxMat[:,:] - featureGroups[0,[:, :]], 2) / (2 *
pow(featureGroups[1,[:, :]],2)) #é tudo na mesma linha, ficou quebrado aqui no relatório!

    #valor da probablilidade
    prob[:,:] *= 1 / gaussianDenominator[:,:] * np.exp(gaussianExponent[:,:])

    probbes = np.prod(prob,axis=1) * labelCounter
    temp = sum(probbes)
    probbes = probbes / temp

    if counter == 0:
        if trainingSet[k,100] == np.argmax(probbes):
            accTest += 1
    else:
        if testSet[k,100] == np.argmax(probbes):
            accTest += 1

if counter == 0:
    print("Acurácia do conjunto de treino:")
else:
    print("Acurácia do conjunto de validação:")

temp2 = accTest / exampleAmount * 100
print("%.2f" % temp2, "%")

```

Questão 2 - Knn

Resultado:

```
0.7633333333333333
0.7033333333333334
0.7166666666666667
0.73
0.7566666666666667

acurácia média:
73.40 %
```

Código:

```
import numpy as np

dataset = np.genfromtxt("class02.csv", delimiter=",")

datasetSize = np.shape(dataset)[0] - 1          # Tamanho do Dataset
testSetSize = int(datasetSize / 5)              # Tamanho do TestSet
trainingSetSize = datasetSize - testSetSize     # Tamanho do TrainingSet

trainingSet = np.zeros((trainingSetSize, 101))
testSet = np.zeros((testSetSize, 101))
normSet = np.zeros(trainingSetSize)             # array para computo das normas
acc = np.zeros(5)                               # array para armazenar as acuracias de cada treino
normElements = np.zeros((trainingSetSize, 100))

for i in range(0, 5):
    testSet = dataset[1 + i * testSetSize : (i + 1) * testSetSize + 1][:]

    #Estratégia para separar o um quinto do dataset que é validation set do resto que é training
    set
    a = dataset[1 : (i * testSetSize) + 1][:]
    b = dataset[1 + (i+1) * testSetSize : datasetSize + 1][:]
    trainingSet = np.concatenate((a,b), axis=0)

    for j in range(0, testSetSize):              #varrendo para cada exemplo
        normElements[:, 0:] = testSet[j, 0:100]

        normSet = np.linalg.norm(normElements - trainingSet[:, 0:100], axis=1) #calculo de norma
```

```

normIndex = np.argsort(normSet)      #definir a menor norma

votes = trainingSet[normIndex[0:10],100]  #computo das classes

#computo do resultado da eleição
election, oi = np.histogram(votes, bins=[0.0, 1.0, 2.0, 3.0, 4.0, 5.0])

result = np.argmax(election)

if votes[result] == testSet[j][100]:
    acc[i] += 1

print(acc[i] / testSetSize)

average = sum(acc/testSetSize)/5*100
print("\nacurácia média:\n%.2f" % average, "%")

```

Questão 3 - LASSO

Resultado:

```

RMSE do Training Set:
19.206305150530632
RMSE do Validation Set:
[28.4362569]

```

Código:

```

import numpy as np

dataset = np.genfromtxt("reg01.csv",delimiter=",")

phiMatrix = np.zeros((np.shape(dataset)[0] - 2, 11))  #999x11
ErroQuadTraining = 0
ErroQuadValidation = 0
alpha = 1
phiMatrix[0:,0] = np.full_like(phiMatrix[0:,0],1)
temp2 = np.zeros((11,1))

for i in range(0,np.shape(dataset)[0] - 2): #dataset[i,:] é a validação!
    #Matrix phi
    phiMatrix[0:,1:] = np.delete(dataset[1:,:10], i, axis=0)  #999x11

```

```

#Matriz t, excluindo-se o ruído gaussiano (observação determinística)
tMatrix = np.delete(dataset[1:,10], i,axis=0)    #999x1

#Cálculo do parametro w, na variável WML

w1 = np.dot(phiMatrix.T,phiMatrix)
w2 = alpha * np.identity(11) + w1
w3 = np.linalg.inv(w2)
w4 = np.dot(w3,phiMatrix.T)
WML = np.dot(w4,tMatrix)

temp = tMatrix - (np.dot(WML.T, phiMatrix.T)).T    #999x1 - ((1x11) * (11x999))'

temp *= temp
WMLAbs = np.absolute(WML)

ErroQuadTraining += np.sqrt(1/999*(sum(temp) + alpha * sum(WMLAbs)))

temp2[0,0] = 1
temp2[1:11,0] = dataset[i+1,0:10]

ErroQuadValidation += np.sqrt((np.power(dataset[i+1,10] - np.dot(WML.T,temp2) , 2) + alpha *
sum(WMLAbs)))    #é tudo na mesma linha, ficou quebrado aqui no relatório!

ErroQuadTraining /= (np.shape(dataset)[0] - 2)
ErroQuadValidation /= (np.shape(dataset)[0] - 2)

print("RMSE do Training Set:")
print(EroQuadTraining)
print("RMSE do Validation Set:")
print(EroQuadValidation)

```

Questão 4 - Árvore de Regressão

Resultado:

MAE do Training Set: 0.0

MAE do Test Set: 45.586111619199826

Código:

```

from sklearn import tree
import pandas as pd
from sklearn.model_selection import cross_validate

```

```

data = pd.read_csv('reg02.csv') # read the data fro .csv
X = data.iloc[:, :-1].to_numpy() # features = all columns but the last one
y = data['target'].to_numpy()    # target

# build the classifier
clf = tree.DecisionTreeRegressor() # default split criteria is already mse

# Get cross validation scores with mean_absolute_error score criteria and k=5
# this operation also trains the classifier
cv_information = cross_validate(clf, X, y,
                                scoring='neg_mean_absolute_error',
                                cv=5,
                                return_train_score=True)

# Media MAE base de treinamento
mae_train_set = abs(pd.Series(cv_information['train_score']).mean())

# media MAE base de testes
mae_test_set = abs(pd.Series(cv_information['test_score']).mean())

print(f"MAE do Training Set: {mae_train_set}",
      f"MAE do Test Set: {mae_test_set}", sep="\n")

```

Questão 5 - Verdadeiro ou Falso

Respostas:

V; V; V; F; V; F; V; V

1- Quando ajustamos um modelo linear, geralmente supomos que os erros tem distribuição normal e são independentes e identicamente distribuídos (i.i.d.).

Verdadeiro. Trata-se de uma hipótese claramente ideal, mas razoável como premissa para assumirmos que o modelo linear pode ser utilizado.

2- Quando ajustamos um modelo de regressão, podemos utilizar os valores preditos e os resíduos do modelo para avaliar se o modelo se adequa bem aos dados.

Verdadeiro. Primeiramente, é óbvio que os valores preditos são adequados para avaliar se o modelo se adequa bem aos dados. Basta compará-los com os dados que treinaram esse modelo (e com dados de fora desse modelo, como validação, também!) e ver se são adequados. A questão mais importante é a métrica que se usa para fazer tal comparação. Nesse contexto, a análise de resíduos, se faz adequada. Trata-se da verificação das diferenças entre os valores medidos e o valor correspondente do modelo (por exemplo, quando se calcular o erro quadrático). Quanto menores os resíduos (menores erros quadráticos), mais próximos dos resultados do dataset os resultados do modelo estão.

3- O coeficiente de determinação (r^2) indica, em termos percentuais, quanto da variabilidade da variável resposta é explicada pelas covariáveis do modelo.

Verdadeiro. O coeficiente de determinação é determinado por:

$$R^2 = \frac{SQ_{\text{exp}}}{SQ_{\text{tot}}} = 1 - \frac{SQ_{\text{res}}}{SQ_{\text{tot}}}.$$

onde:

$$SQ_{\text{tot}} = \sum_{i=1}^n (y_i - \bar{y})^2,$$

(soma dos quadrados das diferenças entre valor do dataset e média das observações)

$$SQ_{\text{res}} = \sum_{i=1}^n (y_i - \hat{y}_i)^2,$$

(soma dos quadrados das diferenças entre valor do dataset e valor previsto pelo modelo)

$$SQ_{\text{exp}} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$$

(soma dos quadrados das diferenças entre valor do dataset e valor da média das observações).

Uma vez que SQ_{res} indica a variabilidade do modelo em relação ao dataset (e portanto indicando a parcela de variabilidade explicada pelas covariáveis do modelo) e que SQ_{tot} representa a variabilidade total, tanto por parte do modelo quanto por parte do dataset, tem-se que faz sentido r^2 indicar a porcentagem de variabilidade da resposta que é devida ao próprio modelo.

4- Os modelos de regressão não são afetados por observações atípicas (outliers) e valores faltantes.

Falso. Evidentemente, observações atípicas podem alterar o modelo encontrado, visto que tal modelo é obtido da minimização do erro (quadrático, na regressão linear, por exemplo) entre o modelo e as observações. O modelo tenderia a “desviar” o comportamento do modelo em favor dessa observação atípica. Evidentemente, caso haja uma proporção muito maior de pontos típicos do que atípicos, provavelmente o modelo ainda apresentará bons resultados. Mesmo assim, não se deve negar a tendência de o modelo ser negativamente afetado nesse caso.

Da mesma forma que observações atípicas implicam no modelo de regressão não retratar bem a realidade, a falta valores adequados em um dataset, que representam a realidade, pode fazer com que a regressão não se aproxime adequadamente do comportamento adequado. Por exemplo, um dataset poderia aproximar uma reta por regressão, sendo que, ao se inserirem dados outrora faltantes, fosse visível que uma regressão quadrática se encaixaria melhor no problema.

5- Considerando um modelo de regressão simples, temos que o coeficiente associado à covariável representa o grau de inclinação da reta.

Verdadeiro. Em uma regressão linear, por exemplo, $y = ax + b$, o coeficiente associado à covariável se trata do coeficiente a , que é a inclinação da reta.

6- Para efetuar regressão com o algoritmo KNN, pode-se fazer uma votação simples dos valores dos k vizinhos encontrados.

Falso. Em uma regressão com Knn, não faz sentido falar em votos, uma vez que estamos tratando de variáveis contínuas. A abordagem a ser adotada é, com os k vizinhos mais próximos segundo a métrica desejada (dist. euclidiana, de Manhattan, etc), calcular a média ponderada (ou não) dos valores preditos para esses k vizinhos mais próximos:

$$y = f(\mathbf{x}_q) = \frac{\sum_{i=1}^k w_i f(\mathbf{x}_i)}{\sum_{i=1}^k w_i}$$

y é o valor predito para o nó no qual se aplica Knn, calculado pela média ponderada dos k vizinhos mais próximos.

7- Para melhor desempenho da árvore de regressão, pode-se utilizar regressões lineares em suas folhas para previsão do valor final.

Verdadeiro. A árvore de regressão tem como única função classificar exemplos semelhantes do dataset. Entretanto, exemplos dentro de uma mesma folha ainda

possuem suas diferenças nos seus valores y e diferenças nos valores de suas features, de forma que seria interessante, em termos de obtenção de melhores acurácias preditivas do modelo (apesar de ser necessário maior custo computacional para tal), aplicar regressões lineares em cada folha, aumentando ainda mais a acurácia preditiva do modelo.

8- No algoritmo Random Forest para regressão, o valor predito é obtido pela média dos valores encontrados em cada árvore.

Verdadeiro. Trata-se, exatamente, de como funciona o algoritmo random forest. Várias árvores são executadas, de forma a minimizar o problema da instabilidade dos modelos de árvore devido a flutuações amostrais (a árvore é reiniciada com datasets diferentes e daí o resultado final é tido como a média dos resultados dessas diversas árvores geradas).