Programming for Data Science
DS2002 course
09.12.2025

Lab 3 – K-Nearest Neighbors and KMeans Algorithms

Summary:
K-Nearest Neighbors (KNN) is one of the simplest and most intuitive classification algorithms in machine learning that relies on the principle that close points behave similarly. It is one of the case-based learning algorithms that can learn non-linear complicated decision boundaries with a single hyperparameter, i.e., K, the number of nearest neighbors. The problem with this algorithm is that, to find the k nearest neighbors of a specific point, you have to compute the distances to all the points in the training dataset, which is very costly in terms of computation, especially with a large amount of data.

K-means is one of the popular and simplest clustering algorithms. It is a centroidbased algorithm, where we calculate the distance between each data point and a centroid to assign it to a cluster.

You are expected to implement the KNN algorithm in two ways. One uses only lists, and the second one uses vercotrized computation using Numpy. Moreover, you are expected to implement K-Means using only vercotrized computation using numpy.

Use the data provided with the lab “Iris dataset” for your experiments. It is a small dataset, so the computations would be fast. But you have the freedom to use a larger dataset of your choice to show the difference in computation between the implementations.

Note: The implementation should follow the Object Oriented Programming concepts.

Task 1: KNN Iterative Way

Goal: Implement the KNN algorithm in an iterative way. In this task, load the data into a list of lists, where each point is represented as a list of numbers. The data is split into training and testing sets. For each test point, you loop through the training points and compute the distances between the test point and all the training points, and the classification is based on the majority voting of the k nearest neighbors (You need to choose this one, 3, or 5, for example). You need to do this for all the test points and compute the accuracy; that is, how do your predictions match the actual class labels of the test points?

Task 2: KNN Vectorized Computation

Goal: Repeat the previous task using Numpy, making use of the built-in math functions and the Broadcasting feature of Numpy. This can be done by looping only on the test set. Compare the execution time between the iterative and vectorized implementations.

Task 3: K-Means Vectorized Computation

Goal: In this task you should implement the k-means algorithm in an vectorized way. Similar to what you did in Task 2. Preferably you can split the algorithm main logic into two steps "calculate centers" and "assign clusters". The algorithm output should return a cluster assignment for each data point. As we discussed in the lecture the algorithm stops when it has converged (or stabilized). You can use the "Iris dataset" for your experiments. But you have the freedom to use a larger dataset of your choice.

Note: You should merge training and testing data and exclude the class feature before using the data in K-means algorithm.

Hint: In each iteration you can compare the "old centroids" to the new ones to check the algorithm has converged.

Task 4: KNN and K-Means Using Scikit-Learn

Goal: In this task you should use Scikit-Learn library to run the KNN and Kmeans algorithms. Use the same data in the previous tasks and compare the results with the results you got with your own implementation of the algorithms.