Trabalho 07 - MC886

Lucas Henrique Morais, 136640

Visão Geral

O programa que implementamos para resolver o problema proposto baseia-se em clusterização por DBScan para identificar sub-sequências anômalas das cinco séries analizadas. Supondo-se que cada sequência apresentaria sempre uma única anomalia, implementou-se um algoritmo de busca binária que procura selecionar o parâmetro `eps` do DBScan, que controla a sua facilidade em agrupar sob o mesmo *label* amostras distantes, de modo que apenas o descritor de uma única sub-sequência recebesse um label distinto dos demais. Esse único descritor a receber um label distinto corresponderia então à sub-sequência anômala da série.

Na presente solução, DBScan não opera diretamente sobre as sub-sequências das séries de teste, mas sobre os vetores correspondentes às três primeiras componentes do resultado da aplicação de uma Discrete Cosine Transform sobre cada uma dessas sub-sequências.

Ao rodar-se o programa, exibem-se as cinco séries analizadas com as suas respectivas regiões anômalas identificadas por barras verticais vermelhas.

Código fonte

```
#!/usr/bin/env python
#encoding: utf-8

from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
from scipy.fftpack import dct
from sklearn import metrics
import numpy as np
import csv

def regular_splitter(some_list, split_size):
    return [some_list[i*split_size:(i+1)*split_size] for i in range(0,len(some_list)/split_size)]

def find_outlier(sample_array):
    def num_outliers(eps_val):
        global db
        db = DBSCAN(eps=eps_val).fit(sample_array)
        return sum(db.labels_) * (-1)

    def outlier_idx():
        for i in range(0, len(db.labels_)):
            if (db.labels_[i] == -1):
                return i

    min_ = 0.001
    max_ = 100000
    m = (min_ + max_) / 2

    while (True):
        num_out = num_outliers(m)
        if (num_out == 1):
            return outlier_idx()
```

```
        elif (num_out > 1):
            min_ = m
        else:
            max_ = m
        m = (min_ + max_) / 2

with open("serie1.csv", 'rb') as f:
    reader = csv.reader(f)
    x1 = [float(b) for [a,b] in list(reader)[1:]]

with open("serie2.csv", 'rb') as f:
    reader = csv.reader(f)
    x2 = [float(b) for [a,b] in list(reader)[1:]]

with open("serie3.csv", 'rb') as f:
    reader = csv.reader(f)
    x3 = [float(b) for [a,b] in list(reader)[1:]]

with open("serie4.csv", 'rb') as f:
    reader = csv.reader(f)
    x4 = [float(b) for [a,b] in list(reader)[1:]]

with open("serie5.csv", 'rb') as f:
    reader = csv.reader(f)
    x5 = [float(b) for [a,b] in list(reader)[1:]]

series = [x1, x2, x3, x4, x5]
splitted_series = [regular_splitter(x, 350) for x in series]
dct_components = np.asarray([[dct(split) for split in s] for s in splitted_series])
light_dct_components = np.asarray([[split[0:3] for split in comp_seq] for comp_seq in
dct_components])

print "::: This program uses DBScan for detecting outlier sub-sequences of five different sequences."
print "::: A binary-search based algorithm is used for setting DBScan parameters in such a way that
only one outlier region is detected."
print "::: Samples used by DBScan for processing the series are the first three DCT components of
each of the series' sub-sequences."
print "::: Now, each of the five analyzed sequences will be displayed. Outlier regions should be
wrapped by red vertical lines."

for i in range(0, len(series)):
    outlier_id = find_outlier(light_dct_components[i])

    plt.plot(series[i])
    plt.ylabel('value')
    plt.xlabel('time')
    plt.axvline(x=outlier_id*350, color='r', linewidth=4)
    plt.axvline(x=(outlier_id+1)*350, color='r', linewidth=4)
    plt.title("Series #%d" % (i))
    plt.show()
```