JYX JYVÄSKYLÄN YLIOPISTO UNIVERSITY OF JYVÄSKYLÄ

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Hämäläinen, Joonas; Kärkkäinen, Tommi; Rossi, Tuomo

Title: Scalable robust clustering method for large and sparse data

Year: 2018

Version: Published version

Copyright: © Authors, 2018

Rights: In Copyright

Rights url: http://rightsstatements.org/page/InC/1.0/?language=en

Please cite the original version:

Hämäläinen, J., Kärkkäinen, T., & Rossi, T. (2018). Scalable robust clustering method for large and sparse data. In ESANN 2018 : Proceedings of the 26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (pp. 449-454). ESANN. https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2018-134.pdf

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

Scalable robust clustering method for large and sparse data

Joonas Hämäläinen, Tommi Kärkkäinen* and Tuomo Rossi

University of Jyvaskyla, Faculty of Information Technology,
P.O. Box 35, FI-40014 University of Jyvaskyla, Finland

Abstract. Datasets for unsupervised clustering can be large and sparse, with significant portion of missing values. We present here a scalable version of a robust clustering method with the available data strategy. More precisely, a general algorithm is described and the accuracy and scalability of a distributed implementation of the algorithm is tested. The obtained results allow us to conclude the viability of the proposed approach.

1 Introduction

Clustering is one of the core techniques in unsupervised learning. Based on a similarity measure (e.g., Euclidean distance), its purpose is to partition a given data into groups, clusters, where members belonging to one cluster are similar to each other and dissimilar to other clusters. Classically, clustering is divided into two main categories, partitional and hierarchical, although a large variety of different approaches have been suggested [1, 2].

Since the real-world clustering problems are becoming larger and larger, applying sequential clustering algorithms to these problems becomes impractical. Over the years, a lot of research related to the parallellizing of the well-known K-means algorithm with various parallel computation models has been carried out [3, 4, 5]. K-means|| [6] is parallelizable version of the K-means++ [7]. Contrary to K-means++, imposed by the inherently sequential nature, K-means|| is scalalable and it can be easily implemented in parallel with multiple parallel programming models. As shown by [6], proper initialization of a parallel algorithm plays an important role both in accuracy and scalability.

K-spatialmedians is prototype-based clustering method which applies available data strategy and spatial median as cluster prototype [8]. The available data strategy refers to an approach, where all distance computations are projected to the available values. This ensures that no assumptions on the unknown distribution of the missing values (MVs) is being made during clustering. Robustness and accuracy of the approach for tens of percents of MVs was extensively tested in [9]. However, differently to the use of the mean as in K-means, one needs to apply an iterative method to compute the cluster prototype. Hence, scalability of the parallel implementation is not self-evident. Therefore, the purpose in this article is twofold: i) to compare clustering results between K-means and K-spatialmedians, ii) to consider scalability of a parallel implementation of K-spatialmedians

*The work of TK has been supported by the Academy of Finland from the projects 311877 (Demo) and 315550 (HNP-AI)

2 Parallel K-spatialmedians||

Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ denote a dataset in M dimensional space and let $\mathbf{P} = \{\mathbf{p}_1, ..., \mathbf{p}_N\}$ be a $N \times M$ projection matrix where

$$(\mathbf{p}_i)_j = \begin{cases} 1, \text{if } (\mathbf{x}_i)_j \text{ exists}, \\ 0, \text{otherwise}. \end{cases} \tag{1}$$

The clustering error function that is, after an initialization, locally minimized by the K-spatialmedians algorithm reads as [10, 11]

$$\mathcal{J}(\{\mathbf{m}_k\}_{k=1}^K) = \sum_{i=1}^N \min_{k=1,...,K} \| \operatorname{Diag}(\mathbf{p}_i)\,(\mathbf{x}_i - \mathbf{m}_k)\|_2, \tag{2}$$

where $\operatorname{Diag}(\mathbf{p}_i)$ creates a diagonal matrix using a vector $\mathbf{p}_i$. The result of the minimization is the set of prototypes $\{\mathbf{m}_k\}_{k=1}^K$, with the cluster memberships $\mathbf{C}_k = \{i : \| \operatorname{Diag}(\mathbf{p}_i)\,(\mathbf{x}_i - \mathbf{m}_k)\|_2 \le \| \operatorname{Diag}(\mathbf{p}_i)\,(\mathbf{x}_i - \mathbf{m}_{k'})\|_2 \text{ for } 1 \le k \ne k' \le K\}$. Multiplication with $\mathbf{p}_i$ in (2) realizes the projection of the distance computation to only the available values of individual observations. As the definition (2) suggests, the iterative relocation of cluster prototypes simply means that one needs to solve the minimization problem iterative in each cluster. For this purpose, successive over-relaxation (SOR) of the well-known Weiszfeld algorithm for a candidate solution can be used [8, 9].

Let us assume that the data is partitioned into Q disjoint subsets: $\mathbf{X} = \{\mathbf{X}_1, ..., \mathbf{X}_Q\}$ such as $\mathbf{X} = \cup_{i=1}^Q \mathbf{X}_i$. Then the cluster memberships are spread to data partitions such as $\mathbf{C}_k = \{\mathbf{C}_{k1}, ..., \mathbf{C}_{kQ}\}$. Moreover, we denote $\mathbf{C}_{kq} = \mathbf{C}_k \cap \{i : \mathbf{x}_i \in \mathbf{X}_q\}$, where $q = 1, ..., Q$. Hence, in the SOR algorithm from the current step t into $t+1$, the candidate prototype $\mathbf{v}_k$ (see [9], p. 138) for the kth cluster can be solved with

$$\begin{aligned} \mathbf{v}_k &= (\sum_{i \in \mathbf{C}_k} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i))^{-1} \sum_{i \in \mathbf{C}_k} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)\mathbf{x}_i \\ &= (\sum_{q=1}^Q \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i))^{-1} \sum_{q=1}^Q \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)\mathbf{x}_i, \end{aligned}$$

where $\alpha_i^t = 1/\sqrt{\| \operatorname{Diag}(\mathbf{p}_i)(\mathbf{u}_k^t - \mathbf{x}_i)\|_2^2 + \epsilon}$, where ϵ is a small positive constant. If we define $\mathbf{A}_{qk}^t = \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)$ and $\mathbf{b}_{qk}^t = \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)\mathbf{x}_i$, we get

$$\mathbf{v}_k = (\sum_{q=1}^Q \mathbf{A}_{qk}^t)^{-1} \sum_{q=1}^Q \mathbf{b}_{qk}^t. \tag{3}$$

Finally, the prototype $\mathbf{u}_k$ is updated as follows

$$\mathbf{u}_k^{t+1} = \mathbf{u}_k^t + \omega(\mathbf{v}_k - \mathbf{u}_k^t), \tag{4}$$

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

Algorithm 1: K-spatialmedians||

```
Input: Data partitions $\mathbf{X} = \{\mathbf{X}_1, ..., \mathbf{X}_Q\}$, projection matrix partitions $\mathbf{P} = \{\mathbf{P}_1, ..., \mathbf{P}_Q\}$,
    the number of clusters $K$, the maximum number of SOR iterations $t_{max}$, the
    threshold for convergence of SOR $\epsilon_{tol}$.
Output: Final prototypes $\{\mathbf{m}_k\}_{k=1}^K$.
 1: Initialize $\{\mathbf{m}_k\}_{k=1}^K$ with parallel K-spatialmedians$||^0$ for the complete rows in $\mathbf{X}$.
    (master and slaves)
 2: Broadcast $\{\mathbf{m}_k\}_{k=1}^K$ to all $Q$ slave processes. (master)
 3: Assign local cluster memberships $\mathbf{C}_{kq}$ for $k = 1, ..., K$. (slaves)
 4: Set $t = 0$ and $\mathbf{u}_k^t = \mathbf{m}_k$ for $k = 1, ..., K$. (master)
 5: Compute $\mathbf{A}_{qk}^t$ and $\mathbf{b}_{qk}^t$ for $k = 1, ..., K$. (slaves)
 6: Compute the global sums $\sum_{q=1}^Q \mathbf{A}_{qk}^t$ and $\sum_{q=1}^Q \mathbf{b}_{qk}^t$ by parallel reduction for the
    master process for $k = 1, ..., K$. (slaves)
 7: Compute $\mathbf{v}_k$ with Eq. 3 for $k = 1, ..., K$. (master)
 8: Compute $\mathbf{u}_k^{t+1}$ with Eq. 4 for $k = 1, ..., K$. (master)
 9: Set $t = t + 1$ and if $t < t_{max}$ and $\underset{k=1,...,K}{\text{median}} \|\mathbf{u}_k^t - \mathbf{u}_k^{t-1}\|_\infty > \epsilon_{tol}$, then repeat steps
    5-8. (master)
10: Set $\mathbf{m}_k = \mathbf{u}_k^t$ for $k = 1, ..., K$. (master)
11: Repeat steps 2-10 until convergence.
```

where $\omega \in [0, 2]$ determines the stepsize along the direction of $(\mathbf{v}_k - \mathbf{u}_k^t)$. For the consecutive SOR iterations t and $t+1$, the stopping criterion for the kth cluster is defined as $\|\mathbf{u}_k^{t+1} - \mathbf{u}_k^t\|_\infty \le \epsilon_{tol}$.

The proposed parallel method K-spatialmedians|| is described in Algorithm 1. The distribution is based on single program multiple data (SPMD) model. The approach assumes that $\mathbf{X}$ and $\mathbf{P}$ are approximately equally distributed to Q processing elements. The proposed method first applies modified K-means|| for the initialization (referred as K-spatialmedians$||^0$). The first modification to K-means|| is that we use the Euclidean distance instead of the squared Euclidean distance during the whole initialization procedure and we apply K-spatialmedians instead of K-means to cluster the sampled points with weights. The second modification deals with the MV handling, where, because we need to have complete prototypes after the initialization, K-spatialmedians$||^0$ is run only for the complete observations in $\mathbf{X}$. In the steps 4-9, the spatial medians are computed in parallel based on the SOR algorithm. The serial version of the SOR algorithm is depicted in [9]. Note that the parallellized SOR algorithm differs from the serial one in the stopping criterion. In the parallel version, the number of SOR iterations required for convergence is the same for each cluster, since the stopping criterion is based on the median of $\{\|\mathbf{u}_k^{t+1} - \mathbf{u}_k^t\|_\infty\}_{k=1}^K$.

3 Experiments and results

The accuracy of K-spatialmedians|| was compared with K-means|| for a synthetic dataset. The scalability properties of the parallel K-spatialmedians|| implementation were experimented with a large real dataset.

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

3.1 Experimental setup

All experiments were performed in the MATLAB R2017b environment. The scalability experiments were performed using a cluster equipped with eight Intel Xeon CPU E7-8837 with each having 128 GB memory and 8 cores. We implemented the parallel K-spatialmedians$\|$ with SPMD paradigm by utilizing MATLAB's parallel computing toolbox (PCT).

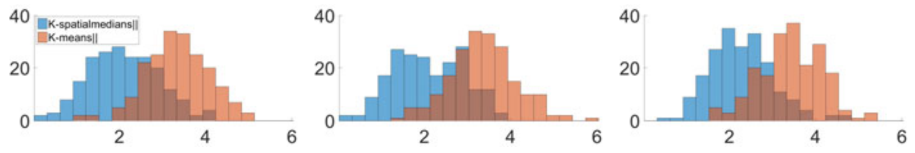
We realized the accuracy experiments with a synthetic S2[1] dataset. S2 is a two-dimensional dataset with 5000 observations. In order to assess robustness of K-spatialmedians$\|$, we disturbed original S2 with outliers and missing values. First, we replaced 250 observations with uniformly random observations, where both values were generated from two times larger range than the original S2. Then, we generated the MVs by randomly selecting elements from data and replacing them with MVs. Moreover, we ensured that we did not replace an observation's both elements with MVs.

For the scalability experiments we selected the Oxford buildings (OXB) dataset[2]. The experiments were run with additional dataset, which consists of 16,334,970 SIFT descriptors extracted from the original dataset with dimensionality 128. Moreover, this dataset was modified by replacing 10 percent of randomly chosen elements with MVs attached to $N/2$ randomly selected observations. The scalability related to the speedup was examined with a random 20 percent sample of OXB dataset with MVs. The scalability of with respect to the data size was tested with varying a random sample size from 20 to 100 percents.

All datasets were min-max scaled to the range $[-1, 1]$. For K-means$\|$, we run the initialization for the full rows of $\mathbf{X}$ and in the K-means search phase we used the available data strategy. For K-means$\|$ and K-spatialmedians$\|^0$, we set $l = 2 * K$ and $r = 5$, based on the experiments in [6]. For K-spatialmedians$\|$, we set $\epsilon_{tol} = 10^{-3}$, $\omega = 1.5$, and $t_{max} = 100$. For S2 we set the number of clusters to $K = 15$. For S2, the clustering iterations were ran until there were no new cluster assignments with respect to the previous iteration, and we repeated these runs 200 times. Since K-means$\|$ and K-spatialmedians$\|$ aim to minimize different cost functions, to get fair comparison, all the reported clustering errors were computed with respect to the ground truth prototypes. For each prototype, we computed Euclidean distance to the closest ground truth prototype and summed these distances. Furthermore, each of the ground truth prototypes and the prototypes archieved from the experiments contributes to the clustering error only once. In the experiments related to the speedup and the data size we set $K = 10$. To analyze the scalability with respect to the number of clusters, we varied K between 10 and 160, and this was conducted with 32 processing elements (MATLAB workers). We varied the number of processing elements between 1 and 32 to test the speedup. In the scalability experiments, clustering was performed once with 20 iterations for each setting.

[1]`http://cs.uef.fi/sipu/datasets/`

[2]`http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`



(a) 0% MVs (b) 10% MVs (c) 30% MVs

Fig. 1: Error distributions for S2 with varying level of MVs.

3.2 Clustering quality

K-spatialmedians|| and K-means|| error distributions for S2 with 0%, 10% and 30% MVs are shown in Figure 1. Clearly, K-spatialmedians|| finds better clustering results than K-means||. Based on visual inspection of the best resulting prototypes (selection based on Eq. 2), K-spatialmedians|| is able to find an optimal clustering result for 0% and 10% MVs. For 30% MVs, K-spatialmedians|| misplaces one prototype incorrectly. Similarly, based on visual inspection of the best resulting prorotypes (selection based on SSE with the available data strategy), K-means|| misplaces six prototypes for 0%, 10% and 30% MVs. These best resulting prototypes for S2 with 10% MVs are shown in Figure 2, where they are plotted in a frame of the original S2 data points with noise.

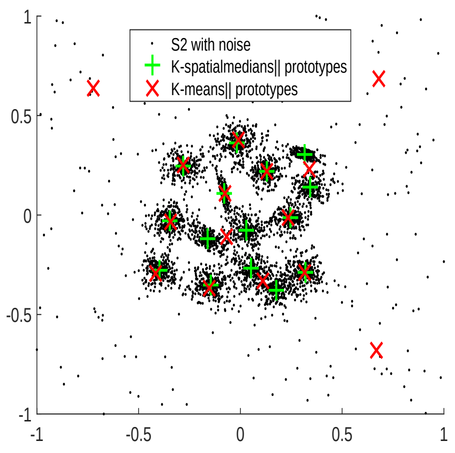
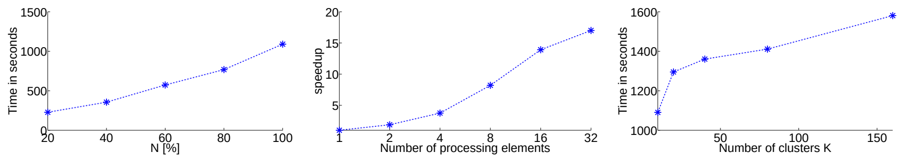


Fig. 2: The best final prototypes out of 200 runs for S2 with 10% MVs.

3.3 Scalability

The scalability results for K-spatialmedians|| are shown in Figure 3. The execution time increases linearly with respect to the data size, similarly as for the original K-spatialmedians. Moreover, we observed that time taken by the initialization is negligible with respect to total running time (about 1% of the total running time). As a function of the number of processing elements, the parallel implementation scales well. Speedup is nearly linear from 1 to 16 processing elements. As a function of the number of clusters, the execution time increases linearly after $K = 20$. The nonlinear behaviour in the beging of the curve is due to a moderate increase of SOR iterations. The total number of SOR iterations for $K = 10$ is 106, for $K = 20$ 130, and for $K = 40$ 127. Finally, we also assessed Gustafson's law with 32 processing elements, and we observed 60% of the theoretical speedup.



(a) Data size (b) Speedup (c) Number of clusters

Fig. 3: Scalability of K-spatialmedians|| for OXB dataset with 10% MVs.

4 Conclusions

In this paper, we proposed K-spatialmedians||, which is a parallel version of K-spatialmedians for large and sparse data. Moreover, K-spatialmedians|| utilizes an initialization strategy based on K-means||. Based on the experiments on the synthetic dataset with noise and missing values, K-spatialmedians|| outperforms K-means|| in terms of clustering quality. Based on the experiments, the proposed algorithm scales well with respect to the size of data, the speedup and the number of clusters. In the future work, we plan to study the proposal in more detail in terms of the initialization.

References

[1] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2013.

[2] Vahan Petrosyan and Alexandre Proutiere. Viral initialization for spectral clustering. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2017*, pages 293–298, 2017.

[3] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. *LargeScale Parallel Data Mining*, 1759(802):245–260, 1999.

[4] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 674–679, 2009.

[5] Reza Farivar, Daniel Rebolledo, Ellick Chan, and Roy H Campbell. A parallel implementation of k-means clustering on GPUs. In *Pdpta*, volume 13, pages 212–312, 2008.

[6] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

[7] David Arthur and Sergei 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.

[8] T Kärkkäinen and S Äyrämö. On computation of spatial median for robust data mining. *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, EUROGEN, Munich*, 2005.

[9] Sami Äyrämö. *Knowledge mining using robust clustering.* University of Jyväskylä, 2006.

[10] Sami Äyrämö and Tommi Kärkkäinen. Introduction to partitioning-based clustering methods with a robust example. *Reports of the Department of Mathematical Information Technology. Series C, Software engineering and computational intelligence 1/2006*, 2006.

[11] Joonas Hämäläinen, Susanne Jauhiainen, and Tommi Kärkkäinen. Comparison of internal clustering validation indices for prototype-based clustering. *Algorithms*, 10(3):105, 2017.