CPSC-8650 Data Mining Spring 2019

Final Project

Title:

Lightweight Coreset Construction for K-means and K-medoids Clustering and Performance Analysis.

Team Number: 5

Team Members:

Cai, Ying cai7@clemson.edu

Zhang, Wenxing wenxinz@clemson.edu

1. Introduction

Unsupervised cluster analysis groups a set of objects such that the objects in the same group (called a cluster) are more similar to each other than to those in other clusters. It is a common technique for statistical data analysis, and is widely used in many fields, such as machine learning, pattern recognition, image analysis, bioinformatics, information retrieval, data compression, etc.

There are several common clustering approaches, such as partitioning approach, hierarchical approach, density-based approach, grid-based approach, etc. K-means and K-medoids both are partitioning approaches. Briefly, they partition a dataset of N objects into k clusters, to minimize some chosen evaluation metrics, such as the sum of Euclidean distances. K-means represents each cluster by using the mean center of the cluster, and assigns each point to the closest cluster center. K-medoids represents each cluster by the centroid object of this cluster, and also assigns each point to the closest cluster. They are very popular methods for clustering analysis.

Both of them are heuristic methods. K-means method, which runs in O(n) time with a large hidden constant, is more efficient than K-medoids method, which runs in $O(n^2)$ time. But K-means is more sensitive to outliers and noisy data.

The recent, unprecedented increase in the size of data sets leads to the big computational challenge for them. Coresets can be employed to speed up the inference.

"Coresets are a proven data summarization approach that can be used to scale clustering problems to massive data sets. Coresets are small, weighted subsets of the original data set such that models trained on the coreset are provably competitive with models trained on the full data set." [1]

Coresets have been successfully constructed by some researchers and applied for a lot of clustering problems. Here, Olivier Bachem, Mario Lucic and Andreas Krause proposed a novel approach in their paper "Scalable k-Means Clustering via Lightweight Coresets" which is published by KDD 2018.

They prove that the sufficient coreset size is $O\left(\frac{dklogk+log\frac{1}{\delta}}{\epsilon^2}\right)$, which is smaller than the $O\left(\frac{dklogk}{\epsilon^4}\right)$ which is proved by previous study. A simple and embarrassingly parallel algorithm is proposed too, to construct such lightweight coresets. kmeans++ algorithm is used to solve the clustering problem on subsamples and full dataset.

We reproduce their result using 2 different datasets. We also use the k-medoids method to cluster subsamples and full datasets too.

2. Subsampling/Coresets construction

We use 3 construction methods.

1). Uniform

The samples are generated by uniformly choosing m points at random from datasets. It can be shown that the cost evaluated on the uniformly constructed corset is an unbiased estimator of the true objective function. However, since single points can potentially have a large impact on the objective

function, this subsampling method does not provide the strong theoretical guarantee required by the definition of a coreset.

2). CS

The coresets are constructed with following two steps:

Step1. Find a rough approximation of the optimal clustering. This is achieved by sampling the 1st cluster uniformly at random then iteratively sampling additional points with probability proportional to the minimum squared distance to the already sampled cluster centers. See algorithm 2 for details.

Step2. Use importance sampling to construct the corsets. The idea is to sample points with a potentially high impact on the objective more frequently but assign them a lower weight. The impact of a sample point is measure by the upper bound of its sensitivity, the maximum ratio between the cost contribution of that point and the average contribution of all points. See algorithm 3 for details.

It can be proven that the samples generated by this methods form coresets[7].

Algorithm 2 Mahalanobis D^2-sampling

Require: $\mathcal{X}$, k, d_A

Uniformly sample $x \in \mathcal{X}$ and set $B = \{x\}$.
for $i \leftarrow 2, 3, \ldots, k$
  Sample $x \in \mathcal{X}$ with probability $\frac{\mathrm{d}_A(x,B)}{\sum_{x' \in \mathcal{X}} \mathrm{d}_A(x',B)}$ and add it to B.
return B

Algorithm 3 Coreset construction

Require: $\mathcal{X}$, k, B, m, d_A

$\alpha \leftarrow 16(\log k + 2)$
for each b_i in B
  $B_i \leftarrow$ Set of points from $\mathcal{X}$ closest to b_i in terms of d_A. Ties broken arbitrarily.
$c_\phi \leftarrow \frac{1}{|\mathcal{X}|} \sum_{x' \in \mathcal{X}} \mathrm{d}_A(x', B)$
for each $b_i \in B$ and $x \in B_i$
  $s(x) \leftarrow \frac{\alpha\, \mathrm{d}_A(x,B)}{c_\phi} + \frac{2\alpha \sum_{x' \in B_i} \mathrm{d}_A(x',B)}{|B_i| c_\phi} + \frac{4|\mathcal{X}|}{|B_i|}$
for each $x \in \mathcal{X}$
  $p(x) \leftarrow s(x) / \sum_{x' \in \mathcal{X}} s(x')$
$\mathcal{C} \leftarrow$ Sample m weighted points from $\mathcal{X}$ where each point x has weight $\frac{1}{mp(x)}$ and is sampled with probability $p(x)$.
return $\mathcal{C}$

3). Light weight coresets (LWCS).

The coresets are constructed by sampling m weighted points from datasets. Each point x has weight $\frac{1}{m*q(x)}$ and is sampled with probability $q(x) = \frac{1}{2n} + \frac{d(x,\mu)^2}{2\sum_{x' \in X} d(x',\mu)^2}$ where μ is the mean of the whole dataset X.

$q(x)$ has 2 terms. The intuition behind the second term is that "the points that are far from the mean of the data have a potentially large impact on the quantization error of a clustering." [1]. The distance used in $q(x)$ is the Euclidean distance. Following is the construction algorithm. [1]

Algorithm 1 Lightweight coreset construction

Require: Set of data points $\mathcal{X}$, coreset size m

1: $\mu \leftarrow$ mean of $\mathcal{X}$
2: **for** $x \in \mathcal{X}$ **do**
3: $\quad q(x) \leftarrow \frac{1}{2}\frac{1}{|\mathcal{X}|} + \frac{1}{2}\frac{\mathrm{d}(x,\mu)^2}{\sum_{x' \in \mathcal{X}} \mathrm{d}(x',\mu)^2}$
4: **end for**
5: $C \leftarrow$ sample m weighted points from $\mathcal{X}$ where each point x has weight $\frac{1}{m \cdot q(x)}$ and is sampled with probability $q(x)$
6: **Return** lightweight coreset C

3. Performance analysis.

The performances are valued by the running time and the relative error vs. full datasets.

Running time includes the time used to construct the coresets and the time used to solve clustering problem on subsamples or full dataset using k-means or k-medoids algorithms.

After we cluster the sample set, we have k clusters. Assign each object in full dataset to these k clusters and compute the quantization error $\phi_\chi(Q) = \sum_{x \in \chi} d(x, Q)^2$. This sample quantization error is compared with the quantization error given by solving clustering problem on full set. This relative error is employed to evaluate the clustering quality.

4. Experimental setup

We choose samples size m ∈ {1000, 2000, 5000, 10000, 20000} and clusters k ∈ {100, 500}.

Programs are coded using Python. The multi-processing package is used to construct coresets. The sklearn/cluster package is employed to solve the k-means clustering problem. Since we don't find a k-medoids package which handles weighted points, we write the algorithm ourselves.

Require: subsample X, number of clusters k, max-iteration maxi, tolerance t

1. Randomly choose k medoids from X
2. while iterations < maxi or reduced cost > t:
    for each object in X:
        assign cluster
    for each cluster:
        for each object O in this cluster:
            cost = $\sum_{x \in X} weight(x) * d(x, O)^2$
        choose the object whose cost is minimum as the new medoid of this cluster
3. Return medoids of k clusters.

5. Environment

We use the Clemson University Palmetto Cluster. The simulations for samples/coresets generation and error analysis are performed on single CPUs with 15 to 60GB memory. The lightweight coresets are constructed in parallel, we used 8 cores in order to speed up the construction. For time analysis, all simulations are run on single CPU with 60 GB memory to ensure valid comparisons.

6. Datasets

Two datasets are analyzed in this study.

1) KDD — 145'751 samples with 74 features measuring the match between a protein and a native sequence [3].
2) SONG — 90 features from 515'345 songs of the Million Song datasets used for predicting the year of songs [5].

The datasets downloaded from the reference are converted to files containing only feature values separated by whitespace using "awk" first before further analysis such as subsampling or K-means.

Due to time limitations, for SONG dataset, k-medoids are not performed.

7. Results and Discussions

Relative error and speedup of different methods vs. Full for K-means								
			Relative error vs. Full			Speedup vs. Full		
k	Data	Method	m = 1000	m = 2000	m = 5000	m = 1000	m = 2000	m = 5000
100	KDD	UNIFORM	1.82(1.38)	1.13(1.02)	1.67(0.57)	555	334	126
		CS	0.25(0.14)	0.14(0.06)	0.07(0.01)	0.1	0.1	0.1
		LWCS	0.24(0.01)	0.16(0.01)	0.09(0.01)	32	32	27
	SONG	UNIFORM	0.22(0.01)	0.16(0.01)	0.10(0.01)	1921	1251	484
		LWCS	0.21(0.009)	0.16(0.008)	0.11(0.005)	30	30	28
500	KDD	UNIFORM	2.72(0.04)	1.03(1.46)	0.66(0.81)	630	417	169
		LWCS	0.37(0.01)	0.30(0.01)	0.19(0.004)	121	110	77
	SONG	UNIFROM	0.44(0.02)	0.32(0.01)	0.22(0.01)	516	316	137
		LWCS	0.33(0.007)	0.26(0.005)	0.20(0.004)	29	28	25

Relative error and speedup of different methods vs. Full for K-medoids								
			Relative error vs. Full			Speedup vs. Full		
k	Data	Method	m = 1000	m = 2000	m = 5000	m = 1000	m = 2000	m = 5000
100	KDD	UNIFORM	1.48(0.69)	1.05(0.74)	0.42(0.55)	82	21	6
		LWCS	-0.24(0.01)	-0.25(0.04)	-0.18(0.12)	22	15	6
500	KDD	UNIFORM	1.40(1.04)	1.60(0.93)	0.35(0.55)	2594	1038	203
		LWCS	-0.33(0.01)	-0.35(0.01)	-0.35(0.01)	39	23	10

- With KDD dataset, the UNIFORM subsampling gives significantly larger errors compared to the other two coresets construction methods. And the standard derivations are much higher too.
- The CS method takes a long time to generate the coresets which largely limits its usage in practice. It might still be useful when memory is the bottleneck for performing clustering methods. A potential improvement can be implement CS algorithm in parallel. Due to time limitations, we didn't do that in this project.
- The UNIFORM method provides the best speedup, but performance depends on different datasets. If a dataset has a small fraction of points that have a large impact on the objective function, the UNIFORM method is likely to perform poorly.

- The LWCS gives the best performance in general. It gives noticeable speedup compared with full dataset. And the relative errors are smaller than UNIFORM.
- With the Kmedoids, the LWCS gives negative relative error. This could indicate that when running Kmedoids on the full dataset, the maximum number of iterations (we used 300) is too small for the algorithm to converge to a local minimum. Another possible explanation is that the bad choice of random seed when running Kmedoids on the full dataset lead the algorithm converge to a local minimum that have relatively high quantization error. Potential solutions to this could be using a larger "maximum number of iterations", running Kmedoid multiple times with different random seeds on the same datasets/coresets/samples and choosing the clustering with the smallest quantization errors.

8. References

1) Olivier Bachem, Mario Lucic, Andreas Krause. Scalable k-Means Clustering via Lightweight Coresets. KDD 2018, August 19-23, 2018, London, United Kingdom.
2) Mario Lucic, Olivier Bachem, and Andreas Krause. 2016. Strong Coresets for Hard and Soft Bregman Clustering with Applications to Exponential Family Mixtures. In Artificial Intelligence and Statistics.
3) KDD Cup. 2004. Protein Homology Dataset. Available at http://osmot.cs.cornell.edu/kddcup/datasets.html.
4) Matthew Faulkner, Michael Olson, Rishi Chandy, Jonathan Krause, K. Mani Chandy, and Andreas Krause. 2011. The Next Big One: Detecting Earthquakes and other Rare Events from Community-based Sensors. In *Information Processing in Sensor Networks*.
5) Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *International Conference on Music Information Retrieval*.
6) Andrew V Uzilov, Joshua M Keegan, and David H Mathews. 2006. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics* (2006).
7) Mario Lucic, Olivier Bachem, and Andreas Krause. Strong Coresets for Hard and Soft Bregman Clustering with Applications to Exponential Family Mixtures. arXiv e-prints, page arXiv:1508.05243, Aug 2015.