K-Means Clustering Parallelization

Team Members: Trystan Bates-Maricle, Lance Carter, Kevin Crabbe, Zach Miller

Abstract

The algorithm our group will be taking on is the K-Means Clustering algorithm. Our task will be to parallelize the k-means algorithm by distributing each group to individual processors, speeding up the overall performance of the raw algorithm.

Problem Description

Clustering algorithms are widely used throughout the industry as a mechanism for investigating how data is structured by using methods such as Euclidean Distance to identify the diversity of groups / sub groups within the data. This approach is considered “unsupervised”, as the algorithm itself is allowed to explore the data and make its own mistakes (and corrections) to the output.

At its core, the K-means algorithm can be thought of as applying the expectation maximization algorithm to a naive bayes net. In simpler terms, we apply the expectation maximization across several nodes to find the one with the highest conditional probability of fitting to that centroid (or, therefore the one with the highest expected return).

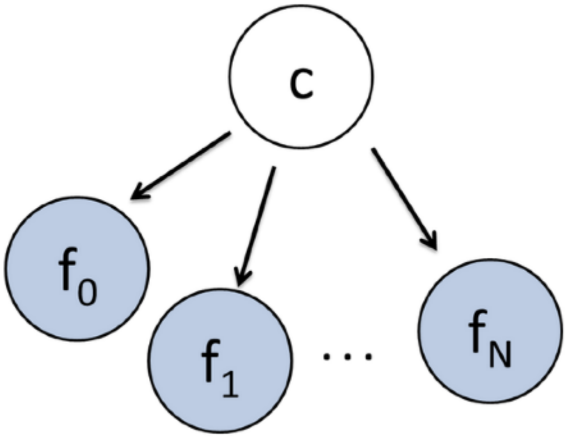
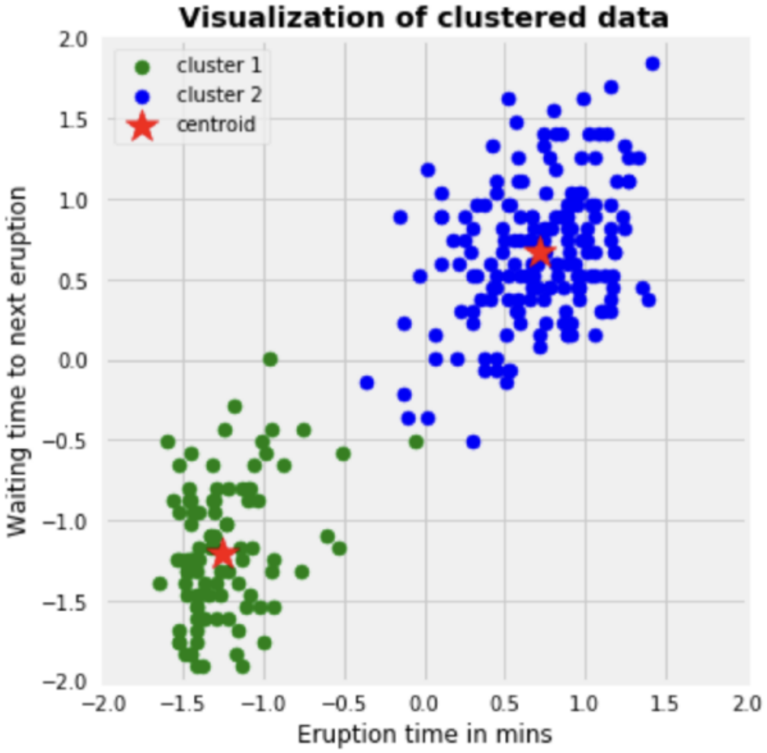


Figure 2: The K-Means algorithm is the EM algorithm applied to this Bayes Net.

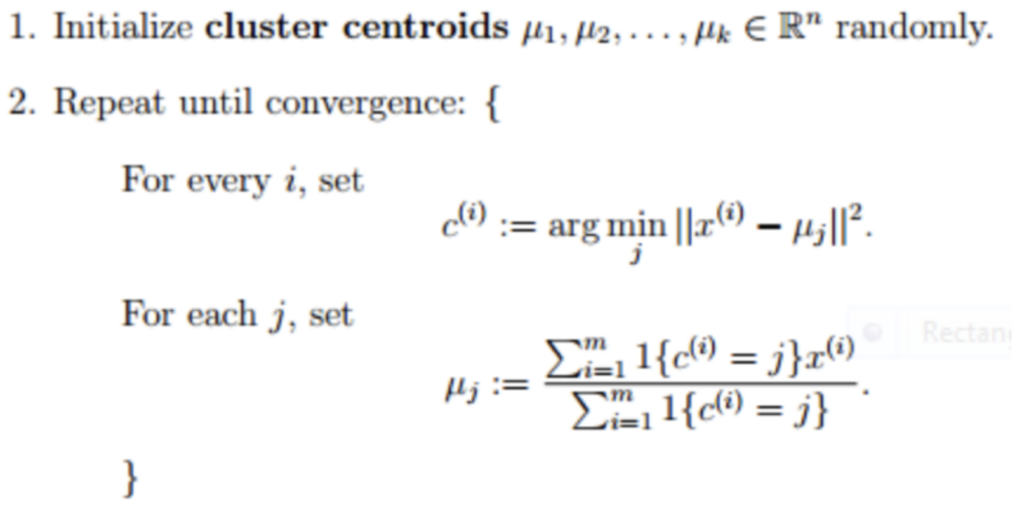
K-Means can be applied to several different scenarios such as market analysis (grouping users together), image processing, and more. Our group will be focusing on tabular numerical data, though it is important to remember this can be applied in other places.



Intellectual Challenge

The K-means algorithm is a good introduction to machine learning for those who are new to the field, while also providing a nice interface for parallelization, with a variety of options to approach from. Our group will be focused on evenly distributing k-centroids to n-processors, allowing for optimization of the computationally heavy portions of the algorithm.

Mathematically, the K-Means algorithm attempts to group the data into k-centroid (clusters) by accepting the data, without labels, and attempting to find which cluster that data point fits best to. Knowledge-wise, this will require a combination of discrete mathematics (MAT 260) and linear algebra (MAT 175) to successfully optimize the algorithm.



```
1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {
        For every $i$, set
            $c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$
        For each $j$, set
            $\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$
    }
```

Resources

Our group will test and compare the efficiency of the algorithm on the School of IT Linux Cluster, the SDSC Supercomputer cluster in San Diego, keeping the resource cost relatively low. One possibility for comparing test times would be testing our implementation's performance against the Sklearn k-means library in Python.

Work Plan Schedule

Our group will have a weekly standup via discord starting the week of April 4th, with 1.5 week sprints. Before this time the group will familiarize themselves with the algorithm and begin planning a route for parallelization.

References

Dabbura, I. (2020, August 10). *K-means clustering: Algorithm, applications, evaluation methods, and drawbacks*. Medium. Retrieved March 23, 2022, from https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

Piech, C., & Ng, A. (n.d.). *K Means* . CS221. Retrieved March 23, 2022, from https://stanford.edu/~cpiech/cs221/handouts/kmeans.html