Homework 8. K-means and Recommendation system

Double Click here to edit this cell

- Name: 전기범
- Student ID: 201703091
- Submission date: 2019/06/25

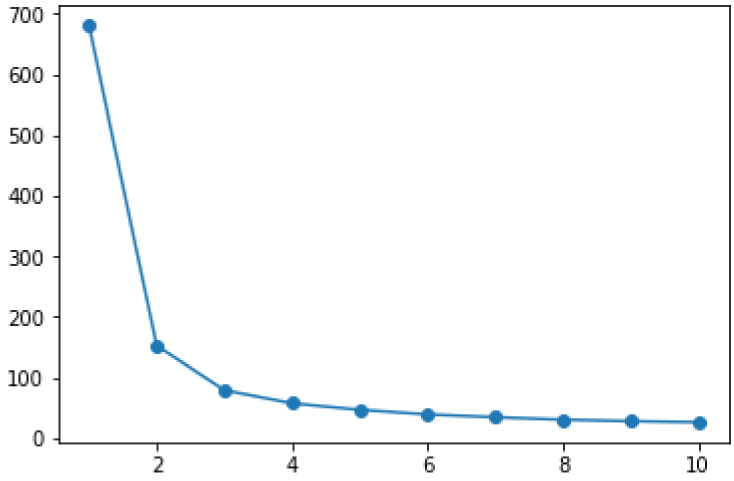
Problem 1 (10 pts): K-means

- We want to cluster data in sample_data_1.csv
- Estimate the best k for sample_data_1.csv
- You must show the process to find the best k
- use `sklearn.cluster.KMeans`

In [2]:

```
# # YOUR CODE HERE. You may use as many code cells as you want.

from sklearn.cluster import KMeans
import numpy as np
import pandas as pd
import matplotlib.pyplot  as plt
import csv
from sklearn import preprocessing

with open('sample_data_1.csv', 'r') as rf:
    reader = csv.reader(rf)
    X2 = np.array(list(reader))

iris_df=pd.DataFrame(X2,columns=['Sepal_Length','Sepal_Width',
                            'Petal_Length','Petal Width'])

plist=[]
for k in range (1, 11):
    kmeans_model = KMeans(n_clusters=k, random_state=1).fit(iris_df.iloc[:, :])
    labels = kmeans_model.labels_
    interia = kmeans_model.inertia_
    print("k:",k, " cost:", interia)
    plist.append(interia)

plt.plot(range(1,11), plist, marker='o', linestyle='solid')
plt.show()
```

```
k: 1  cost: 680.8244
k: 2  cost: 152.36870647733906
k: 3  cost: 78.94084142614602
k: 4  cost: 57.345409315718165
k: 5  cost: 46.53558205128205
k: 6  cost: 38.95701115711985
k: 7  cost: 34.32652991452992
k: 8  cost: 30.227724598930486
k: 9  cost: 27.766706937799047
k: 10  cost: 26.07225182334006
```



Your conclusion:

```
To edit, double-click here
k의 값이 안정적으로 되는 3이 최적의 k의 값이라고 볼 수 있다.
```

Problem 2 (40 pts): K-means implementation

- Make your own implementaion of K-means algorithm
- If the sum of distances between previous centroids and current centroids is less than or equal to `EPSILON` , K-means stops.
- If K-means algorithm reaches the maximum number of iterations `max_iter` , it stops.
- In `fit` method, you must run k-means in `n_init` times with different centroid seeds. Then choose the best.
- `fit` method computes centroids and labels and stores them in `self.cluster_centers_` and `self.labels_`
- `predict` method returns the centroids closest to each point in `X`
- `score` method returns **the negative of** the sum of sqaured distances between each point in `X` and the centroid closest to the point.

In [3]:

```
import numpy as np

class MyKMeans:
    """performs k-means clustering using numpy"""

    def __init__(self, n_clusters=8, n_init=10, EPSILON=1e-4, max_iter=300, random_state=0):
        self.n_clusters = n_clusters          # number of clusters
        self.n_init = n_init                  # number of time the k-means algorithm will be run with different centroid seeds.
        self.EPSILON = EPSILON                # EPSILON; stop if the sum of centroid movements <= EPSILON
        self.max_iter = max_iter              # maximum number of iterations
        self.random_state = random_state      # random number seed
        self.cluster_centers_ = None          # means of clusters
        self.labels_ = None                   # X's assignments to clusters


    def fit(self, X):
        X = X.astype(float)
        # FILL OUT

        return self


    def predict(self, X):
        X = X.astype(float)
        # FILL OUT


    def score(self, X):
        X = X.astype(float)
        # FILL OUT
```

Run the following code:

```
In [4]:
```

```
# DO NOT EDIT
import numpy as np
X = np.array([[1, 2], [1, 4], [1, 0],
              [10, 2], [10, 4], [10, 0]])

kmeans = KMeans(n_clusters=2).fit(X)
print(kmeans.cluster_centers_)
print(kmeans.labels_)
print(kmeans.score(X))
```

```
[[10.  2.]
 [ 1.  2.]]
[1 1 1 0 0 0]
-16.0
```

Your output must be the following:

```
[[10.  2.]
 [ 1.  2.]]
[1 1 1 0 0 0]
-16.0
```

Run the following code:

```
In [7]:
```

```
# DO NOT EDIT
%matplotlib inline

from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
import csv

with open('sample_data_2.csv', 'r') as rf:
    reader = csv.reader(rf)
    X2 = np.array(list(reader))

ks3 = range(1, 21)
%time errors3 = [-MyKMeans(n_clusters=k, n_init=10).fit(X2).score(X2) for k in k
s3]
```

```
---------------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<timed exec> in <module>

<timed exec> in <listcomp>(.0)

TypeError: bad operand type for unary -: 'NoneType'
```

윈도우 시스템 with open('sample_data_2.csv', 'rb') as rf:

Your code will be graded based on the correctness and the performance

My implementation result:

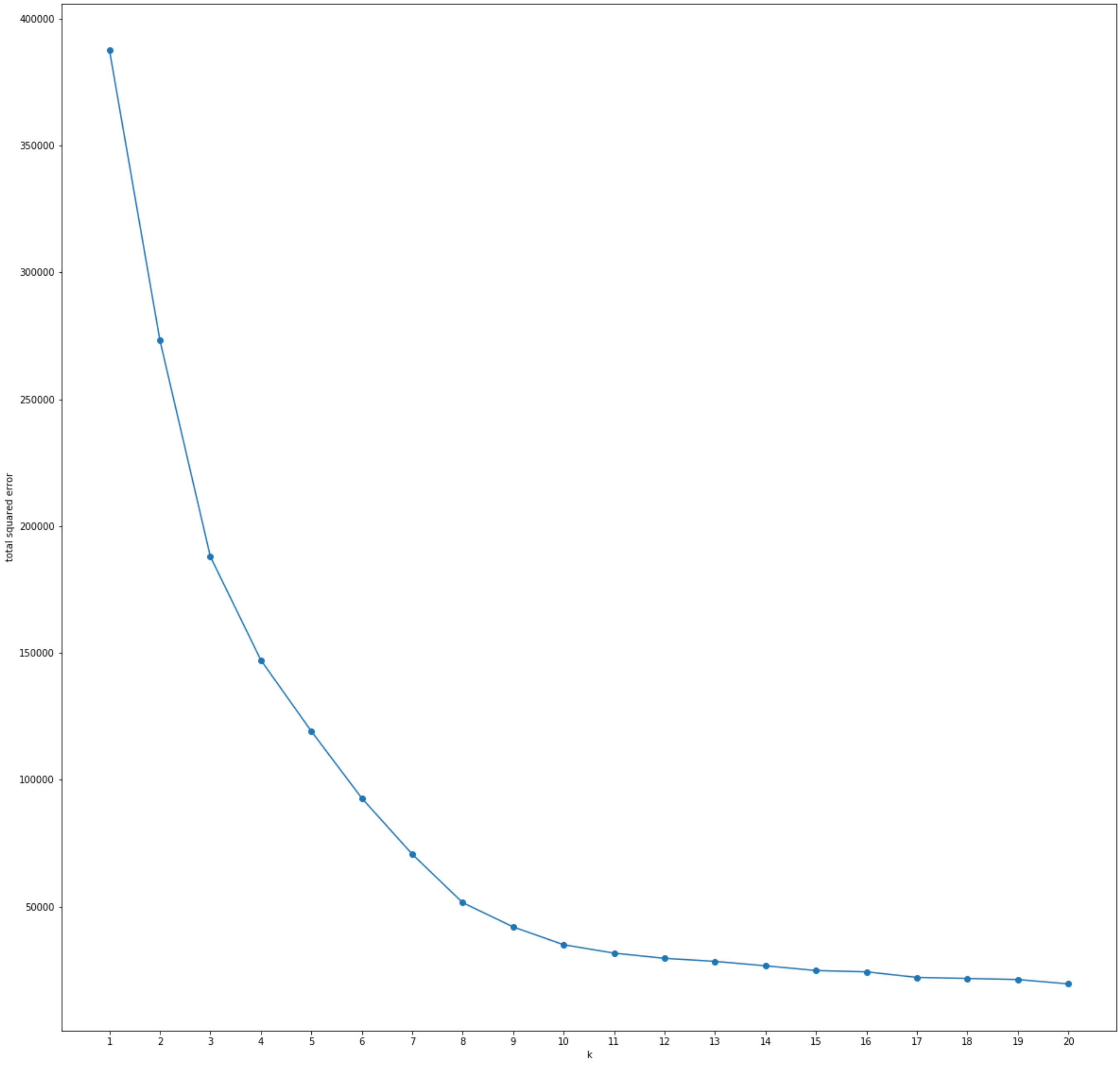
```
CPU times: user 4.18 s, sys: 3.92 ms, total: 4.18 s
Wall time: 4.2 s
```

```
In [8]:
```

```
# DO NOT EDIT
plt.figure(figsize=(20,20))
plt.plot(ks3, errors3, '-o')
plt.xticks(ks3)
plt.xlabel("k")
plt.ylabel("total squared error")
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent cal
l last)
<ipython-input-8-83797a49859c> in <module>
      1 # DO NOT EDIT
      2 plt.figure(figsize=(20,20))
----> 3 plt.plot(ks3, errors3, '-o')
      4 plt.xticks(ks3)
      5 plt.xlabel("k")

NameError: name 'errors3' is not defined

<Figure size 1440x1440 with 0 Axes>
```

Your output must be similar to the following:



Problem 3 (40 pts): Recommender implementation

- Make your own implementaion of item-based recommender system
- You may use the code in textbook, but it may be too slow for large datasets.
- You shouldn't import and use any module implementing recommender system directly
- Use cosine similarity for item similarity
- We will use movie rating dataset

In [9]:

```python
import pandas as pd

movies = pd.read_csv('movies.csv')
movies.head(5)
```

Out[9]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [10]:

```python
movies.count()
```

Out[10]:

```
movieId    9742
title      9742
genres     9742
dtype: int64
```

In [11]:

```python
ratings = pd.read_csv('ratings.csv')
ratings.head(5)
```

Out[11]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

In [12]:

```
ratings.count()
```

Out[12]:

```
userId       100836
movieId      100836
rating       100836
timestamp    100836
dtype: int64
```

The following is top 10 recommendations of movie titles and their genres from top 1 to top 10 for user 1

```
    Ferris Bueller's Day Off (1986)                  Comedy
    Die Hard (1988)                                  Action|Crime|Thriller
    Breakfast Club, The (1985)                       Comedy|Drama
    Fifth Element, The (1997)                        Action|Adventure|Comedy|
    Sci-Fi
    Aliens (1986)                                    Action|Adventure|Horror|
    Sci-Fi
    Mars Attacks! (1996)                             Action|Comedy|Sci-Fi
    Sixth Sense, The (1999)                          Drama|Horror|Mystery
    Austin Powers: The Spy Who Shagged Me (1999)     Action|Adventure|Comedy
    2001: A Space Odyssey (1968)                     Adventure|Drama|Sci-Fi
    Terminator 2: Judgment Day (1991)                Action|Sci-Fi
```

In [13]:

```
ratings.loc[ratings.userId==1].sort_values(by=['rating'], ascending=False).movie
Id[:10]
```

Out[13]:

```
231    5060
185    2872
89     1291
90     1298
190    2948
189    2947
188    2944
186    2899
184    2858
179    2700
Name: movieId, dtype: int64
```

Find top 10 recommendations of movie titles and their genres from top 1 to top 10 for user 2

In [14]:

```
# YOUR CODE HERE. You may use as many code cells as you want.
mov=list(ratings.loc[ratings.userId==2].sort_values(by=['rating'], ascending=Fal
se).movieId[:10])
# print(mov)
df = pd.DataFrame(columns = ['title','genres'])
for i in range (len(mov)):
    df.loc[i]=0

idx=0
for i in mov:
    if (movies['movieId']==i).any():
#           print(movies.loc[movies['movieId']==i,['title','genres']])
#           print(list(movies.loc[movies['movieId']==i].title)[0])
        df.loc[idx]={'title':list(movies.loc[movies['movieId']==i].title)[0],
                     'genres':list(movies[movies['movieId']==i].genres)[0]}
        idx+=1
print(df)
```

```
                                                title  \
0  The Jinx: The Life and Deaths of Robert Durst ...
1                           Mad Max: Fury Road (2015)
2                      Wolf of Wall Street, The (2013)
3                                      Warrior (2011)
4                                Step Brothers (2008)
5                                   Inside Job (2010)
6                            Good Will Hunting (1997)
7                             Dark Knight, The (2008)
8                         Inglourious Basterds (2009)
9                                    Town, The (2010)

                                 genres
0                           Documentary
1  Action|Adventure|Sci-Fi|Thriller
2                    Comedy|Crime|Drama
3                                 Drama
4                                Comedy
5                           Documentary
6                         Drama|Romance
7            Action|Crime|Drama|IMAX
8                    Action|Drama|War
9              Crime|Drama|Thriller
```

Ethics:

If you cheat, you will get negatgive of the total points. If the homework total is 22 and you cheat, you get -22.

What to submit

- Run all cells
- Goto "File -> Print Preview"
- Print the page
- Submit in class
- No late homeworks accepted
- Your homework will be graded on the basis of correctness and programming skills

Deadline: 6/25