Parallel `stable k-means`

Emin Arakelian emin@berkeley.edu
Fadi Kfoury fadi.kfoury@berkeley.edu
Jason Poulos poulos@berkeley.edu

k-means is a popular unsupervised learning 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. One method to determine the optimal number of clusters is stability. Assessing stability requires multiple runs of k-means, which will make it computationally very expensive.

Serial `stable k-means`

Ben-Hur et al.[1] propose an algorithm for any clustering mechanism that will assess the stability of the points and thus let us infer the optimal number of clusters. Let labeling $\mathcal{L}$ be a partition of X into k subsets $S_1, S_2, ..., S_k$. In that case the correlation score between the clustering of intersection of two subsamples is given by:

$$\mathrm{Corr}(\mathcal{L}_1, \mathcal{L}_2) = \frac{\langle \mathcal{L}_1, \mathcal{L}_2 \rangle}{\sqrt{\langle \mathcal{L}_1, \mathcal{L}_1 \rangle \langle \mathcal{L}_2, \mathcal{L}_2 \rangle}}$$

The algorithm for `stable k-means` is displayed in Algorithm 1.

Algorithm 1

Input: X {a dataset}, k_{max} {maximum number of clusters}, num_subsamples {number of subsamples}
Output: $S(i,k)$ {list of similarities for each k and each pair of sub-samples}
Require: A clustering algorithm: cluster(X,k) ; a similarity measure between labels: $s(\mathcal{L}_1, \mathcal{L}_2)$
1: $f = 0.7$
2: for $k = 2$ to k_{max} **do**
3: for $i = 1$ to $num_subsamples$ **do**
4: sub_1 =subsamp(X, f) {a sub-sample with a fraction of f the data}
5: sub_2 =subsamp(X, f)
6: $\mathcal{L}_1$ =cluster(sub_1, k)
7: $\mathcal{L}_2$ =cluster(sub_2, k)
8: Intersect $= sub_1 \cap sub_2$
9: $S(i,k) = s(\mathcal{L}_1(Intersect), \mathcal{L}_2(Intersect))$ {Compute the similarity on the points common to both subsamples}
10: **end for**
11: **end for**

Threading and the GIL

With the existence of GIL we get a cooperative multitasking as shown in Figure 1.

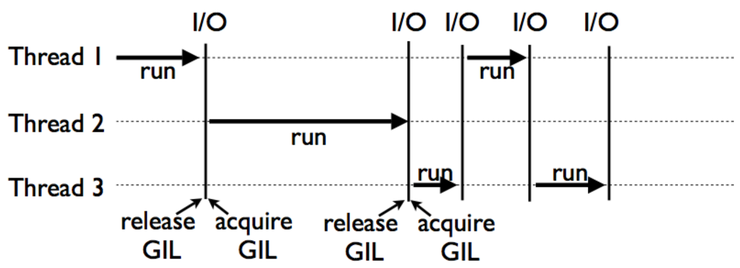

Figure 1: Cooperative multitasking with GIL

CPU-bound threads that never perform I/O are handled differently where there are special check for every 100 "ticks", see Figure 2.

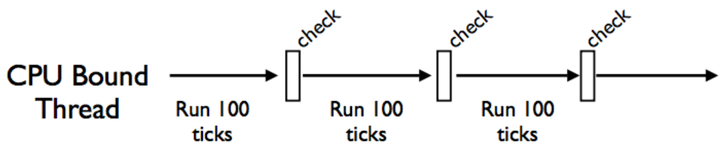

Figure 2: No I/O function checks

In thread switching on a single core two cases can happen, which is illustrated in Figure 3.

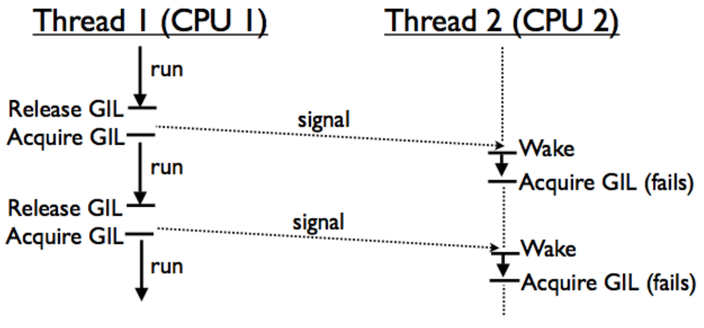

Figure 3: GIL wars on multicored threads

Multiprocessing

A good alternative to multithreading is multiprocessing which came about around Python 2.6. The syntax is exactly the same as multithreading but here python uses processes instead of threads and thus eliminating shared anything. Here we will have multiple instances of the interpreter running each having their own GIL and the way they communicate is message passing. The messages are being passed using "pickle"-ing. The pickle module in Python can serialize and deserialize python objects into byte-streams. Since we are using message passing here, we can deploy our code on multiple machines as well.

Experiments

We will run our algorithm and compare its results on a toy dataset of 500 samples and 2 features. The dataset is plotted in Figure 4. The dataset is obviously split into 5 clusters at most. Running the algorithm on this dataset will yield the following similarity scoring distribution (Figure 5). Figure 6 shows the baseline serial run time; Figure 7 shows the strong and weak scaling of the multithread and multiprocess algorithms; Figure 8 below shows a comparison of the run time on the serial, concurrent (multithread) and parallel (multiprocess) code.

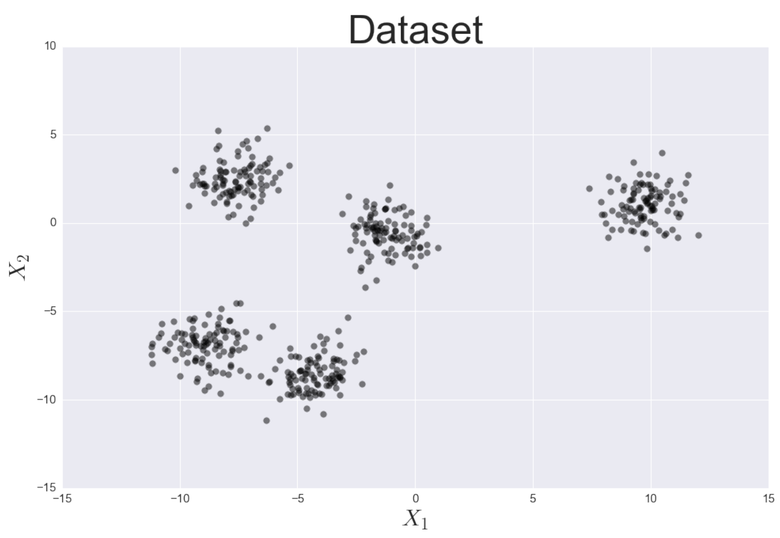

Figure 4: Dataset that will be used to assess `stable k-means`

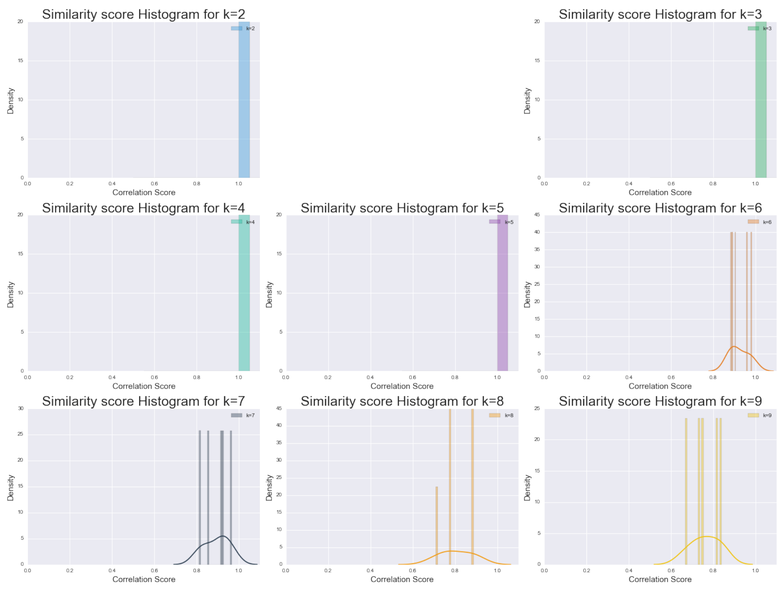

Figure 5: Histograms of Similarity scores for different ks.

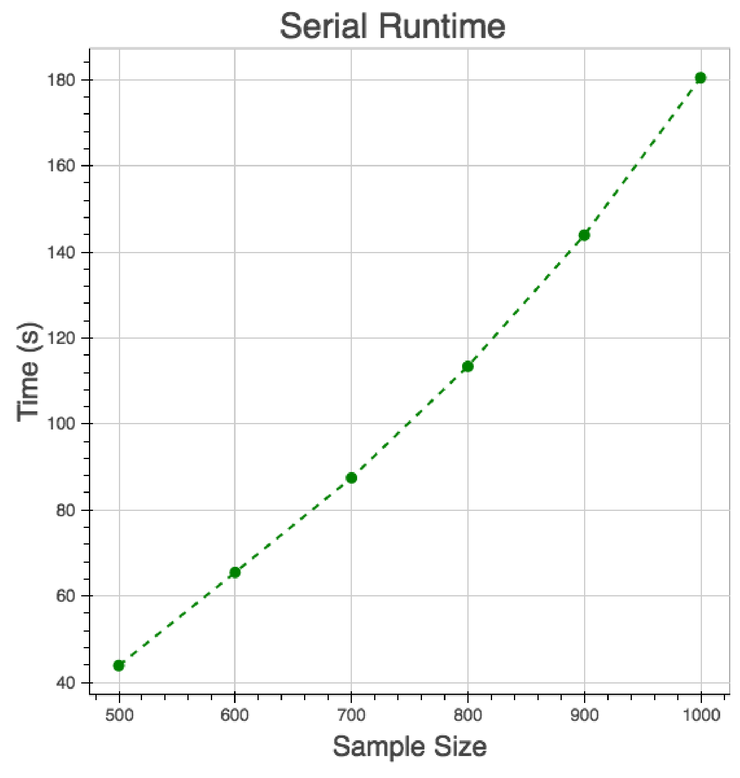

Figure 6: Serial Run Time

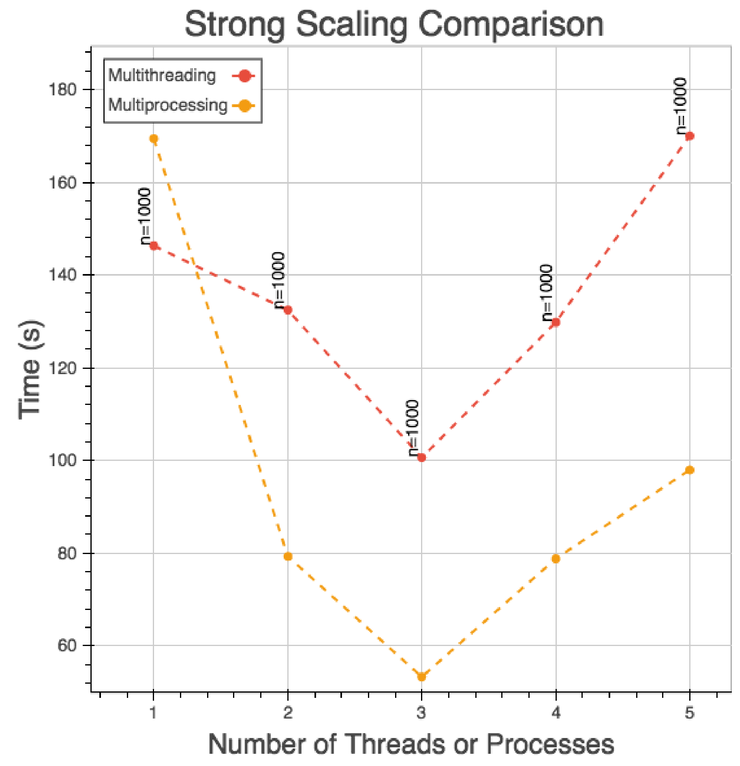

Figure 7: Strong and Weak Scaling Comparison for Multithreading and Multiprocessing

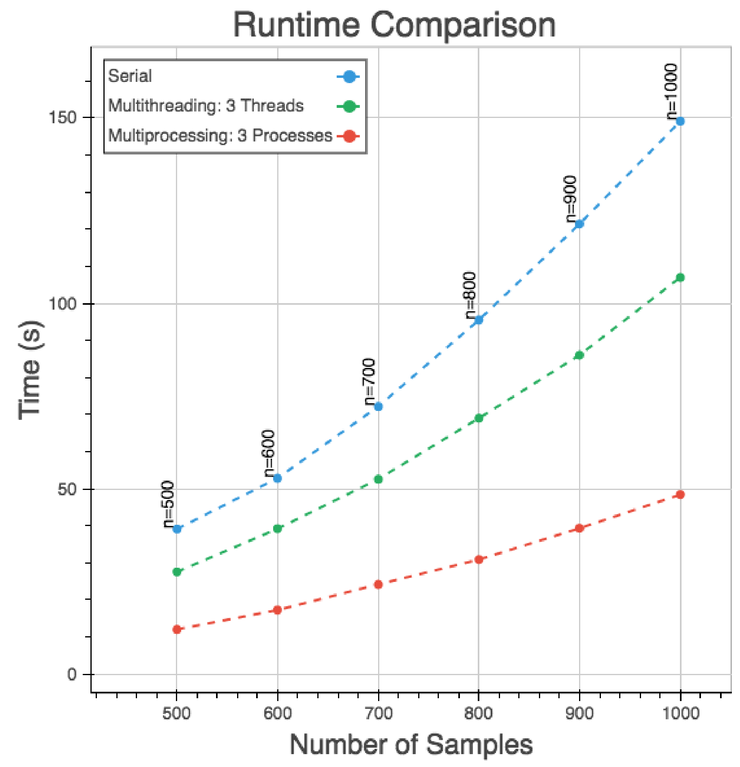

Figure 8: Speed up Comparison

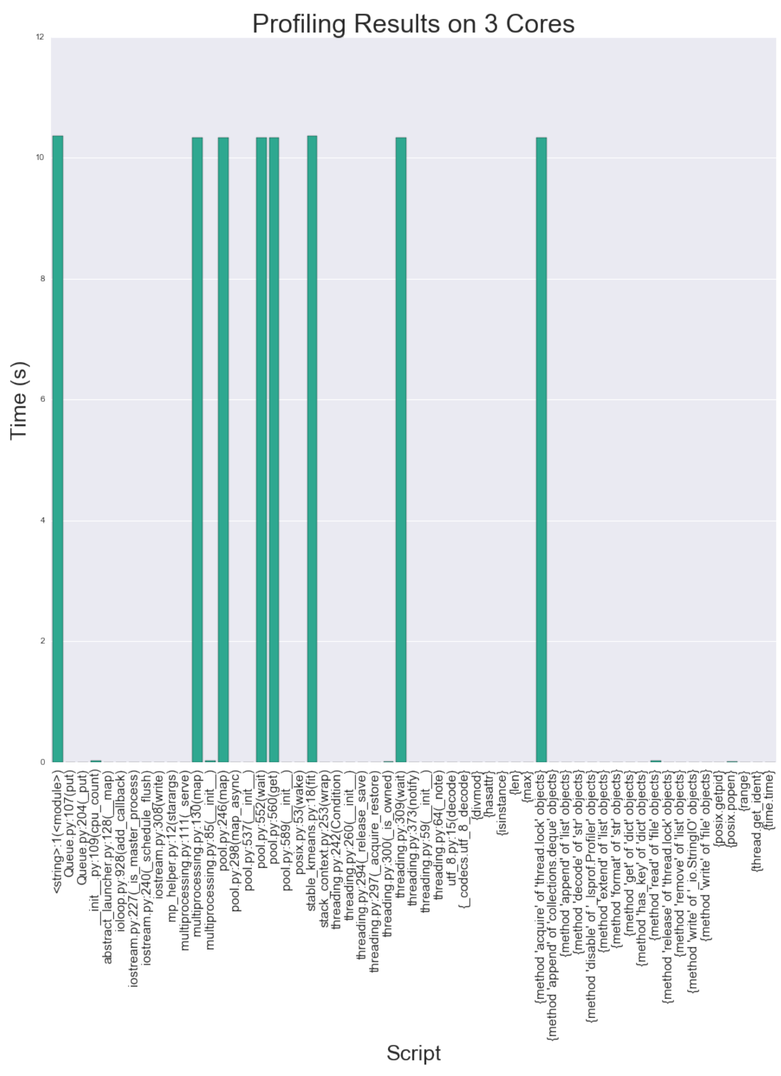

Figure 9: Profiling Result

References

[1] Ben-Hur, Asa and Elisseeff, Andre and Guyon, Isabelle. "A stability based method for discovering structure in clustered data." *Pacific symposium on biocomputing* 7 (2001): 6-17.