ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Initialization of Big Data Clustering using Distributionally Balanced Folding

Joonas Hämäläinen and Tommi Kärkkäinen

Department of Mathematical Information Technology
P.O. Box 35, 40014 University of Jyväskylä - Finland

Abstract. Use of distributionally balanced folding to speed up the initialization phase of *K-means++* clustering method, targeting for big data applications, is proposed and tested. The approach is first described and then experimented, by focusing on the effects of the sampling method when the number of folds created is varied. In the tests, quality of the final clustering results were assessed and scalability of a distributed implementation was demonstrated. The experiments support the viability of the proposed approach.

1 Introduction

Iterative relocation clustering algorithms are known to be sensitive to the initial placement of the prototypes. Actually the twofold aim of clustering, to divide data into groups where observations within a group are more similar to each other than observations in other groups [1], is approached in the well-known algorithms, most prominently in the classical *K-means++* [2, 1], by using the two main steps: *i*) initial location of K separate prototypes, *ii*) local refinement (search) of the initial prototypes to get the final solution. Due to variations of step *i*), it is known that this kind of algorithms do not guarantee unique clustering result or convergence to the global minimum of the clustering error (e.g., [1, 3, 4]).

The final clustering result can be improved by using some other than the random strategy for the initialization [5]. Chen et al. [6] argue that in the high-dimensional space data are inherently sparse. This is due to the well-known *curse of dimensionality* [7]. For example, the random samples tend to concentrate on the corners of a hypercube and the distance between each pair of observations becomes almost the same for a wide variety of data distributions. Chen et al. [6] conclude that, for small data sets, the method by Bradley and Fayyad [8], where the original data set is first splitted into smaller subsets which themselves are clustered, yielded to the best clustering results. In general, one agrees that the initial prototypes should be as far from each other as possible (without being outliers) [9, 1]. Lately, the K-means++ algorithm [10], where the random initialization is based on a density function favoring distinct prototypes, has become the most popular variant to initialize the K-means-type of algorithms.

Sampling is the basic approach to reduce the number of observations in statistics. It has a natural role in big data applications to cope with data volume, although one of the characteristics of big data [11] is precisely the lack of stable distribution, especially with high veracity and velocity. But the way sampling is done matters, and this issue has mostly been considered in relation to the well-known cross-validation, CV (see [12, 13] and articles therein). More precisely, the *Distribution Optimally Balanced Stratified CV, DOB-SCV*, has been evaluated as an appropriate sampling method for large pool of datasets to ensure good approximation of data density in the distinct folds created.

An approach where folding is applied with the K-means++ address the special nature of a high-dimensional data by separating the observations more clearly. This is combined with the good initialization strategies as mentioned above: splitting of data and the distinct initial locations. According to our knowledge, such techniques and in particular the DOB-SCV algorithm, has not been previously suggested or tested in the big data clustering context. This is the main goal of the paper, whose structure is as follows: In Section 2, we describe the clustering approach and in Section 3 provide results from computational experiments. The paper is shortly concluded in Section 4.

2 The Method

Let $\mathbf{X}$ be the given set of observations. In the initialization of the K-means++ algorithm [10], one prototype is first selected at random from $\mathbf{X}$. Then the rest $K-1$ prototypes are selected from $\mathbf{X}$ with probability $d(\mathbf{x})^2 / \sum_{\mathbf{x}\in\mathbf{X}} d(\mathbf{x})^2$, where $d(\mathbf{x})$ is the smallest distance to a prototype that was already selected. Hence, with high probability, one obtains a set of clearly distinct initial prototypes.

The DOB-SCV, as proposed in [12] (see [13] for the actual algorithm), was targeted to create folds (disjoint subsets of data) for the cross-validation. Stratification means that the folds are created classwise, or approximating the whole data distribution if no labelling is available.

The basic strategy to create k folds is, interestingly, opposite to clustering: select a random observation from class j, add it to the first fold and then add its $k-1$ nearest class neighbors to different folds without replacement. This process is then repeated until all the observations within class j are assigned to folds. These steps are applied to all classes. As a result, in addition to the classical stratification of approximating the class sizes, also class densities are preserved in the folds. Our approach here is to use this approach to speed up the initial search of distant prototypes in the K-means++ algorithm. From preliminary tests that we have made with the available K-means implementations on the popular Hadoop platform, we have noticed that the nondistributed initialization, especially for the K-means++ implementations, takes most of the computing time.

The proposed method is formalized in Algorithm 1. We let DOB-SCV to form k distjoint folds $\{\mathbf{X}_i\}_{i=1}^k$ from the data $\mathbf{X}$. The initial selection of distinct prototypes in K-means++ is then done in the folds where the best initial solution by means of the overall clustering (least-squares) error is selected to initialize the actual search (cf. [8]). Steps 1-2 in Algorithm 1 can be easily parallelized by distributing folds to workers and using Single Program Multiple Data (SPMD) model so that each worker processes its local fold. Communication between workers can be done with Message Passing Interface (MPI). Same folds and workers can also be applied in the local refinement step of the K-means algorithm in parallel using SPMD model and MPI, see [14].

3 Experimental results

Our first goal of the experiments is to compare how the initialization method affects to the clustering results in comparison to the whole data K-means++ initialization. Secondly, we test the scalability of the proposed approach when the number of folds is varied, by using SPMD implementation of Algorithm 1. As reference, we used the implementation [15] for K-means++. Matlab Distributed Computing Toolbox SPMD and Message passing functions were used for the parallel implementation. All the tests were performed in Matlab 8.3.0 (R2014a) environment. For the scalability tests, we used the Taito supercluster at IT Center for Science in Finland. Cluster resources were utilized via Matlab Distributed Computing Server (MDCS). The tests were run in the parallel partition with Sandy Bridge nodes having two eight-core Intel E5-2670 2.6 GHz processors with 256 GB RAM (16 GB per core).

Quality experiments were run with 5 data sets: S-sets [16] and USPS database [17]. S-sets consist of four real-valued 2-dimensional synthetic data sets generated from 15 gaussian distributions with known centers. Each S-set consists of 5,000 vectors. The overlappings of clusters increase from S_1 to S_4. The USPS is a well-known benchmark of handwritten digit dataset with 9,298 16×16 images and labels provided. This set consists of 10 classes. The number of clusters K was fixed for S-sets to 15 and for USPS dataset to 10. For all datasets, the variables were min-max scaled into $[-1, 1]$.

We ran Algorithm 1 with varying the number of the folds k between $2-50$ for the even numbers. For each k, K-means++ run was repeated k times and the minimum *sum-of-squared-errors*, SSE, was used to select as the final, reference clustering result. Tests were repeated 10 times for the both methods with DOB-SCV refolding in Algorithm 1. Comparison of the clustering results for K-means++ and Algorithm 1 was performed by analysing SSE and the pairwise prototype distances. For the latter measure, we calculated the sum of the smallest pairwise Euclidean distances, in such a way that each prototype was linked to the corresponding prototype only once. These were then averaged

Algorithm 1 K-means++ with folding initialization

```
Input: Folds $\{\mathbf{X}_i\}_{i=1}^k$ from DOB-SCV, number of clusters $K$.
Output: Set of prototypes $\mathbf{C} = \{\mathbf{c}_j\}_{j=1}^K$.
1: For each fold $\mathbf{X}_i$ do K-means++ clustering initialization.
2: For each set of prototypes $\mathbf{C}_i$ from a fold, calculate clustering error for the
   whole data.
3: Select the set of prototypes $\mathbf{C}_m$ with the smallest clustering error.
4: Do the K-means search starting from $\mathbf{C}_m$.
```

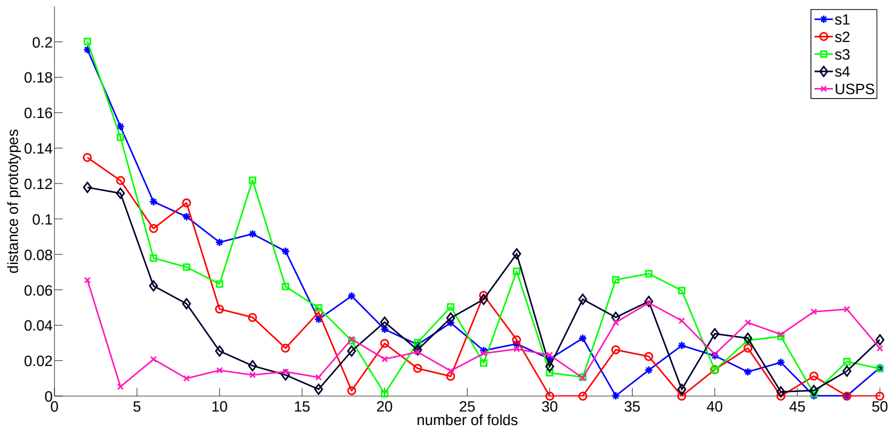

Fig. 1: Normalized prototype distances between Algorithm 1 and K-means++.

over 10 testruns for each k and normalized by dividing the result with the sum of the Euclidean norms of ground truth prototypes. For the USPS dataset, the ground truth prototypes were calculated as the class means.

In Fig. 1, the behavior of the normalized prototype distances are depicted. In general, all the results are very close to each other. We also observe that the prototypes become more similar between the methods when k increases. This might be due to the increase of the accuracy for both methods, more precisely, the number of i) the initial prototypes for Algorithm 1 and ii) the final prototypes for K-means++.

In Fig. 2, the SSE difference is the average SSE for the K-means++ which is subtracted from the average SSE for Algorithm 1. The SSE differences were again normalized, this time with the SSE for the ground truth prototypes. Al-

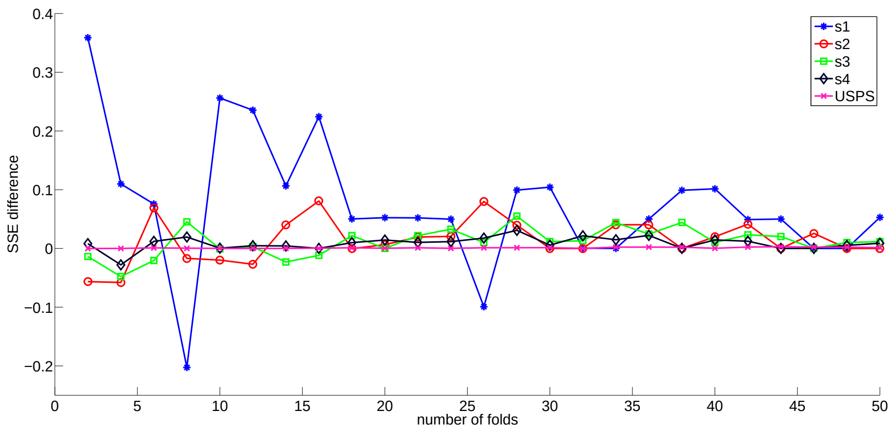

Fig. 2: Normalized SSE difference between Algorithm 1 and K-means++.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

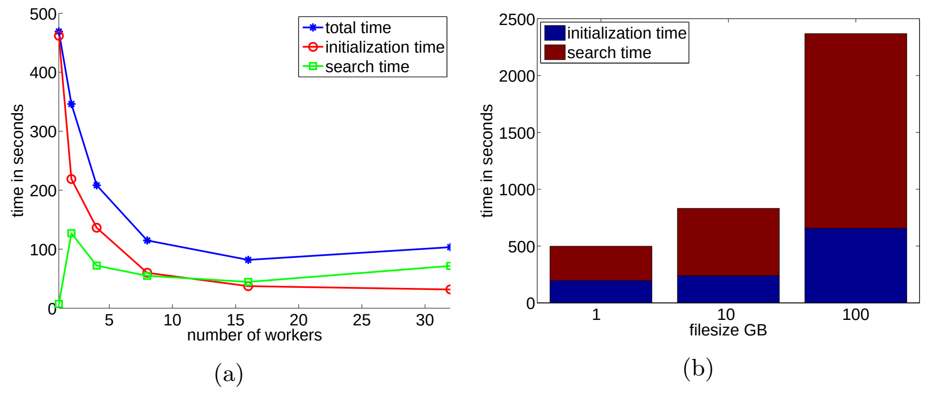

Fig. 3: Initialization and search phases wall times for parallellized Algorithm 1.

gorithm 1 occasionally gives smaller errors than the repeated, full K-means++, especially for the smaller values of k. A strong variation of the SSE difference for the dataset S_1 is most likely a consequence of higher probability to get stuck in a local minimum. For USPS, the SSE difference is close to zero for all the values of k, which indicates that the accuracy of Algorithm 1 is improved when the volume of the problem is increased. This is desirable in big data applications.

For the scalability tests with the implementation as described above, the MNIST database [18] was used. MNIST is another classical benchmark with handwritten digits that consists of 70,000 28×28 images with labels for 10 classes. In the first experiment, the number of workers were varied as 1, 2, 4, 8, 16 and 32. We also increased the data volume to demonstrate the behaviour: each data fold was copied 3 times for both observation and variable direction. Hence, the total data size was approximately 210,000×2,352. The wall clock time in Algorithm 1 for the initialization (Steps 1-3) and for the search phase (Step 4) was measured, again repeating the folding 10 times and averaging the wall clock times. Note that total time for the original (full data) K-means++ is given with one worker in Algorithm 1. As we can see from Fig. 3a, initialization time was reduced rapidly up to 8 workers until the time in communication starts to dominate, especially for the search phase.

In the second experiment, the number of workers was fixed to 100 and data size was varied approximately as, again using MNIST for copying, 210,000×2,352 (1 GB), 630,000×7,056 (10 GB) and 2,030,000×22,736 (100 GB). Algorithm 1 was ran once for each data and wall times were measured similarly as in the first experiment. Results are shown in Fig. 3b, which demonstrates that proposed method, indeed, scales for big data.

4 Conclusions

We proposed and tested distributionally optimal folding as an initialization method for the K-means++ clustering algorithm. The proposed initialization method can be easily parallellized to speedup the initialization phase. Based on the experiments, SSE can be occasionally even smaller for the proposed method compared to the k times repeated full K-means++. Overall, especially for larger values of k, the method provides very similar results compared to the full data approach. In our future work, targeting at big data applications, integration of dimension reduction to the initialization and the search phase is to be studied to reduce the data volume in iterative relocation algorithms even further.

References

[1] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[2] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[3] M. Emre Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 2012.

[4] M. Saarela and T. Kärkkäinen. Analyzing student performance using sparse data of core bachelor courses. *Journal of Educational Data Mining*, 7(1):3–32, 2015.

[5] L. Bai, J. Liang, and C. Dang. An initialization method to simultaneously find initial cluster centers and the number of clusters for clustering categorical data. *Knowledge-Based Systems*, 24(6):785–795, 2011.

[6] L. Chen, L. Chen, Q. Jiang, B. Wang, and L. Shi. An initialization method for clustering high-dimensional data. In *Database Technology and Applications, 2009 First International Workshop on*, pages 444–447. IEEE, 2009.

[7] M. Verleysen and D. François. The Curse of Dimensionality in Data Mining. *Analysis*, 3512:758 – 770, 2005.

[8] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99, 1998.

[9] S. S. Khan and A. Ahmad. Cluster center initialization algorithm for k-modes clustering. *Expert Systems with Applications*, 2013.

[10] D. Arthur and S. 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.

[11] B. Hammer, H. He, and T. Martinetz. Learning and modeling big data. *22th European Symposium on Artificial Neural Networks (ESANN2014)*, (April):23–25, 2014.

[12] J. G. Moreno-Torres, J. A. Sáez, and F. Herrera. Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1304–1312, 2012.

[13] T. Kärkkäinen. On cross-validation for MLP model evaluation. In *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science (8621), pages 291–300. Springer-Verlag, 2014.

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

[15] L. Sorber. k-means++ - File Exchange - MATLAB Central, 2010.

[16] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–775, 2006.

[17] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, May 1994.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.