Charles Catt

Application of k-Means Clustering on Big Data

Introduction

This paper will propose and describe an approach to clustering big data using a specific algorithm: k-means.

Clustering is a problem in data science where we want an algorithm to find groups (or *clusters*) in data sets. Some clusters in data sets are easy to see by a human when the information is visualised clearly. Where clusters in two or three dimensions are easy to see by a human, it becomes impossible to identify beyond three dimensions or for massive data sets.

The k-means algorithm is a popular approach to clustering because it is an easy-to-understand and is commonly taught in undergraduate data mining courses. *Big data* is a term used to describe large data-sets, so large that they cannot fit in the program memory on a single computer. Analysing big data requires lots of resources, and often makes use of a distributed system (e.g. cloud computing). We will look in to ways that the k-means clustering algorithm can be used as a distributed process.

K-Means Algorithm

The k-means algorithm can be described as:

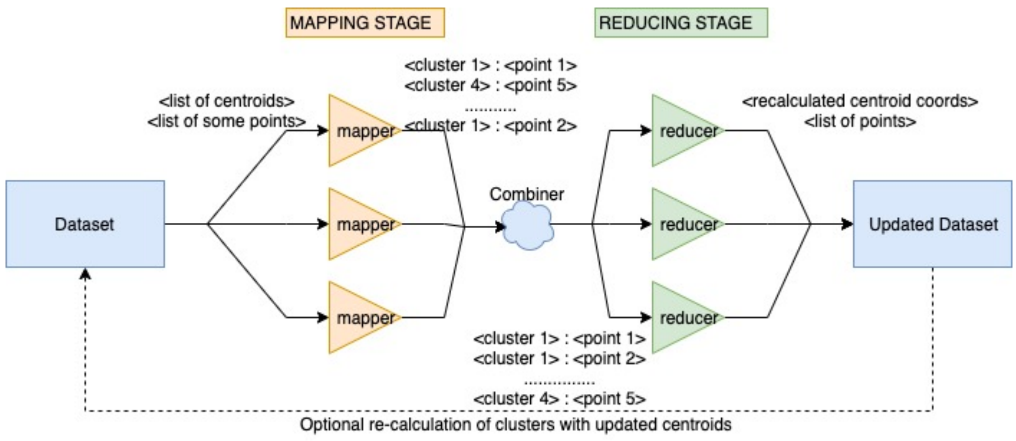
- Let *k* be an estimate of the number of clusters in the data set
- Initialise a *centroid* to a random item from the dataset.
- Until there are k centroids initialise new centroids to a point in the dataset that is furthest from the other centroids.
- Until there is little discernible change in the clusters since the previous iteration:
    - For every point in the dataset
        - find the closest centroid to that point, and classify it as belonging to the cluster of that centroid.
    - Re-calculate the position of each centroid as *the mean of all points in the associated cluster*

A strength of this algorithm is that it is very fast compared to some other algorithms (the algorithmic complexity is O(n) for n points in the data set). It is also accurate for many kinds of data sets. A robust algorithm with a simple implementation is very convenient for many applications.

A limitation of k-means is that the number of clusters *k* needs to be known beforehand. Also, this algorithm is not suitable for a distributed environment. The whole data set must first be loaded in to the memory of the process and then calculations are run with it. When a massive data set is loaded in to memory, it clogs up a lot of the system's resources and often slows down the process. Also, sometimes the data set is too big to fit in memory. It may be able to be adapted to a *map-reduce* structure, allowing it to be run on a distributed system.

Application of k-Means on Big Data

Map-reduce is a common method of mining big data for information. First, the



dataset is split up between a number of processes on *nodes*. Then, each node performs a *map* operation, which pairs a value with a key (into to a key-value pair). Next, the key-value pairs are combined so that each value of a particular key, *a*, is grouped together with all other values of key *a*. Each key (with associated values) is passed to node that performs a *reduce* operation. This map-reduce process may be chained (reducer provides the input to another pass of map-reduce) or the output of the reducer can be stored as the product of the completed computation.

We will adapt the k-means algorithm to conform to the map-reduce structure:

Running k-means on a distributed system means that:

- The complete dataset is not available each process
- Processes do not know what other processes are doing

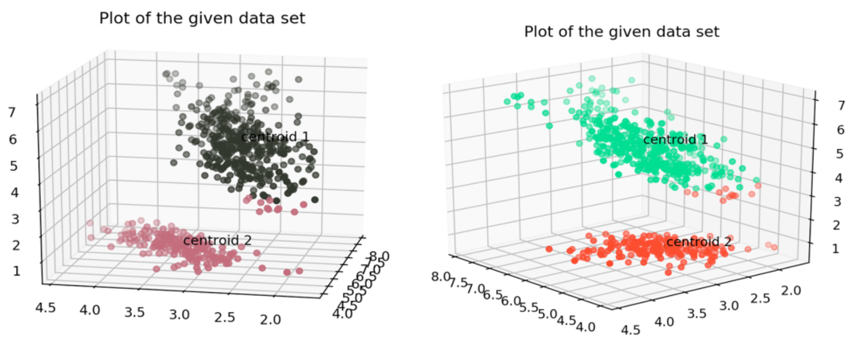
With this information, I have designed an algorithm that finds clusters in large datasets. This may already have been implemented elsewhere but this is one solution. The arrows on the following diagram show the path taken by the information from the dataset. The labels are example formats of the expected input/output of each component.

You may notice that in the description of the k-means algorithm on the previous page, the orange and green colours match the map and reduce stages respectively, because that is what they are doing.

The initial list of centroids before the mapping stage is calculated using a random sample of the dataset. As a sample, the k-means algorithm will run quite well and the optimal value for k can be approximated. Then, the data is distributed and the map-reduce process is started. Upon completion, the result should be in the same format as the initial dataset and can be used to perform another clustering iteration on the dataset.

I have adapted some code from an old university project and simulated a distributed system with the following results:

Operating on a dataset of 600 points (very small only for the demonstration). k=2 was found using 10% of the dataset. Multiple passes were not necessary, for such a small data set, and the 600 points were found quickly.



One pass of k-means clustering (left), and two passes of k-means clustering (right). Note there

Conclusion

The algorithm shows promise for big data applications, but remains largely untested. Further testing on a real distributed system with a much larger data set would be necessary to properly test the proposed algorithm. All code can be found on GitHub at https://github.com/CharlesCatt in the "Distributed-k-means-clustering" repository.