Exercicio № 1

Diego Alonso Fernandez Merjildo, Univesidade de Campinas 26/09/2016

Para este exercicio foi desenvolvido o script chamado de ex_01.py (Codigo Python 4, na parte final do relatorio) que inclui a solucão de todos os problemas abaixo detalhados. Neste relatorio mostamos uma parte do codigo (função) para cada problema e o resultado obtido.

Problema 1

Faca o PCA dos dados (sem a última coluna). Se voce quiser que os dados transformados tenham 80% da variância original, quantas dimensões do PCA vc precisa manter?

Gere os dados transformados mantendo 80% da variância. (Atenção este passo não é 100% correto do ponto de vista de aprendizado de maquina. Não repita este passo em outras atividades).

Considere as primeiras 200 linhas dos dados como o conjunto de treino, e as 276 ultimas como o conjunto de dados

Solução

Para achar o numero de componentes com uma determinada variância usamos a Descomposição de valores singulares que foi implementada na função *chooseComponentsNumber(matrix, percent)*.

A a função *applyPCA(data, numComponents)*, aplica PCA no conjunto de dados e recebe como parametro o numero de componentes (Dimensão PCA) do resultado. A função *getClassTrainTest(classList)* separa o conjunto de dados em dois partes, o primeiro para treinamento e segundo para teste.

Listing 1: Codigo em Python – Função para escolher a dimensão PCA aplicar PCA e separar dados de treinamento e teste

```
1  def chooseComponentsNumber(matrix, percent):
2      print "\n---- PCA - Choose components number ----"
3      print "Variance :",  percent
4      mat = np.matrix(matrix) * np.matrix(matrix).transpose()
5      U,S,V = np.linalg.svd(mat)
6      #print U.shape, S.shape, V.shape
7      s_sum_all = sum(S)
8      totalComponents = matrix.shape[1]
9      num = totalComponents
10     for i in range(totalComponents):
```

```
11          if sum(S[0:i]) / s_sum_all >= percent :
12              print "PCA dimension:",i ,"with variance =", sum(S[0:i]) / ↩
                    s_sum_all
13              num = i
14              break
15      return num
16
17   def getClassTrainTest(classList):
18      print "\n---- Get Class Train and Test ----"
19      classListTrain=classList[0:199]
20      classListTest=classList[200:len(classList)]
21      print len(classListTrain), len(classListTest)
22      return [classListTrain, classListTest]
23
24  def applyPCA(data, numComponents):
25      print "\n---- Apply PCA ----"
26      #pca = PCA(n_components=numComponents)
27      pca = PCA(n_components=numComponents)
28      pcaData = pca.fit_transform(data)
29      print pcaData.shape
30      return pcaData
```

Escolhendo a dimensão com variância 80%:

```
1    ---- PCA - Choose components number ----
2    Variance : 0.8
3    PCA dimension: 12 with variance = 0.812994148825
```

Separando dados de treinamento e teste com os dados obtidos do PCA:

```
1
2   ---- Apply PCA ----
3   (475, 12)
4
5   For PCA data
6   ---- Get Train and Test data ----
7   Data Train size: (200, 12)
8   Data Test size: (275, 12)
```

Problema 2

Treine uma regressão logística no conjunto de treino dos dados originais e nos dados transformados. Qual a taxa de acerto no conjunto de teste nas 2 condições (sem e com PCA)?

Solução

Listing 2: Codigo em Python. Treinamento regressão logistica

```
1 def logisticRegression(data, classList):
2     print "\n ---- Logistic Regression ----"
3     logreg = linear_model.LogisticRegression(C=1e5)
4     logreg.fit(data, classList)
5     return logreg
```

Resultados da regresão logistica sem PCA e com PCA:

```
1  ---- Logistic Regression ----
2 Logistic Regression score:  0.767272727273
3
4  ---- Logistic Regression ----
5 PCA ( 80 \%) Logistic Regression score:  0.796363636364
```

Problema 3

Treine o LDA nos conjuntos de treino com e sem PCA e teste nos respectivos conjuntos de testes. Qual a acurácia nas 2 condições?

Solução

Listing 3: Codigo em Python. Treinamento LDA

```
1 def LDA_train(data, classList):
2     print "\n---- LDA -----"
3     clf = LDA()
4     clf.fit(data, classList)
5     return clf
```

Resultados da treinamento LDA sem PCA e com PCA:

```
1 ---- LDA -----
2 LDA score:  0.694545454545
3
4 ---- LDA -----
5 PCA ( 80 %) LDA score:  0.803636363636
```

Problema 4

Qual a melhor combinação de classificador e PCA ou não?

Solução

Os resultados apresentam um melhor desempenho quando aplicado PCA nos dados originais. Inclusive considerando uma variância de 80% o desempenho dos classificadores se manteve superior.

Script completo do exercicio 1

Listing 4: Codigo em Python – ex_01.py Script.

```
1
2  #!/usr/bin/python
3
4  import sys,os,csv
5  import pandas
6  import numpy
7  import numpy as np
8  from sklearn.decomposition import PCA
9  from sklearn import linear_model
10 from sklearn.lda import LDA
11
12 datFileName="data1.csv"
13 dirPath=os.path.dirname(os.path.realpath(__file__))
14 classList=[]
15 data=[]
16
17 ## Load CSV
18 def loadCsvData(fileName):
19     raw_data = open(fileName, 'rb')
20     rawData = pandas.read_csv(raw_data, delimiter=",", skiprows=1)
21     return rawData.values
22
23 def getData(rawData):
24     print "\n---- Getting data from File ----"
25     lineNum = rawData.shape[0]
26     colNum = rawData.shape[1]
27     print "lineNum:", lineNum
28     print "colNum:", colNum
29
30     data = np.array(rawData[0:lineNum, 0:colNum-1])
31     for i in range(lineNum):
```

```
32          classList.append(rawData[i][colNum - 1])
33      return [data, classList]
34
35  def chooseComponentsNumber(matrix, percent):
36      print "\n---- PCA - Choose components number ----"
37      print "Variance :",  percent
38      mat = np.matrix(matrix) * np.matrix(matrix).transpose()
39      U,S,V = np.linalg.svd(mat)
40      #print U.shape, S.shape, V.shape
41      s_sum_all = sum(S)
42      totalComponents = matrix.shape[1]
43      num = totalComponents
44      for i in range(totalComponents):
45          if sum(S[0:i]) / s_sum_all >= percent :
46              print "Nro components:",i ,"with variance =", sum(S[0:i]) / ↩
                    s_sum_all
47              num = i
48              break
49      return num
50
51  def getTrainAndTestData(data):
52      print "\n---- Get Train and Test data ----"
53      data_train = data[0:200]
54      data_test = data[200:data.shape[0]]
55
56      print "Data Train size:", data_train.shape
57      print "Data Test size:", data_test.shape
58      return [data_train, data_test]
59
60  def getClassTrainTest(classList):
61      print "\n---- Get Class Train and Test ----"
62      classListTrain=classList[0:200]
63      classListTest=classList[200:len(classList)]
64      print len(classListTrain), len(classListTest)
65      return [classListTrain, classListTest]
66
67  def applyPCA(data, numComponents):
68      print "\n---- Apply PCA ----"
69      #pca = PCA(n_components=numComponents)
70      pca = PCA(n_components=numComponents)
71      pcaData = pca.fit_transform(data)
72      print pcaData.shape
73      return pcaData
74
75  def logisticRegression(data, classList):
76      print "\n ---- Logistic Regression ----"
77      logreg = linear_model.LogisticRegression(C=1e5)
```

```
78      logreg.fit(data, classList)
79      return logreg
80
81  def LDA_train(data, classList):
82      print "\n---- LDA -----"
83      clf = LDA()
84      clf.fit(data, classList)
85      return clf
86
87  def main(argv=None):
88      if argv is None:
89          arv = sys.argv
90      rawdata = loadCsvData(dirPath + "/" + datFileName)
91      [data, classList] = getData(rawdata)
92      [data_train, data_test] = getTrainAndTestData(data)
93      [classListTrain, classListTest] = getClassTrainTest(classList)
94
95      variance = 80
96      numComponents = chooseComponentsNumber(data_train, float(variance) / ↩
            100)
97      if numComponents == -1 : print "Invalid components number. Exit"; ↩
            return
98
99      pcaData = applyPCA(data, numComponents)
100     print "For PCA data"
101     [pcaDataTrain, pcaDataTest] = getTrainAndTestData(pcaData)
102
103     logreg = logisticRegression(data_train, classListTrain)
104     print "Logistic Regression score: ", logreg.score(data_test, ↩
            classListTest)
105
106     logregPca = logisticRegression(pcaDataTrain, classListTrain)
107     print "PCA (",variance,"%) Logistic Regression score: ", logregPca.↩
            score(pcaDataTest, classListTest)
108
109     clf = LDA_train(data_train, classListTrain)
110     print "LDA score: ", clf.score(data_test, classListTest)
111
112     clfPca = LDA_train(pcaDataTrain, classListTrain)
113     print "PCA (",variance,"%) LDA score: ", clfPca.score(pcaDataTest, ↩
            classListTest)
114
115 if __name__ == "__main__":
116     sys.exit(main())
```