Distributed SNN Clustering for large document collections

October 28, 2016

Abstract

Your abstract.

1 Introduction

As a consequence of the explosive growth of the WEB, the integration of search engines such as Google into personal computers and mobile devices, and the wide use of social networks, the task of clustering text data, i.e. Tweets or documents in a search engine, has each time an increasing importance because of the necessity of unraveling the underlying categories in large volumes of text data. Nowadays the generation of large amounts of documents surpasses the computational power of personal computers and even of high performance computers. As an example, it is estimated that the amount of WEB pages that popular search engines such as Yahoo! and Google index is higher than the tens of a billion [WWWSize, 2016]. It is therefore of great interest to develop algorithmic techniques capable of automatically organize, classify and summarize document collections distributed in multiple machines of a network, and that also perform efficiently in modern hardware, particularly within parallel processing frameworks in multi-core architectures. In real problems such as *Collection Selection* for distributed document databases, in which for a given query the computer node containing the most suitable sub-collection must be selected to answer it [Crestani and Markov, 2013], the challenges related to the scalability and efficiency of *Knowledge Discovery* methods have become very important. Traditional algorithms often assume that the whole dataset is loaded into main memory (RAM) and thus every document can be accessed at any time with no access latency. In scenarios where the size of the collection is much bigger than the amount of available RAM either because of the number of documents or the length of each document, this ideal loading process is unfeasible due to real constraints in the storage or computational capabilities.

Often, text data is structured in digital collections of documents whose length (number of characters) is variable, e.g. WEB pages or the content generated by users in social networks such as Twitter or Facebook. In order to enable the processing of these collections, in a first stage of preprocessing a set of words occurring in the documents are extracted and sorted in lexicographical order; this word set is referred to as the Vocabulary. Then, the content of every document in the collection is represented as a vector in which each dimension denotes a specific word of the Vocabulary and its value in a document vector is given by a function of the number of occurrences of the word within the document and the number of documents in which it appears. As a natural consequence of the lexical richness of every language, the size of the Vocabulary, and in turn the dimensionality of the vector space onto which a document is represented, is far bigger than the data size that traditional clustering algorithms manage. Because of this, the task of automatic document clustering has high computational and storage (RAM and secondary memory) costs. Along with this, when the number of documents is large (tens or hundreds of thousand and even millions of items) then traditional techniques for processing and clustering documents, and current computational capabilities of a single machine and also high performance machines are insufficient. Even in the most favorable scenario the storage and computational power tackle the challenge but the response time are excessive.

There exist three approaches successfully applied to the construction of clustering algorithms capable of processing large volumes of data. The first one introduces constraints on the number of passes allowed on a document (related to the amount of time it is loaded into main memory)[CITA]. The second one exploits current multi-core architectures to perform parallel processing of the data, which does not solve the massive volume problem [CITA]. The last one combines the computational power of a single machine together with the scalable storage capability of a distributed system by partitioning the dataset into several independent machines connected through a network[CITA].

The last above mentioned approach seems promising since it allows to exploit the local capabilities of single computers with multi-core architectures without sacrificing scalability to large data volumes because of its distributed design. Within this path there are two contexts regarding the data generation scenario. On the one hand, in some problems where the dataset is large but collected in a centralized fashion, the strategy employed consists in partitioning the collection into several machines or nodes of a network. This scheme leads to the transmission of a lot of data during the execution of the algorithm [Nagwani, 2015]. On the other hand, there are some problems where the data is generated in a distributed fashion and where besides it is not feasible to centralize the data because of high transmission costs or privacy issues [Jagannathan et al., 2005, Liu et al., 2012], e.g. search engines work with document collections originated and stored in different geographical locations.

This document is structured as follows: First, a review of the literature on scalable and distributed data clustering methods is presented. Next, the proposed method is shown. Finally,the experimental design along with the attained results and the final remarks are presented.

2 Distributed Clustering Algorithms

As far as we know from the literature, most of the existing efforts for the construction of clustering techniques capable of operating in scenarios where the data is distributed have been focused on low dimensional data (less than 100 attributes) in contrast with document datasets in which a document vector for a small collection may have about 10^4 attributes. Nevertheless, the main advances in distributed data clustering are detailed below, specially highlighting those contributions focused on methods capable of dealing with high dimensional data.

2.1 Main contributions on parallel algorithms

[Xu et al., 2002] and [Dhillon and Modha, 1999] propose parallel extensions for the algorithms DBSCAN and K-Means. Another scalable approach based on secondary memory consists in designing algorithms capable of working within the MapReduce framework[1]. In this context it is possible to highlight the contributions made by [Das et al., 2007] in which they propose an implementation of the EM algorithm and also by [Ene et al., 2011] in which they tackle the K-Median problem by using MapReduce. Another parallel approach for K-Means is presented by [Bahmani et al., 2012] and it is called K-Means++. Additionally, the EM-Tree proposed by [De Vries et al., 2015] is very interesting since it allows to process very large datasets, specifically the authors show that it can handle hundred of millions of WEB pages.

2.2 Approaches capable of dealing with high dimensional data

[Kargupta et al., 2001] propose a method to obtain the Principal Components (PCA) over heterogeneous and distributed data. Based on this contribution on dimensionality reduction they also propose a clustering method that works over high dimensional data. Once the global principal components are obtained by using the distributed method and transmitted to each node, local data are projected onto the components and then a traditional clustering technique is applied. Finally, a central node integrates the local clusters in order to obtain a global clustering model.

Several years later, [Liang et al., 2013] present another algorithm for principal components extraction over distributed data. To this end, each node computes PCA over its local data and transmit a fraction of them to a central node. This node uses the received components to estimate the global principal components, which are later transmitted to each node. After this, in every node, local data are projected onto the global components and the projected data are used for computing a coreset by means of a distributed algorithm. The global coreset built from the local projected data will be finally used to obtain a global clustering model.

[Li et al., 2003] propose an algorithm called D-CoFD for high dimensional and distributed data that also is capable of dealing with homogeneous and heterogeneous environments.

2.3 Density based approaches

[Januzaj et al., 2003] propose a distributed data clustering technique in which local nodes build models and transmit a set of representatives of each cluster to a central node. In this node a centralized clustering method is applied

[1]Hadoop MapReduce is a software solution that enables the construction of applications capable of processing large amounts of data (e.g. Terabytes) in a parallel fashion over big computer clusters.

over the representatives and then the resulting model is re-transmitted to the local nodes to update their models.

[Klusch et al., 2003] propose a clustering technique based on density estimates.

[Januzaj et al., 2004] present a scalable version of DBSCAN that is also capable of operating over distributed collections. First, the best local representatives a selected depending on the number of points that each one represents and then those chosen points are sent to a central node. This central node clusters the received local representatives into a single new model, which is re-transmitted to the other nodes so they can improve their local group structure.

2.4 Approaches based on parametric models

[Merugu and Ghosh, 2003] propose a clustering method that combines local parametric models, each one built on a single node, into a general one. The main contribution of this work consists in a method capable of dealing with distributed data that also considers the privacy of the local data in each node, since it transmits a summary of each local data and not raw points.

[Kriegel et al., 2005] present a technique that fits a Gaussian mixture model in each node using the EM algorithm. Finally, all these Gaussian mixture models are integrated into a general parametric model.

2.5 Approaches based on representative points

[Forman and Zhang, 2000] extend centroid based techniques to identify groups over distributed data. Specifically, they extend K-Means, K-Harmonic-Means and EM.

[Qi et al., 2008] propose an approximate K-Median clustering technique that works over stresaming data, i.e. data is continuously collected.

[Balcan et al., 2013] address the distributed clustering problem by using centroid based techniques that use a novel coreset construction method that works for distributed data.

[Naldi and Campello, 2014] tackle the problem of the parameter selection of the K-Means clustering algorithm by using evolutionary algorithms. Additionally, they propose two strategies inspired in evolutionary algorithms for clustering distributed data using centroids.

2.6 Hierarchical clustering approaches

A hierarchical algorithm that works on distributed and heterogeneous data is presented by [Johnson and Kargupta, 2000]. This method assumes that data is vertically partitioned, thus all nodes contain the same set of points, but characterized by different features. At the beginning, a dendrogram is generated in each node, and then it is transmitted to a central node. Finally, this node combines all the dendrograms into a global model.

[Jin et al., 2015] propose a hierarchical clustering algorithm for distributed data that builds the clusters by incrementally processing the data points. The authors re-state the hierarchical clustering problem as a Minimum Spanning Tree construction problem over a graph. In order to integrate several local models they also propose a technique for combining multiple minimum spanning trees, assuming that these trees were obtained from disjoint subgraphs of the complete original graph. This mixture procedure iterates until a single tree is obtained, which in turn denotes the hierarchical clustering originally pursued.

3 Proposal

In this work we present a distributed clustering algorithm based on *Shared-Nearest-Neighbor* (SNN) clustering. This method automatically identifies the number of underlying groups along with a set of representative points for each one. We pose that this method is able to deal with collections arbitrarily distributed across a Master/Worker architecture as depicted in figure 1 and the overall algorithm operates in two stages: The first one starts when the data is randomly partitioned and distributed into several nodes, then each one generates a set of representative points per cluster, i.e. *core-points*, and finally, transmits back a sample set of these *core-points* to the master node. In the second stage, the central node joins the sample points received and then starts a centralized SNN clustering over them. Finally, the set of representative points is labeled and also a new set of *core-points* that summarizes the overall collection is obtained.

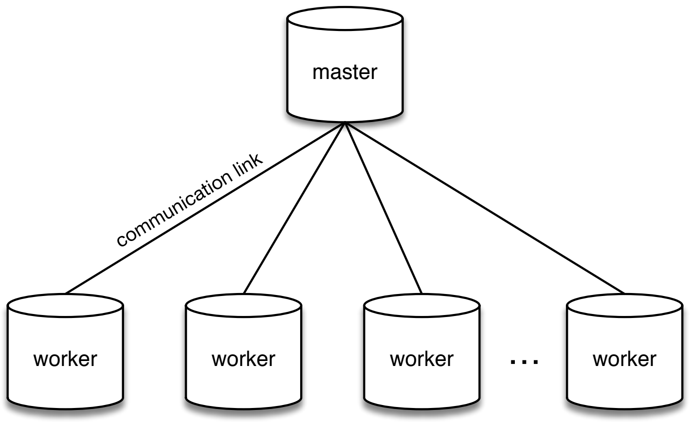

Figure 1: Network architecture employed by the proposed algorithm

Parameters of the algorithm

The proposed method comprehends three parameters, namely k, Eps, MinPts and ω. The value of parameter k determines the size of the neighborhood over which the SNN similarity measure is going to be computed. In the SNN space, the value of Eps denotes the similarity threshold beyond which two points are considered as close. Additionally, a point is identified as a *core-point* when the number of points close to it in the SNN space surpasses the value of parameter MinPts. Finally, the value of the parameter ω rules the size of the sample set of *core-points* that is going to be transmitted.

Initial stage

The method starts by randomly partitioning and distributing the dataset into several worker nodes. In this primary stage, after a worker node n_i receives its data chunk $\mathcal{D}_i$, it starts to identify the core-points by following algorithm $\mathsf{SnnCorePoints}$ described in algorithm 1. Once its core-point set $\mathcal{C}_i$ is built, an attempt to label each point $p \in \mathcal{C}_i$ is performed. That is, when the similarity value between p and its closest core-point q is higher than the value of parameter Eps, then the same label of q is assigned to p. Otherwise, p is tagged by a new label, i.e. it represents a new cluster. Regarding the labeled core-points, a weighted sample from each group is drawn. The weight of a point follows the expression:

$$\frac{1-(N_l/N_c)}{2 \cdot N_l}$$

where N_c denotes the number of labeled core-points and N_l the number of points labeled with label l. The expression shown above denotes that the sampling weight of a core-point is inverse to its group size as it is depicted in figure 2. The overall procedure is described in algorithm 2.

Final stage

See algorithm 3

Algorithm 1: Identification of core-points from a dataset

```
Function SnnCorePoints ($\mathcal{D}$, $k$, Eps, MinPts) :
    foreach $p \in \mathcal{D}$ do
        compute $KNN_k(p)$
    foreach $p \in \mathcal{D}$ do
        foreach $q \neq p \in \mathcal{D}$ do
            $\mathsf{SNN}_k(p,q) \leftarrow \#(KNN_k(p) \cap KNN_k(q))$
        $\mathsf{density}(p) \leftarrow \#\{q \neq p \in \mathcal{D} | \mathsf{SNN}_k(p,q) > \mathsf{Eps}\}$
        if $\mathsf{density}(p) > \mathsf{MinPts}$ then Mark $p$ as core-point
    $\mathcal{C} \leftarrow$ all points marked as core-point
    return $\mathcal{C}$ , $\mathsf{SNN}_k$
```

Algorithm 2: Selection of representative points executed in a node

```
Function SampleLocalData ($\mathcal{D}$, $k$, Eps, MinPts, $\omega \in [0,1]$) :
    $\mathcal{C}, \mathsf{SNN}_k \leftarrow$ SnnCorePoints ($\mathcal{D}$, $k$, Eps, MinPts)
    foreach core-point $p \in \mathcal{C}$ do
        if $p$ is not visited then
            Assign a new cluster label $l_p$ to $p$
            mark $p$ as visited
            foreach Not visited $q \neq p \in \mathcal{C}$ do
                if $\mathsf{SNN}_k(p,q) > \mathsf{Eps}$ then
                    Assign label $l_p$ to $q$
                    mark $q$ as visited
    $N_c \leftarrow$Number of labeled core-points
    $S \leftarrow \emptyset$
    foreach cluster $l$ having size $N_l$ do
        $S \leftarrow$ Add $\omega \cdot N_l$ points sampled from group $l$ with weight $\frac{1-(N_l/N_c)}{2 \cdot N_l}$
    return sampled data $S$
```

Algorithm 3: Procedure executed by the master node to generate the list of representative points per group

```
Input: $\mathcal{D}$ denotes the local subset of the data, $k$, Eps, MinPts, $\omega \in [0, 1]$
Partition the dataset $\mathcal{D}$ into $M$ subsets $\{D_1, D_2, \ldots D_M\}$
On each node $n_i$ execute
    $S_i \leftarrow$ SampleLocalData($D_i$) /* remote procedure call on node $m_i$ */
$S \leftarrow [S_1, S_2, \ldots S_M]$
$\mathcal{C}, \mathsf{SNN}_k \leftarrow$ SnnCorePoints($\mathcal{S}$, $k$, Eps, MinPts)
foreach core-point $p \in \mathcal{C}$ do
    if $p$ is not visited then
        Assign a new cluster label $l_p$ to $p$
        mark $p$ as visited
        foreach Not visited $q \neq p \in \mathcal{C}$ do
            if $\mathsf{SNN}_k(p, q) >$ Eps then
                Assign label $l_p$ to $q$
                mark $q$ as visited
foreach $p \in S \setminus \mathcal{C}$ do
    $n_p \leftarrow \arg\max_{q \in \mathcal{C}} \mathsf{SNN}_k(p, q)$
    if $\mathsf{SNN}_k(p, n_p) <$ Eps then Mark $p$ as noise
    else Assign label $l_{n_q}$ to $p$
Output: $\{p \in S \mid \exists\, l_p\}$
```

4 Experimental design and results

Dataset	Description	instances	$\lvert\mathcal{V}\rvert$	classes	%NNZ
20NG	20-newsgroup data, m5 partition.	4743	41223	5	0.167%
DOE	Department of Energy Abstracts	1664	15755	14	0.365%
FR	Federal Register notes	926	50427	14	1.104%
SJMN	San Jose Mercury News	908	23616	16	0.738%
ZF	Computer select disks by Ziff-Davis	3263	58398	25	0.360%

Table 1: Datasets employed.

Dataset	**Entropy**	**Purity**	**ARI**	**AMI**	**NMI**	**HOM**	**COM**	**VM**
20NG	0.2948	0.8307	0.6050	0.6421	0.6521	0.6619	0.6425	0.6521
DOE	0.2721	0.7139	0.4919	0.7030	0.7276	0.7095	0.7461	0.7273
FR	0.2524	0.7559	0.6185	0.7266	0.7413	0.7375	0.7452	0.7413
SJMN	0.2412	0.7544	0.5953	0.7367	0.7580	0.7505	0.7657	0.7580
ZF	0.4166	0.6028	0.3817	0.5444	0.5800	0.5593	0.6015	0.5796

Table 2: Performance attained by the graph clustering algorithm over the datasets using the number of classes as the value for the number of clusters.

5 Conclusions and future work

6 Acknowledgment

Juan Zamora was supported by an intern postdoc project from Pontificia Universidad Católica de Valparaíso.

Dataset	Entropy	Purity	ARI	AMI	NMI	HOM	COM	VM
20NG	0.3525	0.6036	0.3432	0.3953	0.4373	0.3990	0.4793	0.4355
DOE	0.2865	0.6911	0.4197	0.6370	0.6592	0.6476	0.6711	0.6591
FR	0.1921	0.8531	0.7099	0.7834	0.7944	0.7969	0.7919	0.7944
SJMN	0.2024	0.8282	0.5846	0.6820	0.7336	0.7732	0.6960	0.7326
ZF	0.3330	0.6089	0.2845	0.5084	0.5488	0.5750	0.5238	0.5482

Table 3: Performance attained by the SNN clustering algorithm executed in a centralized fashion over the datasets using the number of classes as the value for the number of clusters.

Dataset	K	Eps	MinPts
20NG	110	25	30
DOE	70	25	30
FR	50	25	20
SJMN	50	20	30
ZF	90	40	25

Table 4: Parameters of the centralized SNN-Clustering algorithm selected after the tuning procedure.

Dataset	Entropy	Purity	ARI	AMI	NMI	HOM	COM	VM
20NG	0.1710	0.8828	0.8400	0.8218	0.8702	0.8262	0.9167	0.8691
DOE	0.1074	0.9318	0.6816	0.7029	0.7972	0.8794	0.7227	0.7934
FR	0.0949	0.9134	0.7224	0.7546	0.8346	0.8947	0.7784	0.8325
SJMN	0.1802	0.7656	0.6360	0.7836	0.8046	0.8052	0.8040	0.8046
ZF	0.0117	0.9943	0.7809	0.7701	0.8823	0.9882	0.7877	0.8766

Table 5: Best performance attained by the distributed SNN-Clustering algorithm over the datasets.

Dataset	K	Eps	MinPts
20NG	30	8	25
DOE	30	10	20
FR	30	10	5
SJMN	30	10	10
ZF	30	10	15

Table 6: Parameters of the distributed SNN-Clustering algorithm selected after the tuning procedure.

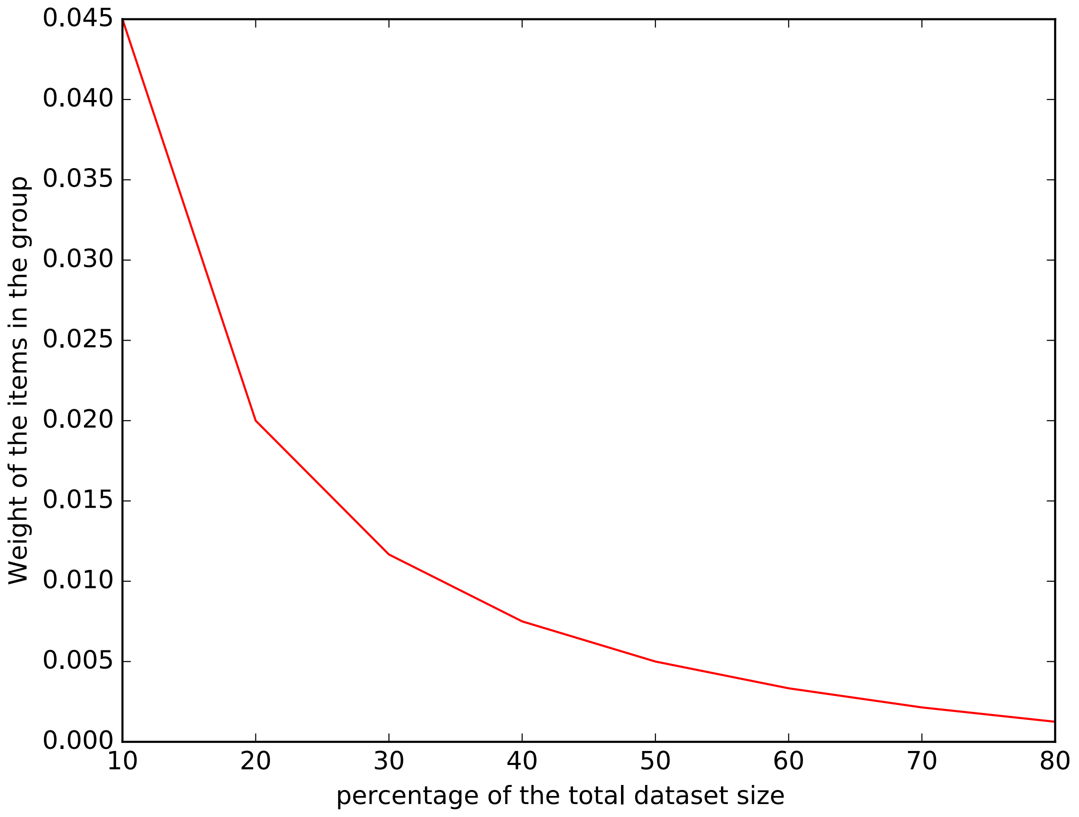

Figure 2: Weight of each item in a group vs the relative size of the group in comparison with the total dataset size.

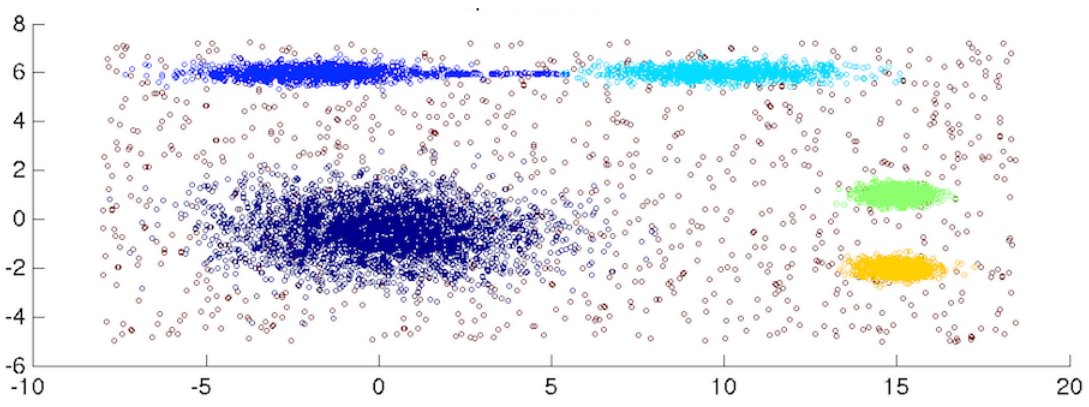

Figure 3: Toy example: Original data.

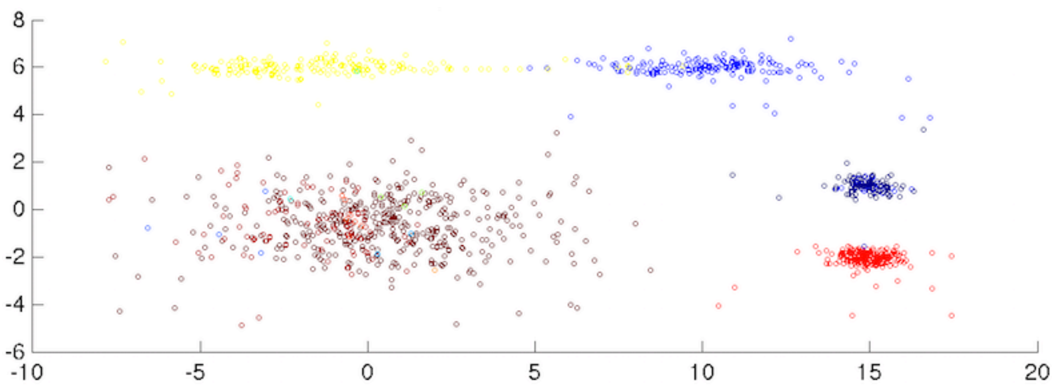

Figure 4: Toy example: Data after distributed snn clustering with $Eps = 50$.

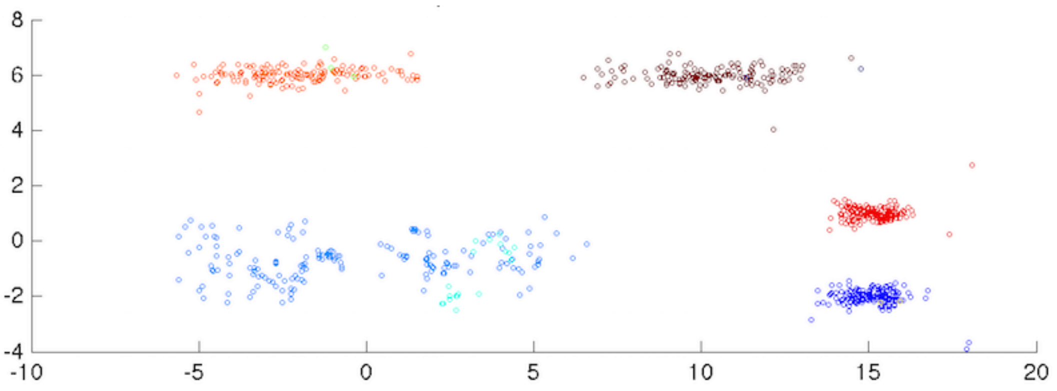

Figure 5: Toy example: Data after distributed snn clustering with $Eps = 60$.

References

[Bahmani et al., 2012] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. (2012). Scalable K-Means ++. *Proceedings of the VLDB Endowment (PVLDB)*, 5:622–633.

[Balcan et al., 2013] Balcan, M. F., Ehrlich, S., and Liang, Y. (2013). Distributed k -Means and k -Median Clustering on General Topologies. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 1–9.

[Crestani and Markov, 2013] Crestani, F., and Markov, I. (2013). Distributed Information Retrieval and Applications. *35th European Conference on IR Research*, 865–868.

[Das et al., 2007] Das, A., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM.

[De Vries et al., 2015] De Vries, C. M., De Vine, L., Geva, S., and Nayak, R. (2015). Parallel streaming signature em-tree: A clustering algorithm for web scale applications. In *Proceedings of the 24th International Conference on World Wide Web*, pages 216–226. ACM.

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

[Ene et al., 2011] Ene, A., Im, S., and Moseley, B. (2011). Fast Clustering using MapReduce. *Kdd*, 681–689.

[Ertöz et al., 2003] Ertöz, L., Steinbach, M., and Kumar, V. (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. *Proceedings of the SIAM International Conference on Data Mining*, 47–58.

[Forman and Zhang, 2000] Forman, G. and Zhang, B. (2000). Distributed data clustering can be efficient and exact. *ACM SIGKDD Explorations Newsletter*, 2(2):34–38.

[Han et al., 2011] Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[Jagannathan et al., 2005] Jagannathan, G., and Wright, R. N. (2005). Privacy-preserving Distributed K-means Clustering over Arbitrarily Partitioned Data. *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 593–599.

[Januzaj et al., 2003] Januzaj, E., Kriegel, H.-P., and Pfeifle, M. (2003). Towards Effective and Efficient Distributed Clustering. *Workshop on Clustering Large Data Sets*, pages 49–58.

[Januzaj et al., 2004] Januzaj, E., Kriegel, H.-P., and Pfeifle, M. (2004). Scalable Density-Based Distributed Clustering. pages 231–244.

[Jin et al., 2015] Jin, C., Chen, Z., Hendrix, W., Agrawal, A., and Choudhary, A. (2015). Incremental, Distributed Single-linkage Hierarchical Clustering Algorithm Using Mapreduce. *Proceedings of the Symposium on High Performance Computing*, pages 83–92.

[Johnson and Kargupta, 2000] Johnson, E. and Kargupta, H. (2000). Collective, hierarchical clustering from distributed, heterogeneous data. *Lecture Notes in Computer Science*, 1759:221–244.

[Kargupta et al., 2001] Kargupta, H., Huang, W., Sivakumar, K., and Johnson, E. (2001). Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448.

[Klusch et al., 2003] Klusch, M., Lodi, S., and Moro, G. (2003). Distributed clustering based on sampling local density estimates. *IJCAI International Joint Conference on Artificial Intelligence*, pages 485–490.

[Kriegel et al., 2005] Kriegel, H.-p., Kr, P., Pryakhin, A., and Schubert, M. (2005). Effective and Efficient Distributed Model-based Clustering.

[Li et al., 2003] Li, T., Zhu, S., and Ogihara, M. (2003). Algorithms for Clustering High Dimensional and Distributed Data. *Intelligent Data Analysis Journal*, 7(February):1–36.

[Liang et al., 2013] Liang, Y., Balcan, M.-f., and Kanchanapally, V. (2013). Distributed PCA and k-Means Clustering. *The Big Learning Workshop in NIPS 2013*, pages 1–8.

[Liu et al., 2012] Liu, J., Huang, J. Z., Luo, J., and Xiong, L. (2012). Privacy Preserving Distributed DBSCAN Clustering. *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, 177–185.

[Merugu and Ghosh, 2003] Merugu, S. and Ghosh, J. (2003). Privacy-preserving Distributed Clustering using Generative Models. *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, pages 0–7.

[Nagwani, 2015] Nagwani, N. K. (2015). Summarizing large text collection using topic modeling and clustering based on MapReduce framework. *Journal of Big Data*, 2:1–18.

[Naldi and Campello, 2014] Naldi, M. C. and Campello, R. J. G. B. (2014). Evolutionary k-means for distributed data sets. *Neurocomputing*, 127:30–42.

[Qi et al., 2008] Qi, Z., Jinze, L., and Wei, W. (2008). Approximate clustering on distributed data streams. *Proceedings - International Conference on Data Engineering*, 00:1131–1139.

[Rousseeuw, 1987] Rousseeuw Peter J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

[Xu et al., 2002] Xu, X., Jäger, J., and Kriegel, H. (2002). A fast parallel clustering algorithm for large spatial databases. *High Performance Data Mining*, 290:263–290.

[WWWSize, 2016] http://www.worldwidewebsize.com/ Accessed at 26th September 2016