Exercicio 4: KMeans

Diego Fernandez Merjildo

October 31, 2016

1 Descrição do exercicio

Use os dados do arquivo cluster-data.csv (os dados sao uma media de 30 medidas por vez da pessoa 1 do dataset Activity Recognition from Single Chest-Mounted Accelerometer Data Set

1. Rode o kmeans nos dados, com numero de restarts = 5
2. Use alguma metrica interna (algum Dunn, Silhouette, Calinski-Harabaz index) - apenas uma -para escolher o k entre 2 e 10.
3. O arquivo cluster-data-class.csv contem a classe correta de cada ponto. Use alguma medida externa (Normalized/adjusted Rand, Mutual information, variation of information) para decidir no k.
4. Plote os graficos correspondentes das 2 metricas (interna e externa) para os varios valores de k (extra).

2 Resultados

Os resultados foram obtidos rodando o script apresentado na seção 3.
Rodando o script em python obtemos os seguintes valores:

```
--- Metricas Internas ---
         K   Silhouette  Calinski
       k=2   0.583   0.516
       k=3   0.583   0.465
       k=4   0.456   0.474
       k=5   0.444   0.465
       k=6   0.410   0.447
       k=7   0.422   0.437
       k=8   0.408   0.428
       k=9   0.410   0.418
      k=10   0.384   0.413
```

```
best score:  0.583031835373
best metric intern:  Silouette
best k: 2

--- Metricas Externas ---
         K    Homogen  v_meas adj_rand  mutual
       k=2    0.295    0.413    0.365    0.295
       k=3    0.387    0.446    0.465    0.386
       k=4    0.396    0.451    0.467    0.395
       k=5    0.457    0.462    0.496    0.456
       k=6    0.434    0.400    0.352    0.369
       k=7    0.456    0.411    0.366    0.372
       k=8    0.509    0.442    0.384    0.389
       k=9    0.498    0.410    0.317    0.347
      k=10    0.542    0.436    0.341    0.363
best score:  0.541596182352
best metric extern:  Homogeneity
best k:  10
```

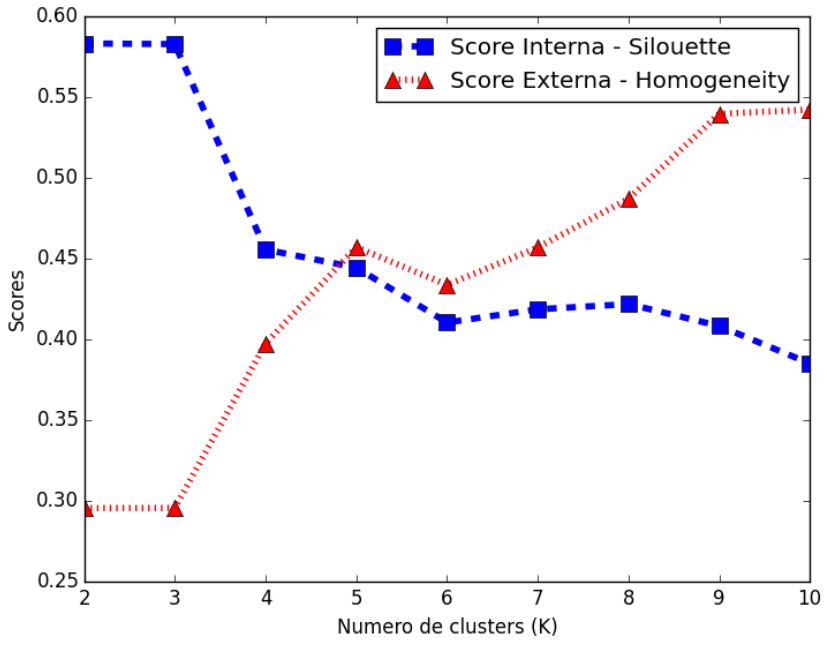

Figure 1: Figura dos scores interno e externo

3 Codigo fonte em python

Listing 1: Codigo em Python

```
#!/usr/bin/python

import sys,os,csv
import pandas
import numpy as np
import math
from sklearn.model_selection import StratifiedKFold
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn import metrics
from sklearn.preprocessing import scale
import matplotlib.pyplot as plt

datFileName="cluster-data.csv"
labelsFileName="cluster-data-class.csv"
dirPath=os.path.dirname(os.path.realpath(__file__))
classList=[]
data=[]

def load_data(fileName):
    raw_data = open(fileName, 'rb')
    #rawData = pandas.read_csv(raw_data, delimiter=" ")
    rawData = pandas.read_csv(raw_data, delimiter=",", skiprows=1)
    return rawData.values

def getData(rawData):
    #print "\n---- Getting data from File ----"
    lineNum = rawData.shape[0]
    colNum = rawData.shape[1]
    data = np.array(rawData[0:lineNum, 0:colNum-1])
    for i in range(lineNum):
        classList.append(rawData[i][colNum - 1])
    return [data, np.array(classList) ]

def get_labels(fileName):
    labelData = load_data(dirPath + "/" + fileName)
    labels = labelData[:,0].clip(min=0)
    return np.array(labels)


def bench_k_means_inter(estimator, name, data):
    estimator.fit(data)
    best_score = 0.0
```

```
44      best_metric = ''
45      scores = [metrics.silhouette_score(data, estimator.labels_,↩
           metric='euclidean', sample_size=5416),
46                metrics.calinski_harabaz_score(data, estimator.labels_↩
                  )/10000 ]
47      print('% 9s   %.3f   %.3f'
48            % (name, scores[0], scores[1]))
49
50      if scores[0] > best_score:
51          best_score = scores[0]
52          best_metric = "Silouette"
53
54      if scores[1] > best_score:
55          best_score = scores[1]
56          best_metric = "calinski"
57
58      return [best_score, best_metric]
59
60  def bench_k_means_ext(estimator, name, data, labels):
61      estimator.fit(data)
62      best_score = 0.0
63      best_metric = ''
64      scores = [metrics.homogeneity_score(labels, estimator.labels_),
65                metrics.v_measure_score(labels, estimator.labels_),
66                metrics.adjusted_rand_score(labels, estimator.labels_)↩
                     ,
67                metrics.adjusted_mutual_info_score(labels,  estimator.↩
                     labels_) ]
68      print('% 9s   %.3f   %.3f   %.3f   %.3f'
69            % (name, scores[0] , scores[1], scores[2], scores[3]) )
70
71      if scores[0] > best_score:
72          best_score = scores[0]
73          best_metric = "Homogeneity"
74
75      if scores[1] > best_score:
76          best_score = scores[1]
77          best_metric = "v_measure"
78
79      if scores[2] > best_score:
80          best_score = scores[2]
81          best_metric = "adjusted_rand"
82
83      if scores[3] > best_score:
84          best_score = scores[3]
85          best_metric = "adjusted_mutual"
86
87      return [best_score, best_metric]
```

```
88
89  def main(argv=None):
90      if argv is None:
91          arv = sys.argv
92
93      data = load_data(datFileName)
94      labels = get_labels(labelsFileName)
95      data = scale(data)
96
97      best_metric_int = ''
98      best_metric_ext = ''
99      range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10]
100
101     print("--- Metricas Internas ---")
102     print('%9s   %s  %s' % ('K', 'Silhouette', 'Calinski'))
103     best_score = 0.0
104     best_metric = ''
105     best_k = 0
106     for n_clusters in range_n_clusters:
107         new_name = "k=" + str(n_clusters)
108         [last_best_score, last_best_metric] = bench_k_means_inter(↩
                KMeans(init='k-means++', n_clusters=n_clusters, n_init↩
                =5), name=new_name, data=data)
109         if last_best_score > best_score:
110             best_score = last_best_score
111             best_metric_int = last_best_metric
112             best_k = n_clusters
113
114     print "best score: ", best_score
115     print "best metric intern: ", best_metric_int
116     print "best k:", best_k
117
118     best_score=0.0
119     print("")
120     print("--- Metricas Externas ---")
121     print('%9s   %s  %s %s  %s' % ('K', 'Homogen', 'v_meas', '↩
            adj_rand', 'mutual'))
122     for n_clusters in range_n_clusters:
123         new_name = "k=" + str(n_clusters)
124         [last_best_score, last_best_metric] = bench_k_means_ext(↩
                KMeans(init='k-means++', n_clusters=n_clusters, n_init↩
                =5), name=new_name, data=data, labels=labels)
125         if last_best_score > best_score:
126             best_score = last_best_score
127             best_metric_ext = last_best_metric
128             best_k = n_clusters
129
130     print "best score: ", best_score
```

```
131     print "best metric extern: ", best_metric_ext
132     print "best k: ", best_k
133 
134     final_score_int=[]
135     final_score_ext=[]
136 
137     for n_clusters in range_n_clusters:
138         kmeans = KMeans(init='k-means++', n_clusters=n_clusters, ↩
              n_init=5)
139         kmeans.fit(data)
140         final_score_int.append( metrics.silhouette_score(data, ↩
              kmeans.labels_, metric='euclidean', sample_size=5416))
141         final_score_ext.append( metrics.homogeneity_score(labels, ↩
              kmeans.labels_))
142 
143     legen_int = 'Score Interna - ' + str(best_metric_int)
144     legen_ext = 'Score Externa - ' + str(best_metric_ext)
145 
146     plt.plot(range_n_clusters, final_score_int, 'bs--', linewidth↩
          =4, markersize=10, label=legen_int)
147     plt.plot(range_n_clusters, final_score_ext, 'r^:', linewidth=4,↩
          markersize=10, label=legen_ext)
148     plt.xlabel('Numero de clusters (K)')
149     plt.ylabel('Scores')
150     plt.legend()
151     plt.show()
152 
153 if __name__ == "__main__":
154     sys.exit(main())
```