IMEN281 Information System Technology
Pohang University of Science and Technology
Fall 2020

Homework 1: k-means clustering

Due Date: Oct.23 (Friday) 11:59 p.m.

Problem Description

k-means clustering is a method of vector quantization, partitioning n observations into k clusters. k-means clustering is the simplest but powerful clustering algorithm and popular for cluster analysis in data mining. Intuitively, k-means clustering minimizes within-cluster variances (squared Euclidean distances) with the given number of clusters.

1. **Mathematical Description**. Given a set of data points $(x_1, x_2, \cdots, x_n)$ where each data point is a d-dimensional real vector. k-means clustering aim to partition the n data points into $k(\leq n)$ sets $S = \{s_1, s_2, \cdots, s_k\}$ to minimize the within-cluster sum of squares, which can be written as below,

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in s_i} \|x - \mu_i\|^2 = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} |s_i| Var s_i, \tag{1}$$

where μ_i is the means of points (centroid) in s_i.

2. **Heuristic Algorithms**. By definition, the problem itself is computationally difficult (NP-hard); however, an efficient heuristic algorithm converges quickly to a local optimum. The most common algorithm is Lloyd's algorithm, often called "the k-means algorithm".

Initialization step: First, the initial set of k means need to be initialized. In general, the Forgy method, which is randomly chosen k observation from the dataset and uses theses as an initial $m_i^{(1)}, \cdots m_k^{(1)}$, is commonly used.

Given an initial set of k means $m_i^{(1)}, \cdots m_k^{(1)}$, the algorithm proceeds iterative refinement by alternating between two procedures: assignment step and update step.

Assignment step: Assign each data points to the cluster with the nearest mean. Here, simply use least squared Euclidean distance (L2 norm).

$$s_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 <= \|x_p - m_j^{(t)}\|^2, \forall j, 1 \leq j \leq k\}, \tag{2}$$

where each x_p is assigned to only one $S^{(t)}$. If it could be assigned to more than two clusters, just randomly choose among them.

Update step: Recalculate means of data points assigned to each cluster (centroid).

$$m_i^{(t+1)} = \frac{1}{|s_i^{(t)}|} \sum_{x_j \in s_i^{(t)}} x_j, \tag{3}$$

where each x_p is assigned to only one $S^{(t)}$. If it could be assigned to more than two clusters, just randomly choose among them.

Ideally, the algorithm has converged when the assignments no longer change but relative tolerance λ usually is used as a stop criteria. Algorithm stop when

$$\lambda \geq \sum_{i=1}^{k} \frac{\|m_i^{(t)} - m_i^{(t-1)}\|}{\|m_i^{(t-1)}\|}. \tag{4}$$

Problem

1. Implement k-means class (in kmeans.py). Implementation should be objective oriented. Default value for relative tolerance λ is 0.001.

2. Read "data.txt" file, test k-means algorithms with data, write result cluster on "result.txt" and also report centroid of each class, elapsed time in second (in kmeans.py). The number of the cluster, k is 5 and the execution screen should follow the below example.

```
➜  Downloads python kmeans.py
Detected centroid:
Centroid 0: 4.9974, 2.5092
Centroid 1: -1.9962, -0.7454
Centroid 2: -2.9913, 2.4861
Centroid 3: 0.9673, 0.9935
Centroid 4: 4.9951, -2.0319

Total elapsed time: 0.018 sec
```

Figure 1: Running Example

3. Implement "k-means++". Difference between k-means and k-means++ is initialization step. Initialization step of k-means++ is as follows (in kmeans++.py).

   (a) Choose one center uniformly at random among the data points.

   (b) For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.

   (c) Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$. (you can use python's random. choices)

   (d) Repeat Steps a and b until k centers have been chosen.

   Now that the initial centers have been chosen, proceed using standard k-means clustering. You can easily implement k-means++ with inherits k-means class and overload initialize function.

Input/Output Data Format

1. **Input data (data.txt)**: Each row of input data correspond to one data point, containing index, x, y (e.g. 0, 3.2, 4.5).

2. **Output data (result.txt)**: Each row of input data correspond to one data point, containing index, cluster_no. (e.g. 0, 2).

Submission

1. **Implemented code with comment**: kmeans.py, kmeans++.py.
2. **Report**: Report format is given as below.
   (a) Objective
   (b) Method and Algorithm
   (c) Discussion
   (d) If you receive help from someone else on the homework, you must specify who, which part (debugging, logic, coding) and how much (time) in the report.
   (e) Time spent on assignment

* Do not use external libraries such as numpy, pandas, scikit-learn, etc.

Due Date and Late Submission

* Due Date is Oct.23 (Friday) 11:59 p.m.

* 20% deduction for one day late and not received after 3 days (zero point) - No exception applies unless there is a special reason.

Hint: Diagram for design class

	kmeans
Data	points: array number_of_cluster: int assigned_cluster: array current_centroid: array previous_centroid: array tolerance: float
Methods	__init__() initialize_centroid() assign_points_to_cluster() update_centroid() calculate_tolerance()

Figure 2: Schematic Diagram for kmeans class