Aprendizagem de Máquina - Relatório
Prática 1

1 - Questões

Questão 01

Inicialmente foi calculado o valor da matriz de covariância, utilizando a função COV da biblioteca numpy, método este que foi chamado de padrão. Após isso, foi feito o cálculo desta matriz de acordo com os métodos especificados na descrição do trabalho.

Abaixo está listado os resultados obtidos para cada método.

Método Padrão

Matriz de covariância

[[297.09763261, 108.54231786, 229.39228019, 188.55531508, -56.80491868, 413.51370361]
 [108.54231786, 100.1666743, 80.36217231, 8.37564359, 4.35411976, 149.55768086]
 [229.39228019, 80.36217231, 344.25328916, 149.03010824, -19.85222026, 371.89790796]
 [188.55531508, 8.37564359, 149.03010824, 180.17967179, -61.15903876, 263.95602326]
 [-56.80491868, 4.35411976, -19.8522202, -61.15903876, 177.35253093, -13.03739632]
 [413.51370361, 149.55768086, 371.89790796, 263.95602326 , -13.03739632, 1410.68047512]]

Método 1

Matriz de covariância

[[296.13925315, 108.19218135, 228.6523051, 187.94707213, -56.62167701, 412.17978844]
 [108.19218135, 99.843556, 80.10293949, 8.34862539, 4.34007421, 149.07523673]
 [228.6523051, 80.10293949, 343.14279468, 148.54936595, -19.78818084, 370.69823729]
 [187.94707213, 8.34862539, 148.54936595, 179.59844704, -60.96175154, 263.10455222]
 [-56.62167701, 4.34007421, -19.78818084, -60.96175154, 176.78042599, -12.9953402]
 [412.17978844, 149.07523673, 370.69823729, 263.10455222, -12.9953402, 1406.12989294]]

Método 2

Matriz de covariância

[[296.13925315, 108.19218135, 228.6523051, 187.94707213, -56.62167701, 412.17978844]
 [108.19218135, 99.843556, 80.10293949, 8.34862539, 4.34007421, 149.07523673]
 [228.6523051, 80.10293949, 343.14279468, 148.54936595, -19.78818084, 370.69823729]
 [187.94707213, 8.34862539, 148.54936595, 179.59844704, -60.96175154, 263.10455222]
 [-56.62167701, 4.34007421, -19.78818084, -60.96175154, 176.78042599, -12.9953402]
 [412.17978844, 149.07523673, 370.69823729, 263.10455222, -12.9953402, 1406.12989294]]

Método 3

[[296.13925315, 108.19218135, 228.6523051, 187.94707213, -56.62167701, 412.17978844]
 [108.19218135, 99.843556, 80.10293949, 8.34862539, 4.34007421, 149.07523673]
 [228.6523051, 80.10293949, 343.14279468, 148.54936595, -19.78818084, 370.69823729]
 [187.94707213, 8.34862539, 148.54936595, 179.59844704, -60.96175154, 263.10455222]
 [-56.62167701, 4.34007421, -19.78818084, -60.96175154, 176.78042599, -12.9953402]
 [412.17978844, 149.07523673, 370.69823729, 263.10455222, -12.9953402, 1406.12989294]]

Como podemos observar, os resultados do cálculo da matriz de covariância pelos métodos diferem da matriz original(utiliza o método padrão) somente nas últimas casas decimais. Assim podemos considerar que elas são iguais(utilizando um arredondamento dos

valores). Portanto podemos concluir que independentemente do método que for usado, o resultado final será o mesmo.

Questão 02

De acordo com uma das matrizes de covariância calculadas na questão anterior, foram calculados os coeficientes de correlação entre todas as variáveis aleatórias envolvidas. Ou seja, para cada valor da mesma foi calculado sua correlação com os demais elementos da matriz.

Não foi possível utilizar o método 1, visto que a biblioteca numpy não possui a função COR. Porém, os cálculos foram realizados para os demais métodos e os resultados obtidos estão listados abaixo.

Método 2

Utilizando a função da biblioteca numpy CORRCOEF.

Matriz de coeficientes:

```
[[ 1,         0.62919878,  0.71728237,  0.81495999, -0.2474672,   0.63874275]
 [ 0.62919878,  1,         0.43276387,  0.06234529,  0.03266781,  0.39786228]
 [ 0.71728237,  0.43276387,  1,         0.59838689, -0.08034361,  0.53366701]
 [ 0.81495999,  0.06234529,  0.59838689,  1,        -0.34212835,  0.52355746]
 [-0.2474672,   0.03266781, -0.08034361, -0.34212835,  1,        -0.02606501]
 [ 0.63874275,  0.39786228,  0.53366701,  0.52355746, -0.02606501,  1       ]]
```

Método 3

Utilizando a equação proposta pelo método.

Matriz de coeficientes:

```
[ 0.30628062  0.0953995   0.21343629  0.20398395 -0.0539884   0.31428415]
[ 0.0953995   0.075058    0.06374819  0.00772506  0.00352811  0.09690987]
[ 0.21343629  0.06374819  0.28909243  0.14551253 -0.01702914  0.25510881]
[ 0.20398395  0.00772506  0.14551253  0.2045502  -0.06099751  0.21052373]
[-0.0539884   0.00352811 -0.01702914 -0.06099751  0.15539827 -0.00913519]
[ 0.31428415  0.09690987  0.25510881  0.21052373 -0.00913519  0.79044832]
```

Observando os resultados, vemos que o coeficiente de correlação de um atributo para ele mesmo é igual a 1. O método 2 retorna valores com uma taxa de aproximação maior e faz uso da variância para realizar este cálculo, enquanto que o método 3 utiliza o desvio padrão.

Portanto, por mais que exista uma diferença de valores em algumas casas decimais entre os mesmo, podemos concluir que eles possuem resultados semelhantes.

Questão 3:

Gráfico da distribuição dos coeficientes para um par de variáveis que apresentam correlação positiva.

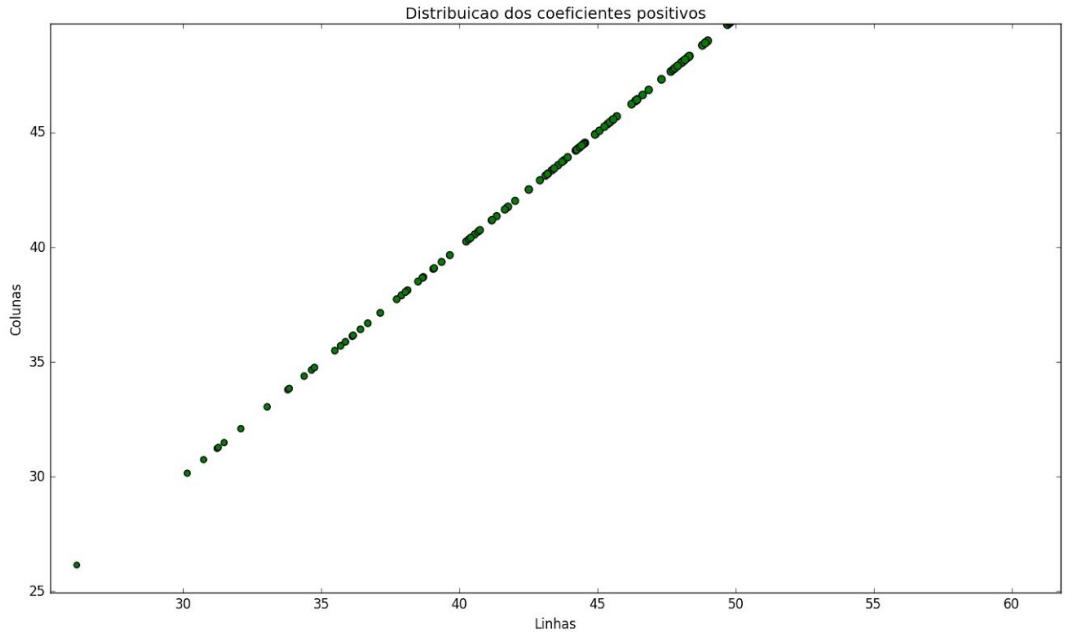

Figura 1: Par de variáveis positivas.

Gráfico da distribuição dos coeficientes para um par de variáveis que apresentam correlação negativa.

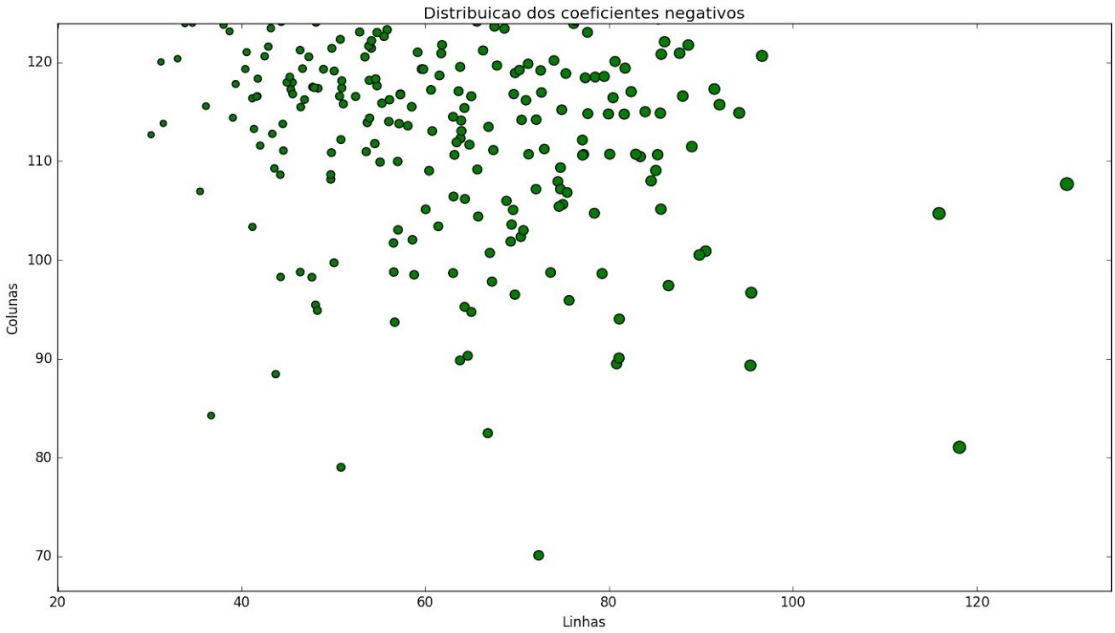

Figura 2: Par de variáveis negativas.

Analisando o gráfico podemos inferir que um par de variáveis que possuem um coeficiente de correlação positiva, se comportam de forma uniforme e crescente. Porém, um par de variáveis que possuem um coeficiente de correlação negativa se comporta de modo não uniforme (Ficam mais dispersos) e variam entre crescimento ou decrescimento. De modo

que seu gráfico não segue um padrão e fica subentendido que estão muitos mais dispersos entre si.

Essa análise do gráfico permite identificar qual a relação dos dados entre si, e com isso conseguir obter informações suficientes para decidir sobre qual atributo usar um método de descorrelação de atributos e se é necessário usar tal técnica.

2 - Código Fonte

```
#-*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt

# Método 1
def metodo1(data, media):
  covariancia = 0
  # Cálculo das diferenças
  for i in data:
      diferenca = (i.reshape(6, 1) - media)
      covariancia += np.dot(diferenca, diferenca.T)

  # Calculando o produto
  covariancia = covariancia * (1.0 / float(len(data)))
  return covariancia

# Método 2
def metodo2(data, media):
  # Matriz de correlação
  correlacao = 0
  for i in data:
      correlacao += np.dot(i.reshape(6, 1), i.reshape(6, 1).T)
  correlacao *= (1.0 / float(len(data)))

  covariancia = (correlacao - np.dot(media, media.T))
  return  covariancia

# Método 3
def metodo3(data, media):
  # Matriz de correlação
  correlacao2 = 0
  for n in xrange(0, len(data)):
      x = data[n].reshape((6, 1))
      correlacao2 = (n / float(n + 1) * correlacao2) + ((1 / float(n + 1)) * np.dot(x, x.T))

  covariancia = (correlacao2 - np.dot(media, media.T))
  return covariancia
```

```
def plotarGrafico(linha, coluna, tipo):
  plt.scatter(x=linha, y=coluna, s=linha ,c='g')
  plt.xlabel("Linhas")
  plt.ylabel("Colunas")
  plt.title("Distribuicao dos coeficientes " + tipo)
  plt.axis([1,len(linha), 1, coluna[-1]])
  plt.show()

# Retorna o primeiro elemento positivo que a matriz tiver
def getPositivo(matriz):
  n = len(matriz)
  i = 0
  while i < n:
     j = 0
     while j < n:
        if matriz[i][j] > 0:
           return  (i,j)
        j += 1
     i += 1
# Retorna o primeiro elemento negativo que a matriz tiver
def getNegativo(matriz):
  n = len(matriz)
  i = 0
  while i < n:
     j = 0
     while j < n:
        if matriz[i][j] < 0:
           return (i, j)
        j += 1
     i += 1

# Lendo o arquivo
data = np.genfromtxt("coluna.dat", delimiter=",")

# Calculando a Matriz de Covariância pelos métodos
#Questão 1
# Matriz de covariância 1
matriz_cov = np.cov(data.transpose())

# Média geral
media = np.mean(data, axis=0).reshape((6, 1))
print  "\nQuestão 1 \n"
print "Método Padrão\n"
for linha in  matriz_cov:
  print linha
```

```
print "\nMétodo 1\n"
for linha in metodo1(data, media):
  print linha

print "\nMétodo 2\n"
for linha in metodo2(data, media):
  print linha

print "\nMétodo 3"
for linha in metodo3(data, media):
  print linha

# Questão 02
print  "\nQuestão 2 \n"

# Método 1
# A biblioteca Numpy não possui a função CORR..

# Método 2
cor_numpy = np.corrcoef(data.transpose())
print  "\nMétodo 2 - Matriz de Coeficientes de Correlação.\n"
for linha in cor_numpy:
  print linha

# Método 3
coeficientes = np.zeros((6,6))

# Utilizando a matriz de covariância do método 2
covariancia = metodo2(data, media)

for i in xrange(0,len(covariancia)):
  for j in xrange(0, len(covariancia)):
      d_i = [x for x in data[i]]
      d_j = [x for x in data[j]]
      coeficientes[i][j] = (covariancia[i][j] / (np.dot(np.std(d_i), np.std(d_j))))

print  "\nMétodo 3 - Matriz de Coeficientes de Correlação.\n"
for linha in coeficientes:
  print linha

# Questão 3
positivo = getPositivo(coeficientes)
negativo = getNegativo(coeficientes)
```

```
positivo_linha = [data[i][positivo[0]] for i in range(len(data))]
positivo_coluna = [data[i][positivo[1]] for i in range(len(data))]

#plotarGrafico(positivo_linha, positivo_coluna, "positivos")

negativo_linha = [data[i][negativo[0]] for i in range(len(data))]
negativo_coluna = [data[i][negativo[1]] for i in range(len(data))]

#print coeficientes

#plotarGrafico(negativo_linha, negativo_coluna, "negativos")
```