Parallel stability-based `k-means`*

Jason Poulos† Emin Arakelian‡ Fadi Kfoury§

Abstract

Abstract...

1. Introduction

Unsupervised learning is a branch of machine learning that infers patterns from data that has no labels. `k-means` is a popular unsupervised algorithm for finding clusters and cluster centers in data. The goal is to choose k cluster centers to minimize the total squared distance between each data point and its closest center. Given k initial centers chosen uniformly at random from the data points, `k-means` alternates between two steps until convergence: (*assignment step*) each point is assigned to the nearest cluster center and; (*update step*) each center is recomputed as the center of mass of all points assigned to it.

`k-means` is NP-hard, and can be solved in time $O(n^{dk+1} \log n)$, where n is the number of points in d dimensions. While it's hard to parallelize `k-means` itself due to its sequential nature, researchers can find shortcuts in the algorithm, or run numerous `k-means` instances on sub-samples to evaluate the stability of clustering. Our project compares sequential and parallelized versions of the stability-based method.

After briefly reviewing other parallel implementations of `k-means` in Section 2, we describe the stability-based method and its parallel implementation in Section 3. We then describe experiments and results in Section 4. Finally, we draw conclusions in Section 5.

2. Related work

`k-means++` chooses only the first cluster center uniformly at random; subsequent centers are selected from the data points, weighed by a probability proportional to its contribution to the overall error. This initialization algorithm is shown to obtain a set of initial centers that is close to the optimum solution [1].

`k-means||` `k-means++` is ill-suited for massive data because it makes k sequential passes over the data points in order to obtain the initial centers. Bahmani et al. [2] propose a method of parallelizing the initialization that reduces the number of passes, which they call `k-means||`. Instead of sampling a single point in each pass, `k-means||` samples $O(k)$ points and repeats for $O(\log n)$ rounds.

3. Stability-based method

Ben-Hur et al. [3] propose an algorithm that uses stability of clustering with respect to perturbations such as sub-sampling as a means of defining meaningful partitions. Computing the stability measure is the bottleneck, so parallelizing the algorithm will involve enhancements to re-use parts of the computations and avoid storing/writing the full matrix multiplication for computing the measure.

3.1. Parallel implementation

4. Experiments

We compare the algorithms using data from a dialect survey, which includes linguistic binary encoded responses [4]. The purpose of the survey is to figure whether a stable number of clusters can be found across the survey participants. We implement the serial and parallel stability-based methods in Python and run all implementations on Edison. Performance is evaluated on three dimensions – clustering cost, running time, and space complexity – for different values of k.

† `poulos@berkeley.edu`
‡ `emin@berkeley.edu`
§ `fadi.kfoury@berkeley.edu`

*The code used for this project is available on Github: `https://github.com/plumSemPy/parallel_kmeans`.

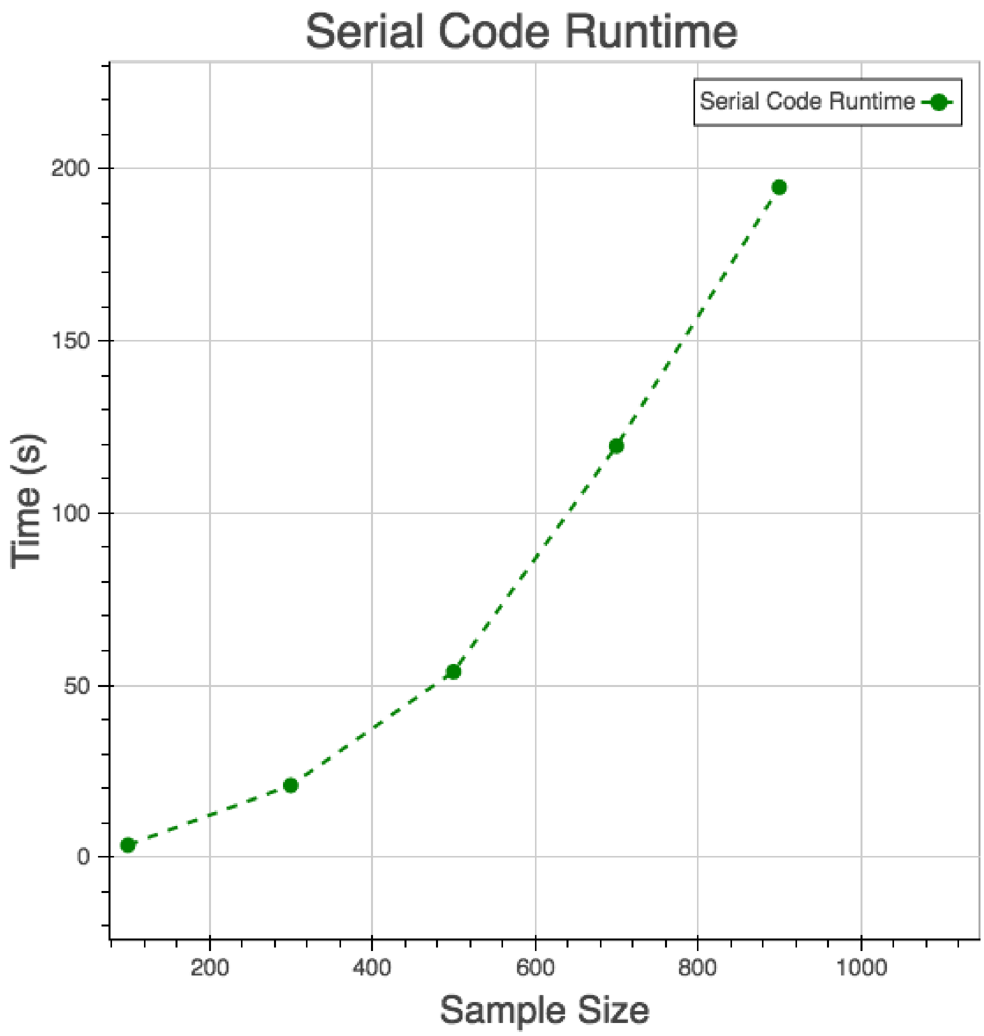

Figure 1. Serial code complexity.

5. Conclusion

References

[1] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[2] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

[3] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Pacific symposium on biocomputing*, volume 7, pages 6–17, 2001.

[4] B. Vaux and S. Golder. The harvard dialect survey. *Cambridge, MA: Harvard University Linguistics Department*, 2003.