Estimating Dictionary Size For Representation Learning

Zhenye Lin
Williams College, Williamstown, MA
zl3@wlliams.edu

ABSTRACT

Sparse coding is a class of algorithms for learning concise representations from unlabeled data. The goal of the sparse representation learning methods is to learn the *dictionary*, the set of atoms that effectively summarize the dataset, as well as the *sparse codes*, the linear combination of atoms in the dictionary for each data instance. Despite the abundance of methods for dictionary learning, the k-means algorithm remains one of the most popular choices due to its simplicity, efficiency, and high accuracy. Unfortunately, estimating the size of the dictionary, which directly affects the quality of sparse codes, remains largely unexplored. In this paper, we first study how measures that assess the clustering quality correlate with the reconstruction quality of sparse coding methods. Then, we design an hierarchical version of k-means that relies on the best performing clustering quality measure to navigate the possible choices of atoms for inclusion in the dictionary by avoiding the expensive construction of sparse codes in each step. Our extensive evaluation across 85 datasets demonstrates the robustness of our method, which achieves significant runtime improvement over the k-means algorithm without significant loss in terms of accuracy. In addition, the findings of our study may have implications in subsequent designs of dictionary learning methods.

ACM Reference Format:
Zhenye Lin. 2019. Estimating Dictionary Size For Representation Learning. In *Proceedings of 2019 International Conference on Management of Data (SIGMOD 19).* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/TBD

1 INTRODUCTION

Sparse coding is a class of unsupervised methods for learning sets of basis vectors to effectively represent data as a linear combination of the basis vectors. More formally, sparse coding methods find a set of k basis vectors, $\Phi = \{\phi_1, ..., \phi_k\}$, such that each input vector x is constructed as a linear combination of these basis vectors: $x = \sum_1^k c_i \phi_i$, where c_i is the coefficient of each basis vector. Despite the plethora of methods for dictionary learning, the k-means algorithm is one of the most popular choices due to its simplicity, efficiency, and high accuracy. Unfortunately, there is a dearth of research in estimating the size of the dictionary (i.e., number of atoms) for sparse coding methods, which directly affects the quality of the sparse representation. In this paper, we first study how measures that assess the clustering quality correlate with the reconstruction quality of sparse coding methods. We find noticeable trends that we leverage in our hierarchical version of k-means for estimating the number of atoms in the dictionary. Considering the rise of Internet of Things applications, we focus on databases of time series.

2 BACKGROUND

Sparse coding mechanisms require the selection of a method for dictionary learning and for constructing sparse representations. When k-means is the choice for dictionary learning, the procedure is as follows:

(1) Run k-means for a user-defined k value.
(2) Use the k centroids as the dictionary.
(3) Optimize an objective function to learn for each data instance the coefficients associated with a subset of the k centroids.

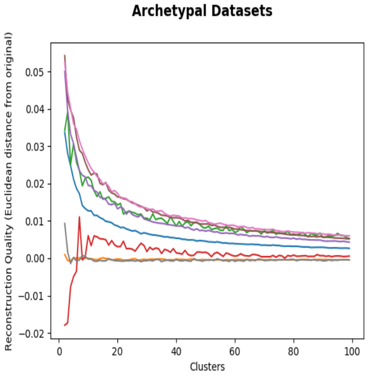
In this study, we optimize the *LASSO* function [7], which is one of the most widely used objective functions for sparse representation learning. Next, we examine the correlation between clustering quality measures and the reconstruction quality. Based on our findings, we propose a novel method for computing atoms of the dictionary by avoiding the expensive LASSO computation for each time series.

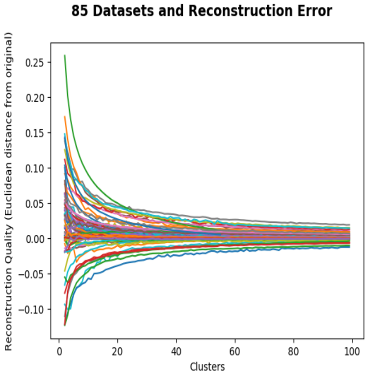
3 EXPERIMENT APPROACH

We follow the above described k-means *sparse coding* algorithms, but we alter the number of atoms/centroids in our dictionary and test the effect of changing the size of the dictionary on the quality of reconstruction. We used the UCR dataset [8] (the standard for time series), which contains 85 different datasets of various size and time series length. The nature of UCR datasets ensures the robustness of our experimental result. Since *sparse coding* assumes the data is unlabeled, we combine the training and the testing data set into a single large data set. The datasets are preprocessed by normalizing all the time series to be between -1 and 1. The nature of the *sparse coding* algorithm suggests that the number of atoms in the dictionary scales with the quality of the reconstruction. The higher the number of atoms, the higher the reconstruction quality, but also the higher the runtime. Thus, we should be able to achieve a trade off between speed and accuracy of clusters in the reconstruction of the dictionary.
Proposal: We propose to examine three indexes that evaluate the clustering quality in conjunction with the reconstruction quality of the time series. The clustering metrics are Calinski-Harabaz score, Silhouette score, and Davies-Bouldin score [1, 3, 5]. These three clustering metrics test for quality of unlabeled clusters. Each uses a form of Euclidean distance metric to evaluate the distinctness of each cluster. If there is a strong correlation between specific indexes, we may skip measuring the reconstruction quality step (the *Lasso*) and evaluate the clustering quality metric at each step. We examine two versions of the K-Means Algorithm, K-Means Flat

SIGMOD 19, June 2019,

ACM ISBN None.
https://doi.org/TBD





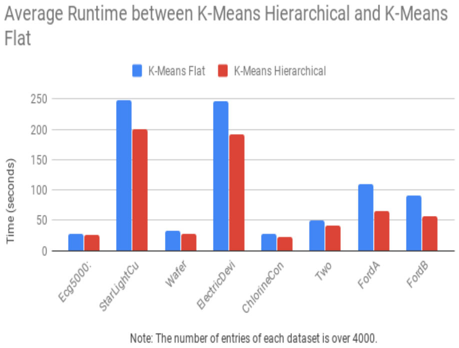


Table 1: Correlation between Clustering quality metrics and Reconstruction Quality across 85 datasets

Metric	Pearson Correlation with Distance
Calinski Harabaz score	0.759
Silhouette score	0.629
Davies Bouldin score	0.530

Version and our proposed K-Means Hierarchical Version, that attempts to improve the runtime of the algorithm based on clustering index result.

K-Means Flat Version: Run k-means from 2 to min(ceil(0.5*n), 100). [1] For each k, we test its correlation between different clustering quality metrics, the reconstruction quality, and runtime.

K-Means Hierarchical Version: In the K-means Hierarchical Version, we apply a Depth First Search (DFS) approach. We first select a small number of k centroids in the first iteration of the K-means algorithm. In each centroid, we split the centroids into more centroids by applying the k-means algorithm. If the clustering quality metric improves with this split, we add the new centroids to the dictionary and delete the original centroid. Otherwise, we keep the original centroid. The depth of each centroid search is chosen to be arbitrary. This hierarchical version is inspired by Petitjean's hierarchical classification of indexes of time series. [6]

Evaluation Metric: Reconstruction quality is measured by taking the average distance between the constructed time series and the original time series. [2]

4 RESULT

The two graphs above indicate that there is a strong correlation between the number of clusters and the reconstruction quality. The Euclidean distance between the reconstructed time series and the original time series decreases as we increase the number of atoms in our dictionary. Eventually, the distance plateaus. Thus, we aim can find the best size of the dictionary by applying this algorithm and stopping before it plateaus.

Correlation between clustering quality indexes: The result of our first proposal found substantial correlation between the Calinski-Harabaz score and our clustering quality. The Pearson Correlation between the reconstruction quality and the Calinski-Harabaz score is 0.759. The other two indexes, while they are somewhat correlated, are not sufficient to replace the LASSO reconstruction entirely. This indicates that while it is not a perfect substitute for the LASSO reconstruction, we are able to skip the LASSO reconstruction in cases of large datasets or datasets where one values speed over a small amount of loss. In general, the LASSO function takes on average 35.6% of the time of the overall algorithm. When we are searching for the best number of k, we may skip the LASSO function entirely, leading to an substantial decrease in runtime.

K-Means flat vs K-Mean Hierarchical: The result of our second proposal found that the K-Means flat version performs better than the hierarchical version. Although the difference is not substantial, the K-Means flat version performs 0.00214 distance better than the K-Means hierarchical version across 85 datasets. This result indicates that while there is an increase in the reconstruction quality of each cluster in each DFS, the increase is smaller than the complete repartitioning of clusters in the K-means flat version. For larger datasets with time series entries over 4000, we saw a substantial decrease in runtime by using the hierarchical version of the data with a similar amount of losses. We've seen a decrease in runtime by as much as 41.49%. Overall, the runtime decrease in all the datasets with number of entries above 4000 is 21.89%.

5 CONCLUSION AND FUTURE WORK

There are two major takeaways from our experiments. First, we observed a substantial correlation between the reconstruction quality of the time series and the clustering quality metric of the Calinski-Harabaz score. The correlation is substantial enough for us to skip the expensive *LASSO* function. By leveraging this metric, we built the k-means flat version, that does not uses this metric, and the k-means hierarchical version, that does. In the hierarchical version, we observed an average drop of 21.89% in runtime for datasets with over 4000 entries compared to the K-means flat version. In the future, we plan on testing this algorithm on longer and larger time series datasets. Furthermore, we aim to replicate our findings with faster K-Means approximation algorithms such as K-means++ and AFKMC2 [2, 4].

[1] for big datasets the number of k will be from k=2 to 100 and for small datasets k=2 to 10-20-30

[2] All tests are performed on Chameleon Cloud Testbed Haswell machine.

ACKNOWLEDGMENTS

The author would like to thank John Paparrizos for his advice, his teaching, and everlasting support for this project. The author would also like to thank Aaron J. Elmore for his insightful comments and all participants and mentors of the REU BigDataX for their support. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

REFERENCES

[1] T. CaliÅĎski and J Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics* 3, 1 (1974), 1–27. https://doi.org/10.1080/03610927408827101 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/03610927408827101

[2] Sergei Vassilvitskii David Arthur. 2007. k-means++: The Advantages of Careful Seeding. *SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), 1027–1035.

[3] D. L. Davies and D. W. Bouldin. 1979. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1, 2 (April 1979), 224–227. https://doi.org/10.1109/TPAMI.1979.4766909

[4] S. Hamed Hassani Andreas Krause Olivier Bachem, Mario Lucic. [n. d.]. Fast and Provably Good Seedings for k-Means. ([n. d.]).

[5] Peter Rousseeuw. 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.* 20, 1 (Nov. 1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

[6] Chang Wei Tan, Geoffrey I. Webb, and FranÃğois Petitjean. [n. d.]. *Indexing and classifying gigabytes of time series under time warping.* 282–290.

[7] R. Tibshirani. 1996. Regression shrinkage and selection via the lasso. *J. Royal. Statist.* 58, 1 (1996).

[8] Bing Hu Nurjahan Begum Anthony Bagnall Abdullah Mueen Yanping Chen, Eamonn Keogh and Gustavo Batista. [n. d.]. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/