Sardar Patel Institute of Technology,Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2021-2022)
Data Analytics

Name: Sneha Ghuge **UID:** 2019110015 **BE ETRX** **DA LAB 9**

Aim: The focus of this lab is k-means clustering. We will look at the vanilla algorithm, its performance, and some better variants. Finally, we will use clustering for classifying the MNIST data set.

PROBLEM STATEMENT :

Task 1: Implementing k-means clustering (3 marks)

Implement all of the following 6 functions in kmeans.py.

1. distance_euclidean(p1, p2)
2. distance_manhattan(p1, p2)
3. iteration_one(data, means, distance)
4. has converged(old_means, new_means, epsilon)
5. iteration_many(data, means, distance, numiter, epsilon)
6. performance_SSE(data, means, distance) Test your code by running this command.

python kmeans.py --input datasets/flower.csv --iter 100 --epsilon 1e-3 --init forgy --dist euclidean --k 8

--seed $RANDOM

Try different values of , and try both Euclidean and Manhattan distances.

Evaluation: Each correctly implemented function will fetch you mark. Test with python autograder.py 1.

Task 2: Testing and Performance (2 marks) Test your code on the following data sets. datasets/100.csv: Use , numexperiments datasets/1000.csv : Use , numexperiments datasets/10000.csv : Use , numexperiments

Use epsilon and the Euclidean distance metric for every experiment. Here is an example.

python kmeans.py --epsilon 1e-2 --init forgy --dist euclidean --input datasets/100.csv --k 2

-numexperiments 100

Answer the following 2 questions in a file named solutions.txt.

1. Run your code on datasets/garden.csv, with different values of . Look at the performance plots and answer whether the SSE of the k-means clustering algorithm ever increases as the iterations are performed. [1 mark]

2. Look at the files 3lines.png and mouse.png. Manually draw cluster boundaries around the 3 clusters visible in each file (no need to submit the hand drawn clustering). Test the k-means algorithm with different random seeds on the data sets datasets/3lines.csv and datasets/mouse.csv. How does the algorithm's clustering compare with the clustering you did by hand? Why do you think this happens? [1 mark]

Evaluation: The text questions carry marks as specified. Make sure to write clean, succinct answers.

It is worth noting that k-means can sometimes perform poorly! Test your algorithm on the datasets/rectangle.csv data set several times, using . Depending on the

initialisation, k-means can converge to poor clusterings.

Task 3: Implementing k-means++ (3 marks)

Implement the following function in kmeans.py.

1. initialization_kmeansplusplus(data, distance, k)

Note: You are expected to provide elaborate comments along with your code (for the function). Your marks depend on whether the TAs are able to understand your code and establish its correctness.

Test with python autograder.py 3.

Use your code by running the following command (this is for you to check that the code is working, not used for evaluation).

python kmeans.py --input datasets/flower.csv --epsilon 1e-2 --init kmeans++ --dist euclidean --k 8

After implementing your code, test it on these data sets. datasets/100.csv: Use , numexperiments

datasets/1000.csv : Use , numexperiments datasets/10000.csv : Use , numexperiments

Use epsilon and the Euclidean distance metric for every experiment. Answer the following question in the file solutions.txt.

1. For each data set and initialisation algorithm (Forgy and k-means++), report "average SSE" and "average iterations". Explain the results.

Evaluation: Correct implementation of the kmeans++ function will fetch you mark. The text question is worth marks.

Notice how:

- kmeans++ initialisation leads to considerably less cluster movement compared to Forgy initialisation;
- Despite using kmeans++, the algorithm will sometimes converge to poor solutions.

Task 4: MNIST classification (2 marks)

Template File: kmeans.py

Data set: datasets/mnist.csv

Run your algorithm on the MNIST data set as follows.

python kmeans.py --input datasets/mnist.csv --iter 100 --epsilon 1e-2 --init kmeans++ --dist euclidean --k 10 --output mnist.txt

Plot the so found cluster centres by executing this command. python mnistplot.py mnist.txt

Look at the plots and find out a good mean for each of the 10 digits. Compile these means into the file mnistmeans.txt. You will have to run the clustering algorithm several times to get satisfactory means. Use the random seed to get different means. Naturally, you should not use the labels of the points in any way while clustering them: that is, your clustering algorithm should run on the entire (unlabeled) data set.

File Format for mnistmeans.txt:

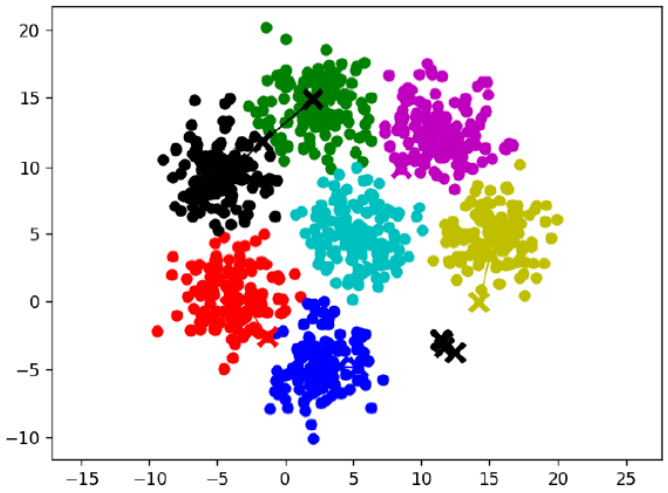
The -th line must contain the cluster mean for the -st digit. Each mean is represented by a comma separated list of floats.

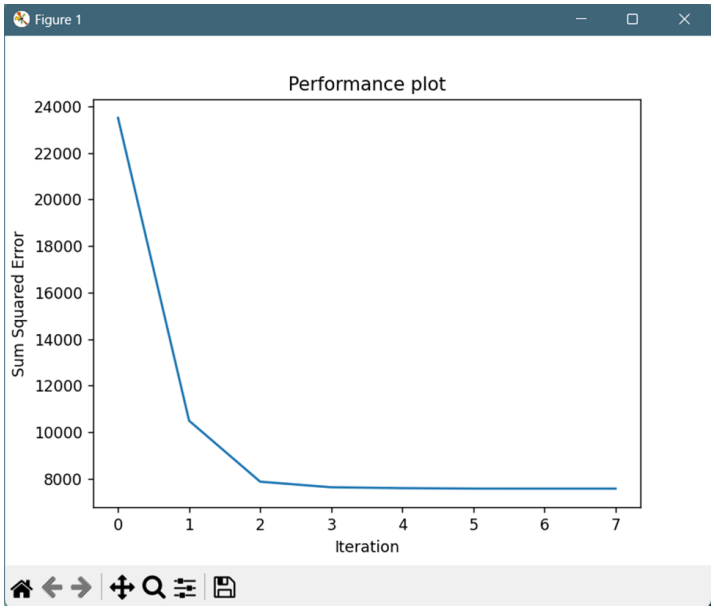
Evaluation: Your cluster means will be used to classify the MNIST data set Test with python autograder.py 4.

In practice, you would want to choose multiple cluster means per class instead of just one, for increased accuracy.

CODE & OUTPUT:

TASK 1 -






```
Arguments:
{'dist': <function distance_euclidean at 0x00000265C5088910>,
 'epsilon': 0.001,
 'init': <function initialization_forgy at 0x00000265C508BA30>,
 'input': 'datasets/flower.csv',
 'k': 8,
 'maxiter': 100,
 'numexperiments': 1,
 'output': None,
 'seed': 0,
 'verbose': False}
----------------------------------------

Number of points in input data: 1000

Experiment: 1
Seed: 0
Sum Squared Error: 7573.88060479682
Number of iterations till termination: 7
Convergence achieved: True


Average SSE: 7573.88060479682
Average number of iterations: 7.0
Visualizing...
```

TASK 2 -

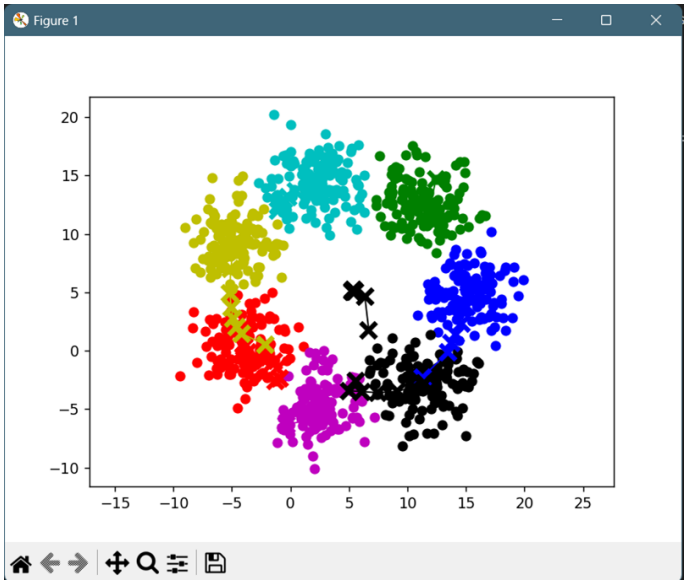
```
Arguments:
{'dist': <function distance_euclidean at 0x0000027E269FB910>,
 'epsilon': 0.01,
 'init': <function initialization_forgy at 0x0000027E269FBA30>,
 'input': 'datasets/100.csv',
 'k': 2,
 'maxiter': 10000,
 'numexperiments': 100,
 'output': None,
 'seed': 0,
 'verbose': False}
----------------------------------------

Number of points in input data: 100

Experiment: 1
Seed: 0
Sum Squared Error: 8472.633114692015
Number of iterations till termination: 2
Convergence achieved: True
Experiment: 2
Seed: 1
Sum Squared Error: 8472.633114692015
Number of iterations till termination: 2
Convergence achieved: True
Experiment: 3
Seed: 2
Sum Squared Error: 8472.633114692015
Number of iterations till termination: 3
Convergence achieved: True
```

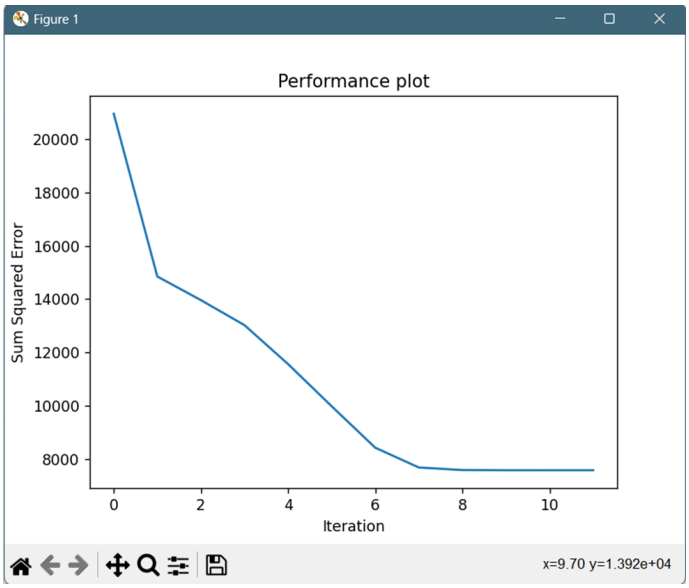
```
TERMINAL    OUTPUT    DEBUG CONSOLE

Seed: 98
Sum Squared Error: 739453085.7049414
Number of iterations till termination: 3
Convergence achieved: True
Experiment: 100
Seed: 99
Sum Squared Error: 742748774.7269629
Number of iterations till termination: 3
Convergence achieved: True


Average SSE: 735138649.9610686
Average number of iterations: 3.32
```

TASK 3 -

```
==================== TASK 3 ====================
  b'\xe2\x9c\x93'  Function initialization_kmeansplusplus
NOTE: The autograder doesn't check for correct implementation of this function.
Your marks depend on whether the TAs are able to understand your code and establish its correctness.
------------------------------
  b'\xe2\x9c\x93'  Function initialization_randompartition
NOTE: The autograder doesn't check for correct implementation of this function.
Your marks depend on whether the TAs are able to understand your code and establish its correctness.
------------------------------

NOTE: This task has an additional manually graded question worth 1 mark. Answer it in solutions.txt

grade: 3
```

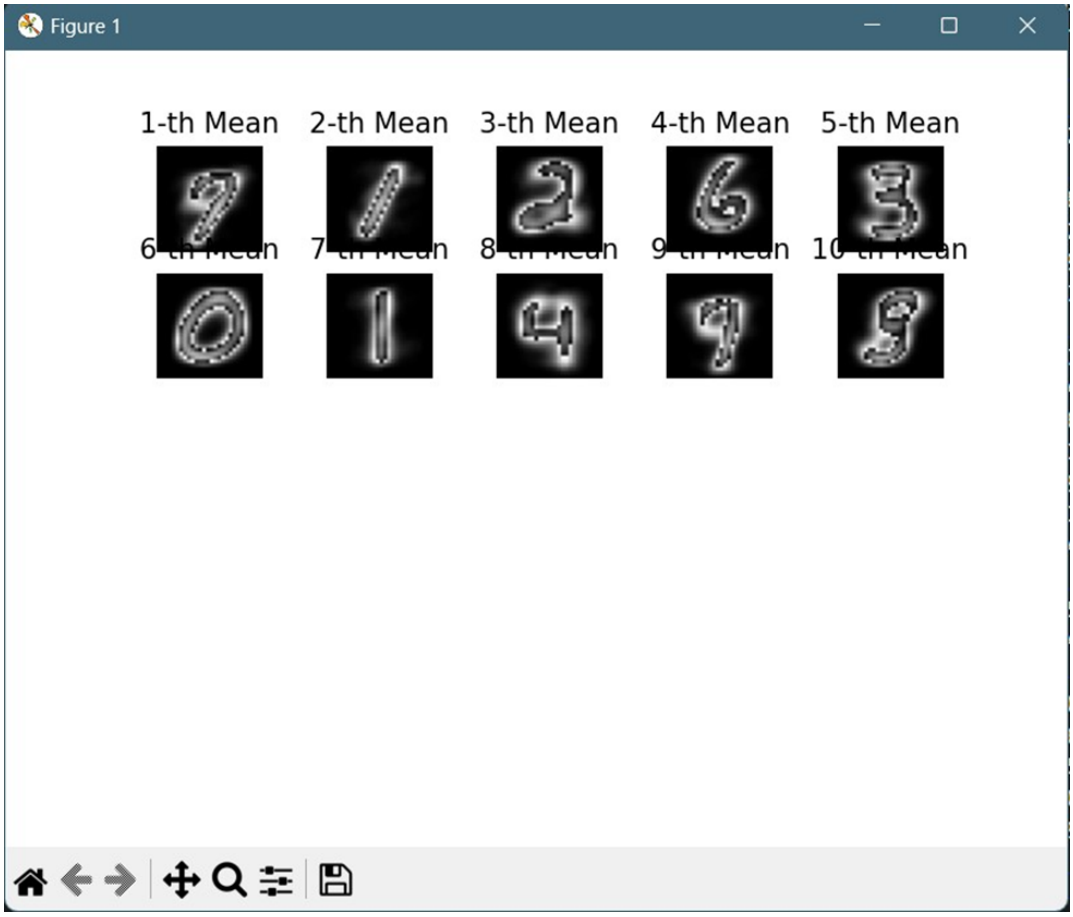
```
Arguments:
{'dist': <function distance_euclidean at 0x0000020EC510F9A0>,
 'epsilon': 0.01,
 'init': <function initialization_kmeansplusplus at 0x0000020EC510FBE0>,
 'input': 'datasets/flower.csv',
 'k': 8,
 'maxiter': 10000,
 'numexperiments': 1,
 'output': None,
 'seed': 0,
 'verbose': False}
--------------------------------------

Number of points in input data: 1000

Experiment: 1
Seed: 0
Sum Squared Error: 7573.88060479682
Number of iterations till termination: 11
Convergence achieved: True


Average SSE: 7573.88060479682
Average number of iterations: 11.0
Visualizing...
```





TASK 4 -

```
Arguments:
{'dist': <function distance_euclidean at 0x0000022F7FB6F9A0>,
 'epsilon': 0.01,
 'init': <function initialization_kmeansplusplus at 0x0000022F7FB6FBE0>,
 'input': 'datasets/mnist.csv',
 'k': 10,
 'maxiter': 100,
 'numexperiments': 1,
 'output': 'mnist.txt',
 'seed': 0,
 'verbose': False}
----------------------------------------

Number of points in input data: 5000

Experiment: 1
Seed: 0
```




```
==================== TASK 4 ====================
Accuracy acieved: 51.9%
grade: 2
```