MATLAB Implementation Details of a Scalable Spectral Clustering Algorithm with the Cosine Similarity

Guangliang Chen(✉)

Department of Mathematics & Statistics, San José State University,
San José, CA 95192-0103, USA
guangliang.chen@sjsu.edu

Abstract. We present the implementation details of a scalable spectral clustering algorithm with cosine similarity (ICPR 2018, Beijing, China), which are based on simple, efficient matrix operations. The sensitivity of its parameters is also discussed.

1 Introduction

In our recent work [1] we introduced a scalable implementation of various spectral clustering algorithms, such as the Ng-Jordan-Weiss (NJW) algorithm [3], Normalized Cut (NCut) [4], and Diffusion Maps (DM) [2], in the special setting of cosine similarity by exploiting the product form of the weight matrix. We showed that if the data $\mathbf{X} \in \mathbb{R}^{n\times d}$ is large in size (n) but has some sort of low dimensional structure – either of low dimension (d) or being sparse (e.g. as a document-term matrix), then one can perform spectral clustering with cosine similarity solely based on three kinds of efficient operations on the data matrix: *elementwise manipulation*, *matrix-vector multiplication*, and *low-rank SVD*, before the final k-means step. As a result, the algorithm enjoys a linear complexity in the size of the data. We present the main steps of the algorithm in Algorithm 1 and refer the reader to the paper [1] for more details.

Remark 1. The outliers detected by the algorithm may be classified back to the main part of the data set by simple classifiers such as the nearest centroid classifier, or the k nearest neighbors (kNN) classifier.

2 Implementation Details

We implemented Algorithm 1 and conducted all the experiments in MATLAB. Note that the *Statistics and Machine Learning Toolbox* is needed because of the k-means function used in the final step (the rest of the steps consist of very basic linear algebra operations). If unavailable, the toolbox may be avoided if one uses a freely-available substitute k-means function such as *litekmeans*.[1]

[1] Available at http://www.cad.zju.edu.cn/home/dengcai/Data/Clustering.html.

Algorithm 1. Scalable Spectral Clustering with Cosine Similarity

```
Input: Data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ (sparse or of moderate dimension, with $L_2$-normalized rows), #clusters $k$, clustering method (NJW, Ncut, or DM), fraction of outliers $\alpha$
Output: Clusters $C_1, \ldots, C_k$ and a set of outliers $C_0$
1: Calculate the degree matrix $\mathbf{D} = \operatorname{diag}(\mathbf{X}(\mathbf{X}^T\mathbf{1}) - \mathbf{1})$ and remove the bottom $(100\alpha)\%$ of the input data that have the lowest degrees as outliers (stored in $C_0$).
2: For the remaining data, normalize them by $\widetilde{\mathbf{X}} = \mathbf{D}^{-1/2}\mathbf{X}$ and find its top $k$ singular values $\lambda_1, \ldots, \lambda_k$ and corresponding left singular vectors $\widetilde{\mathbf{u}}_1, \ldots, \widetilde{\mathbf{u}}_k$ by rank-$k$ SVD. Let $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_k) \in \mathbb{R}^{k \times k}$ and $\widetilde{\mathbf{U}} = [\widetilde{\mathbf{u}}_1 \ldots \widetilde{\mathbf{u}}_k] \in \mathbb{R}^{n \times k}$.
3: Form the matrix $\mathbf{Y} \in \mathbb{R}^{n \times k}$ dependent on the clustering method (the rows of $\mathbf{Y}$ are regarded as an embedding of the input data):
    – NJW: $\mathbf{Y} = \widetilde{\mathbf{U}}$;
    – NCut: $\mathbf{Y} = \mathbf{D}^{-1/2}\widetilde{\mathbf{U}}$
    – DM: $\mathbf{Y} = \mathbf{D}^{-1/2}\widetilde{\mathbf{U}}\mathbf{\Lambda}^t$, where $t$ is a positive integer representing the number of diffusion steps.
4: Normalize the rows of $\mathbf{Y}$ to have unit $\ell_2$ norm and apply the $k$-means algorithm to find $k$ clusters $C_1, \ldots, C_k$.
```

To promote the simplicity and efficiency of the implementation, we did the following things:

– We extended the value of the parameter t (hidden in DM) to include the other two clustering methods: NCut ($t = 0$) and NJW ($t = -1$).
– When the input data matrix $\mathbf{X}$ is sparse (e.g., as a document-term matrix), we take advantage of the sparse matrix operations in MATLAB.
– Diagonal matrices are always stored as vectors.
– All the multiplications between a matrix and a diagonal matrix (such as $\mathbf{D}^{-1/2}\mathbf{X}$ and $\mathbf{D}^{-1/2}\widetilde{\mathbf{U}}\mathbf{\Lambda}$), are implemented as element-wise binary operations through the *bsxfun* function in MATLAB. Additionally, the matrix-vector product $\mathbf{X}^T\mathbf{1}$ is implemented through *transpose(sum(X*, 1*))* in MATLAB.
– The *svds* function is used to find only the top k singular values and associated singular vectors of $\widetilde{\mathbf{X}}$.
– The k-means clustering is initialized with the default *plus* option, and uses 10 restarts.
– We implemented the nearest centroid classifier for assigning the outliers back into the clusters due to its faster speed than k-NN.

The software, as well as the data sets used in [1] and this paper, has been published at https://github.com/glsjsu/rprr2018.

3 Parameter Setting

The algorithm has only one parameter α that needs to be specified. It indicates the fraction of input data to be removed and treated as outliers.

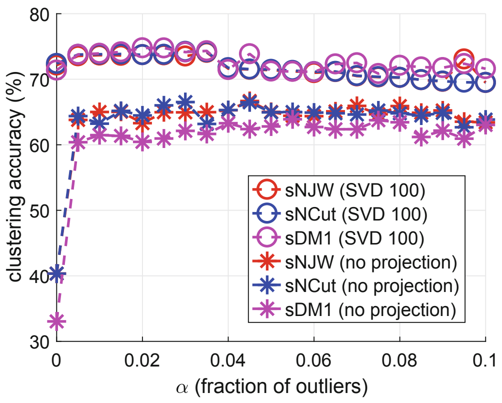

Fig. 1. Clustering accuracy rates of the three scalable methods on both versions of 20 newsgroups data corresponding to different α values.

Experiments conducted in [1] showed that the parameter α was not sensitive in the scalable NJW algorithm, as long as it is set bigger than zero.[2] Here, we further test the sensitivity of the α parameter in all three scalable methods on the same two versions of 20 newsgroups data [1] and report the accuracy results in Fig. 1.

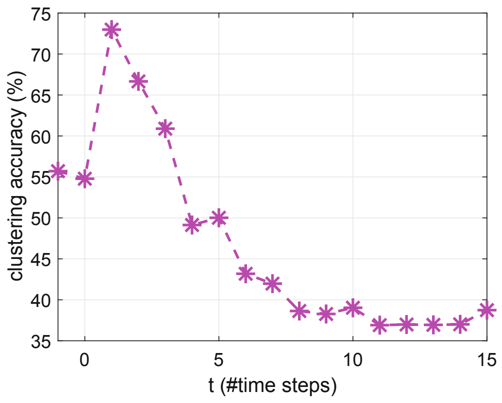

Fig. 2. Clustering accuracy of Algorithm 1 with DM and each value of $t = -1 : 15$ obtained on the top 30 categories of the TDT2 data set [1] ($\alpha = .01$ is always fixed). Note that the two special values $t = -1, 0$ correspond to the NJW and NCut options.

There is another parameter t in the code that needs to be specified, which in the case of DM represents the number of steps taken by a random walker.

[2] This is actually a necessary condition for the algorithm to work; see [1, Section III.B]. The value zero was included to verify the necessity of the condition.

In general, its optimal value is data-dependent. We have observed that DM with $t = 1$ often gives better accuracy than NCut (corresponding to $t = 0$); see Fig. 2 for an example.

4 Conclusions

We presented the MATLAB implementation details of a scalable spectral clustering algorithm with cosine similarity. The code consists of a few lines of simple linear algebra operations and is very efficient and fast. There are two parameters associated to the algorithm – α and t – but they are easy to tune: for the former, it is insensitive and we observed that the value $\alpha = .01$ often works adequately well; for the latter, it is truly a parameter when DM is used and in that case, setting it to $t = 1$ seems to achieve good accuracy in most cases. Lastly, all steps except the last step of k-means clustering in Algorithm 1 are deterministic and thus for fixed data and parameter values, the code yields very consistent results (any inconsistency is caused by the k-means clustering step).

Acknowledgments. We thank the anonymous reviewers for helpful feedback. G. Chen was supported by the Simons Foundation Collaboration Grant for Mathematicians.

References

1. Chen, G.: Scalable spectral clustering with cosine similarity. In: Proceedings of the 24th International Conference on Pattern Recognition (ICPR), Beijing, China, pp. 314–319 (2018)
2. Coifman, R., Lafon, S.: Diffusion maps. Appl. Comput. Harmon. Anal. **21**(1), 5–30 (2006)
3. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: analysis and an algorithm. Adv. Neural Inf. Process. Syst. **14**, 849–856 (2001)
4. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)