Data Mining

Homework 3

Due: 8/12/2019, 23:59

Instructions

You must hand in the homework electronically and before the due date and time.

This homework has to be done by each **person individually**.

Handing in: You must hand in the homework by the due date and time by an email to Andrea (`mastropietro@diag.uniroma1.it`) that will contain as attachment **(not links to some file-uploading server!)** a .zip file with your answers. The filename of the attachment should be
`DM_Homework_1__StudentID_StudentName_StudentLastname.zip`;
for example:
`DM_Homework_1__1235711_Robert_Anthony_De_Niro.zip`.
The email subject should be
`[Data Mining] Homework_1 StudentID StudentName StudentLastname`;
For example:
`[Data Mining] Homework_1 1235711 Robert Anthony De Niro`.
After you submit, you will receive an acknowledgement email that your project has been received and at what date and time. If you have not received an acknowledgement email within 2 days after you submit then contact Andrea.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

For information about collaboration, and about being late check the web page.

Problem 1. We are given a set of points $V \subset \mathbb{R}^d$, with $|V| = N$, and a clustering $\mathcal{C} = C_1, \ldots, C_K$ in K clusters, that is, a partition of the points into sets C_k such that $\cup_{k=1}^{K} C_k = V$ and $C_k \cap C_\ell = \emptyset$ for $k \neq \ell$. Recall that the k-means cost function for clustering $\mathcal{C}$ is the

$$\sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|_2^2 ,$$

where μ_k is the average of the points in C_k.

1. Assume instead that we use the objective function

$$\sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|_1 ,$$

   that is, we try to minimize the ℓ_1 distance between the points and the cluster center. How should we modify the k-means algorithm? Justify your answer.

2. Assume that the optimal solution for the k-means cost function has cost C. You are now asked to cluster the points, but under the constraint that the cluster representative (i.e., the point corresponding to μ_k) has to be one of the input points in V. Prove that there exists a solution with cost at most $4C$.

Problem 2.

Benchmark Data Set Consider an instance generated as follows. Let I_k be the $k \times k$ identity matrix. Let $\mathcal{N}^{k,d}(0,\sigma^2)$ be the $k \times d$ matrix in which every entry is a Gaussian random variable with mean 0 and variance σ^2. We now consider the instance A obtained by stacking n copies of the matrix I_k $\mathcal{N}(0,\sigma^2)^{k,d}$, that is,

$$A = \begin{bmatrix} I_k & \mathcal{N}^{k,d}(0,\sigma^2) \\ I_k & \mathcal{N}^{k,d}(0,\sigma^2) \\ \vdots & \\ I_k & \mathcal{N}^{k,d}(0,\sigma^2) \end{bmatrix}$$

A has $k \cdot n$ points and $k + d$ dimensions. The "correct" clustering of the rows of A is to simply put A_i into cluster $i \bmod k$, that is, the matrix I_k is also an indicator matrix for cluster membership and the correct clustering matrix satisfies

$$X_{i,j} = \begin{cases} \frac{1}{\sqrt{n}} & \text{if } i \text{ mod } k = j+1 \\ 0 & \text{otherwise} \end{cases}.$$

To set n, k, d, and σ^2, we suggest the following values (experiment with them).

- $k = 50, 100, 200$
- $n = 1.000, 10.000, 100.000$ (go to a million if your computer is too good)
- $d = k, 100k, 100k^2$
- $\sigma^2 = 1/k, 1/\sqrt{k}, 0.5$

To generate this data set, we recommend the loosely python inspired pseudocode (do **not** copy/paste, check for errors):

```
import numpy as np
import math
from random import gauss
set n,k,d and $\sigma$ as necessary
data = np.ndarray(shape=(k*n, d+k), dtype=float, order='F')
for i in range(k*n): do
  for j in range(d): do
    data[i][k+j] = gauss(0,$\sigma$)
  end for
  for j in range(k): do
    if $i\%k == l$: then
      data[i][j] = 1
    else
      data[i][j] = 0
    end if
  end for
end for
```

Algorithms We have seen the following two algorithms in class:

PCA: project the data onto the first m principal components. Recall that $m \geq k$. If we set m close to k we remove more noise. If $m \approx k/\varepsilon$, we remove less noise but have more accuracy wrt the k-means objective.

```
import numpy as np
set m as necessary
u,s,vh = np.linalg.svd(data, full_matrices=True)
smat = np.zeros((k*n, d+k), dtype=float)
smat[:s.size, :s.size] = np.diag(s)
for  i in range(0,s.size): do
  if i > m-1: then
    s[i] = 0
  end if
end for
smat[:d+k, :d+k] = np.diag(s)
projected = np.dot(u, np.dot(smat, vh))
```

***k*-means++**:

```
C={random input point}
for i=1 to k-1 do
  pick point $p$ proportionate to $\min_{c \in C} \|p - c\|^2$
  add $p$ to $C$
end for
```

Tasks Experiment with a data analysis chain. In particular, try the following two options:

1. Run k-means++ on the data set.
2. Apply PCA and then apply k-means++.

For all variants, consider whether the algorithm was able to recover the ground clustering. Try to explain why certain combinations were more successful than others.

Finally, consider the running time of the various steps. Which parts were the most expensive parts? Did this behavior change, depending on the order in which the algorithms were executed? Remember, you can log the running time using:

```
from datetime import datetime
tStart = datetime.now()
run Algorithm
tEnd = datetime.now()
```