Song Yang (sy540)
Xin Yang (xy213)
Yi Wu (yw641)
Zhuohang Li (zl299)

Project Pre-report of Data Structure and Algorithms

Clustering Algorithms

Introduction

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups or clusters. Clustering is widely used in data mining and machine learning. There are a lot of algorithms for clustering divided into five categories. In our project, we will mainly discuss two featured clustering algorithms which are the partition-based k-mean algorithm and the density-based DBSCAN algorithm.

Science and Engineering Problem Motivation

As we declared in the first paragraph, K-means and DBSCAN are two typical algorithms in clustering problem.
K-mean belongs to the partition-based method and is of great significance in machine learning, especially the unsupervised learning field. Both science and industries implement this algorithm in cases such as clients segmentation and recommendation systems. Since the use cases are closely related to massive data processing, choosing the value of k can crucially vary the performance. Besides, when it comes to non-convex problems, k-means algorithms can be incapable. Thus optimize the performance and make the scope of application more widely, it's necessary to go through this algorithm and find out proper methods.
Different from k-mean, DBSCAN is a density-based method. In practical use, density-based is suitable for decreasing the noise of sequences. It is common, and have many optimized version.

Description and Explanation of the Algorithm

K-means clustering is a typical unsupervised learning algorithm that aiming at classifying unlabeled data into k clusters. The algorithm works in an iterative way described as follows:

1. Choose k arbitrary points as centroids.
2. Assign every data points to their nearest centroids.
3. Recalculate the centroids.

4. Reassign every data point to its nearest centroid.
5. Iterate from step 2 until no point is reassigned in step 4.

The DBSCAN algorithm can be described as following steps:

1. Find the point such that with a given radius, the density of the point is larger than the given number.
2. Find the connected components of core points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is a ε (eps) neighbor, otherwise, assign it to noise.
4. Delete the noise points.

A naive implementation of this requires storing the neighborhoods in step 1, thus requiring substantial memory. The original DBSCAN algorithm does not require this by performing these steps at one point each time.

Experimental Configurations and Details.

We will design experimental programs to analyze and verify the efficiency of the algorithm. As k-means is useful in solving clustering problems, we are going to build a classification case. There are variations of implementations that can have impacts on the performance. Since the Iris dataset provided by UCI is well-organized and possesses all following features we need, including sepal length, sepal width, petal length and petal width, we choose to run our project on this set of data. Iris Dataset: http://archive.ics.uci.edu/ml/datasets/Iris

Iris Data Set

Download: Data Folder, Data Set Description

Abstract: Famous database; from Fisher, 1936

Data Set Characteristics:	Multivariate	**Number of Instances:**	150	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	4	**Date Donated**	1988-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	1773709

As mathematical vector operations are required for this algorithm, we might choose python as our programming language. Thanks to the third-party libraries like Numpy, it will be easier to perform matrix computations. Moreover, Python is widely used in machine learning and pattern recognition.

Here we will mostly put our efforts on scalar data since the data type varies in the actual environment. If time permits, we might test vector cases as well(using cosine measurement as distance). For scalars, we can regard Euclidean distance, Manhattan distance, and Pearson Correlation Coefficient as dissimilarity degree.

Reference: http://blog.csdn.net/zhoubl668/article/details/7881313

CODE:

https://www.ibm.com/developerworks/cn/analytics/library/ba-1607-clustering-algorithm/index.html