CMPSC497 Fall 2018 Programming Assignment 3.

Assigned: Friday, November 16, 2018

Due: Friday, December 7, 2018 (by 5pm, submit a package of codes in .py and a report .pdf via Canvas)

Maximum: 100 point

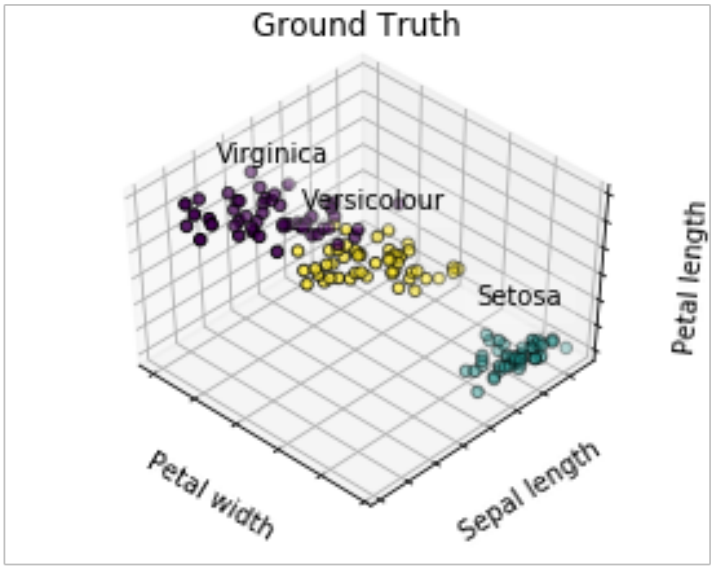
Note: This assignment is to be done by an individual student, no team work allowed.

In this assignment, you will implement k-Means cluster algorithms for use on the Iris Flower dataset, and compare your result with sklearn k-Means function.

Iris Flower dataset consists of 3 different types of irises: Setosa, Versicolour, and Virginica. The dataset includes 150 data samples and 4 attributes (Sepal Length, Sepal Width, Petal Length, and Petal Width). Here we chose three of features to demonstrate.

Dataset

- The Iris Flower dataset consists of 3 different types of irises: Setosa, Versicolour, and Virginica. The dataset includes 150 data samples and 4 attributes (Sepal Length, Sepal Width, Petal Length, and Petal Width). With the 3D visualization tool available in sklearn, the dataset is shown below by choosing three features.



k-Means Algorithm

Before you get started, let's revisit the k-Means clustering algorithm, which is used for explorative analysis of unknown (unlabeled) data. The goal is to find k clusters in the data, where the number of clusters is specified by k. The algorithm works iteratively to assign each data point to one of k clusters based on the features/attributes of the data points. The results of the k-Means clustering algorithm are (1) the *centroids* of the k clusters, which can be used to denote (label) their corresponding clusters; (2) label for each data point which is assigned to one of the clusters.

Implement your own k-Means

In this assignment, you will follow the instructions below step by step to implement your own k-Means in Python! A python template *kmeans.py* is provided, with codes for key functions, i.e., *rand_center(data, k), update_centroids(data, centroids, k=3)* and *converged(centroids1, centroids2)* removed. Your task is to implement these functions.

Step 1. Let C = {c_1, c_2, c_3} denote the set of centroids. Randomly pick 3 data points from the 150 iris data points as the initial cluster centroids.

Step 2. Assign each data point *x* to its nearest cluster, based on its Euclidean distances to the centroids.

$$\arg\min_{c_i \in C} dist(x, ci)$$

where dist(·) denotes the Euclidean distance.

Step 3. Update each cluster centroid to the mean of all the points assigned to the corresponding cluster.

$$c_i = \frac{1}{|Si|} \sum_{x \in Si} x$$

where *Si* stands for the set of points assigned to *i*-th cluster.

Step 4. Repeat Step 2 and Step 3 until the centroids converges. (Note: add proper code to handle infinite loop if it never converges.)

Evaluation Metrics:

As discussed in classes, *SSE* is used to decide the best clusterings. Additionally, if some ground truth (i.e., flower types in this assignment) is available, metrics such as *Impurity (Gini Index)* may be used for verification/evaluation as well. As part of the assignment, you need to implement *SSE(centroids, data)* and *gini(predict, ground_truth)*.

Answer the following questions:

1. Run your own k-Means multiple times (at least 10 times) to find the best three clustering based on SSE (sum of squared errors) and compare it with the result obtained from k-Means in scikit-learn for comparison. Show both clustering results using matplotlib.
2. Use the results obtained in (1) to explain why it's important to choose proper initial centroids.
3. Between the best clusterings you obtained via scikit-learn and your own k-Means implementation, which clustering is better? Please make comparison in terms of Impurity (Gini Index) of the clusterings. To compute. Gini Index for a clustering, use the ground truth (i.e., flower types of data points) to do majority vote in order to assign a flower type for each cluster. Accordingly, calculate the probability of misclassification and correct classification. Finally, calculate Gini Index for a clustering using the calculated probabilities.
4. In lab3.py, the data points are normalized (see line 12) by default as a data preprocessing step. What happens if you use the raw data (line 11) without any data preprocessing? Between normalized and unnormalized datasets, which one obtains better clustering? Please make comparison in terms of Impurity (Gini Index) of the clusterings and computational cost (number of iterations to converge).
5. By comparing the clusterings in (3) and (4) against the ground truth, explain whether Impurity is a reasonable quality measure for clustering?

Submission

- Please submit a report in pdf and a code file in .py.

Packages

- sklearn (http://scikit-learn.org/). A machine learning framework in Python

Reference

[1] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014