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/.

PSCEG: An unbiased Parallel Subspace Clustering algorithm using Exact Grids

Bo Zhu, Bruno Ordozgoiti, and Alberto Mozo *

Universidad Politécnica de Madrid - Departamento de Sistemas Informáticos
bozhumatias@ict-ontic.eu, bruno.ordozgoiti@etsisi.upm.es, a.mozo@upm.es

Abstract. The quality of grid-based subspace clustering is highly dependent on the grid size and the positions of dense units, and many existing methods use sensitive global density thresholds that are difficult to set a priori. We propose PSCEG, a new approach that generates an exact grid without the need to specify its size based on the distribution of each dimension. In addition, we define an adaptive density estimator that avoids dimensionality bias. A parallel implementation of our algorithm using Resilient Distributed Datasets achieves a significant speedup w.r.t. the number of cores in high dimensional scenarios. Experimental results on synthetic and real datasets show PSCEG outperforms existing alternatives.

1 Introduction and Related work

Clustering techniques group similar objects into entities called clusters. However, clusters can hide in different subspaces of the full feature space. Subspace clustering algorithms try to find all low-dimensional clusters hidden in subspaces of high dimensional data. One of the main families of such algorithms, known as grid-based methods, partition the data space into non-overlapping cells by discretizing each dimension into small units. In many of these algorithms a global density threshold is used to determine whether a unit is dense. Candidate dense units of higher dimensionality are iteratively generated in a bottom-up fashion. Afterwards, adjacent dense cells in the same subspace are merged together to form clusters. CLIQUE [1] is recognized as the pioneering grid-based method for subspace clustering. It is the first to use the anti-monotonicity property to prune sparse candidates. Mafia [2] improves CLIQUE by generating an adaptive grid in order to have less candidates in higher dimensions. Parallelism is also introduced to strengthen scalability and efficiency. Another variant of CLIQUE is ENCLUS [3], which uses entropy to search for interesting subspaces that may contain clusters. Similar to ENCLUS, SCHISM [4] tries to find interesting subspaces using support and Chernoff-Hoeffding bounds. nCluster [5] further improves CLIQUE by partitioning each dimension into many overlapping small bins, and then selecting only maximal nClusters for higher dimensionality. Similar conclusions are provided in experimental evaluation works [6, 7, 8]: the efficiency and accuracy of existing grid-based subspace algorithms is highly dependent on the proper tuning of the granularity of the grid, the positions of dense units, and

*The research leading to these results has received funding from the European Union under the FP7 grant agreement n. 619633 (project ONTIC) and H2020 grant agreement n. 671625 (project CogNet) the density thresholds. It has also been observed that a global density threshold will cause a bias to a certain dimensionality [6, 9]. Dimensionality bias is first discussed in [9] referring to the fact that a tight global density threshold may distinguish clusters well from noise in low dimensions but lead to a loss of high dimensional clusters. However, a loose one can detect high-dimensional clusters as well as extensive untranslatable low dimensional clusters.

We propose a novel approach that generates an exact grid without the need to specify the size of the grid a priori. It can capture the positions of dense units and proper size of the grid based on the data distribution of each dimension. To avoid the effect of dimensionality bias, we conduct subspace clustering with different granularities by defining and using a novel unbiased density estimator that is adaptive to dimensionality. A new parallel subspace clustering algorithm called PSCEG is proposed and implemented using Resilient Distributed Datasets (RDDs) [10], the rapidly evolving new paradigm for Big Data processing, on top of the Spark distributed computing platform. Experimental results on synthetic and real datasets show that PSCEG has better scalability, accuracy and efficiency compared to existing alternatives. In addition, PSCEG achieves a significant speedup w.r.t. the number of cores in high dimensional scenarios.

2 PSCEG algorithm

Given a dataset of d dimensions and o objects, we use the following notation:

D: the set of all dimensions, i.e. $D = \{1, 2, 3, \ldots, d\}$; O: the set of all objects; V_m: the set of object values for the m-th dimension; $o_{mn} \in \mathbb{R}$: object value for the m-th dimension and the n-th object $(1 \leq m \leq |D|, 1 \leq n \leq |O|)$; CDU^k: the set of candidate dense units for dimensionality k; DU^k: the set of dense units for dimensionality k; B: set of equally sized intervals.

PSCEG solves the problem of subspace clustering by first generating a grid and then detecting subspace clusters of increasing dimensionality iteratively like other grid-based algorithms. The novelty of our algorithm is twofold: (1) It does not need an a priori choice of grid size to find the exact positions of dense units, and (2) it updates the adaptive density thresholds for the generation of higher-dimensional dense units. A definition of dense units is given in [7]. In order to avoid the dimensionality bias, we propose a more general definition considering variable density thresholds instead of the global one used in [7].

Definition 1 *Dense unit. A dense unit in subspace* $S \subset D$ *is a set of 1-dimensional clusters* $C = \{C_m | m \in S\}$ *such that* $\forall C_m \in C \; |\{o \in O | o_m \in C_m\}| > \tau_{C_m}$*, where* $\tau_{C_m} = \frac{\alpha(max(C_m) - min(C_m))|O|}{max(V_m) - min(V_m)}$.

C_m represents the usual notion of density-based cluster as defined in the paper of DBSCAN [11]. α is a parameter called cluster dominance factor [2] that estimates the density of the expected clusters. By updating the density thresholds for each dimensionality, our density estimator avoids a dimensionality bias.

The dataset is preprocessed so that the object values of all dimensions fall in a user-specified range $[V_1, V_2]$. Our algorithm consists of two phases: the generation of an exact grid without the need to specify the grid size a priori and the iterative process of finding subspace clusters using adaptively updated

Algorithm 1 PSCEG(dataRDD, α, θ)

Phase 1: GenerateExactGrid(dataRDD, α, θ)

```
1: $CDU^1 \leftarrow \emptyset \;\; DU^1 \leftarrow \emptyset$
2: for $m \in D$ do
3:     $B = \{[i\Delta + V_1, (i+1)\Delta + V_1) \mid 0 \le i < N, i \in \mathbb{N}\}$
4:     $\forall b \in B : \omega_b \leftarrow |\{o \in O | o_{mn} \in b\}|$
5:     $B \leftarrow \{b \in B | \omega_b \ge \frac{\alpha|O|}{N}\}$
6:     $CDU^1 \leftarrow CDU^1 \cup \text{DBSCAN}(B, \epsilon = \alpha\theta|V_m|, minPoints = \alpha\theta|O|,\ \{\omega_b | b \in B\})$
7:     for $cdu^1 \in CDU^1$ do
8:         $\forall C_m \in cdu^1 : \tau_{C_m}^{cdu^1} \leftarrow \frac{\alpha(max(C_m)-min(C_m))|O|}{max(V_m)-min(V_m)}$
9:     end for
10: end for
11: $\forall cdu^1 \in CDU^1 : w_{cdu^1} \leftarrow |\{o \in O | \forall C_m \in cdu^1\ o_{mn} \in C_m\}|$
12: $DU^1 \leftarrow \{cdu^1 \in CDU^1 | \forall C_m \in cdu^1 : w_{cdu^1} \ge \tau_{C_m}\}$
```

Phase 2: Grid-basedClustering(dataRDD, α)

```
13: $k \leftarrow 2 \;\; FinalDUs \leftarrow \emptyset$
14: while $|CDU^k| > 0$ do
15:     $CDU^k \leftarrow \{du_1^{k-1} \cup du_2^{k-1} \mid du_1^{k-1}, du_2^{k-1} \in DU^{k-1} \bigwedge |du_1^{k-1} \cap du_2^{k-1}| = k-2\}$
16:     $CDU^k \leftarrow \{cdu^k \in CDU^k | \forall cdu^{k-1} \subset cdu^k : cdu^{k-1} \in DU^{k-1}\}$
17:     for $cdu^k \in CDU^k$ do
18:         $\forall C_m \in cdu^k : \tau_{C_m}^{cdu^k} \leftarrow \frac{\alpha(max(C_m)-min(C_m))|O|}{max(V_m)-min(V_m)}$
19:     end for
20:     $\forall cdu^k \in CDU^k : w_{cdu^k} \leftarrow |\{o \in O | \forall C_m \in cdu^k\ o_{mn} \in C_m\}|$
21:     $DU^k \leftarrow \{cdu^k \in CDU^k | \forall C_m \in cdu^k : w_{cdu^k} \ge \tau_{C_m}\}$
22:     $FinalDUs \leftarrow FinalDUs \cup \{du^{k-1} \in DU^{k-1} | \nexists du^k \in DU^k : du^{k-1} \subset du^k\}$
23:     $k \leftarrow k + 1$
24: end while
25: return FinalDUs
```

density thresholds. Pseudo-code of the centralized version of PSCEG is given in Algorithm 1. In **Phase 1**, PSCEG starts by partitioning the value range of each dimension into N equally sized intervals of length $\Delta = \frac{V_2 - V_1}{N}$ (line 3). The object count in each interval represents its weight (line 4). Such intervals and their weights are used as inputs for DBSCAN [1](line 6), whose output is a set of 1-D candidate dense units. By using weighted objects as the input of DBSCAN we can reduce the time complexity from $\mathcal{O}(|O|^2)$ to $\mathcal{O}(N^2)$ with slight loss of accuracy. θ regulates ϵ and $minPoints$ (line 6). Intervals having a density lower

[1]We use the implementation of DBSCAN in scikit-learn which can take weighted objects as input (http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html).

than $\frac{\alpha|O|}{N}$ are filtered out before DBSCAN. Initial density thresholds are calculated using our density estimator (line 8). The object count in each candidate dense unit (cdu^1) is set as its weight (line 11), and those whose weight is higher than the corresponding threshold enter the next phase as dense units (du^1s) (line 12). **Phase 2** is the iterative process of detecting subspace clusters of increasing dimensionality. It first generates cdu^ks by combining two du^{k-1}s that share exactly $k-2$ 1-D clusters (line 15). The anti-monotonicity property [1] is used to prune redundant cdu^ks (line 16). The density thresholds are recalculated to adapt to the increasing dimensionality (line 18). The weight of each cdu^k is then assigned (line 20). A cdu^k is recognized as a du^k if its weight is larger than all thresholds of all 1-D clusters (line 21). Those du^{k-1}s that are not subsets of any du^k are reported as final dense units (line 22). This iterative process terminates when no more dense units appear.

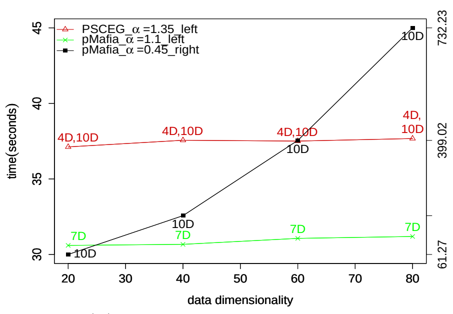
The **parallelization** of PSCEG is based on the Resilient Distributed Datasets (RDDs) [10] paradigm. The key idea distinguishing RDD from MapReduce is its in-memory computation capabilities. Operations on RDDs run in parallel on all cores. PSCEG operates on RDDs as follows: firstly all object values are read and stored in *data-RDD*. The partitioning, the initialization of the weights and the filtering (lines 3-5) are done by a map, a word-count-like [12] and a filter operation on *data-RDD* respectively. One or more 1-D centralized DBSCANs (line 6) run in parallel for each dimension on each core, combining the results and storing them in *CDU-RDD* via a reduce operation. The initialization of density thresholds (line 8) is done by a map operation. The assignment of weights and the generation of DU (line 11-12) are a word-count-like and a map operation on *CDU-RDD*. The result is stored in *DU-RDD*. The generation and pruning of CDU^k is done using two map operations on *DU-RDD* and *CDU-RDD*. Lines 18-21 are done like lines 8-12. The final results are filtered using a map operation on *DU-RDD* and then collected in the master node.

3 Experimental results

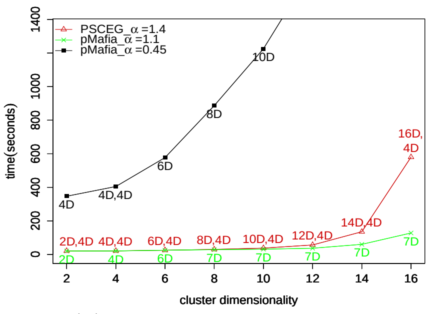
We provide experimental results considering accuracy and scalability w.r.t. the size and dimensionality of the data, as well as the dimensionality of hidden subspace clusters in comparison with MPI-based pMafia (the only available similar parallel algorithm that we know of). We reimplemented pMafia on Spark for the consistency of the experiments. The experiments were run on a 17-node cluster (each with 4 Core(TM) i5 CPUs with 8GB of RAM) using Spark 1.4.1. We used a synthetic data generator (similar to those described in [2, 8]) to test scalability. Public datasets *sset1* [2] (synthetic) and *glass* [3] (real) were used to test the accuracy of both algorithms.

Scalability. To assess the scalability w.r.t. dataset dimensionality we generated 30,000-object datasets of increasing dimensionality, with one 10-D cluster and one 4-D cluster. We set $\theta = 0.01$, which gives very robust performance for PSCEG. Figure 1a shows run-times and clusters found by PSCEG and pMafia

[2]Clustering datasets in University of Eastern Finland, http://cs.joensuu.fi/sipu/datasets/
[3]UCI machine learning repository, http://archive.ics.uci.edu/ml/



(a) data dimensionality



(b) cluster dimensionality

Fig. 1: Runtime w.r.t. dimensionality

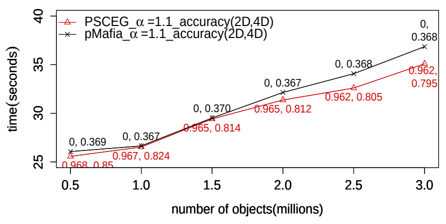


Fig. 2: Runtime w.r.t. datasize

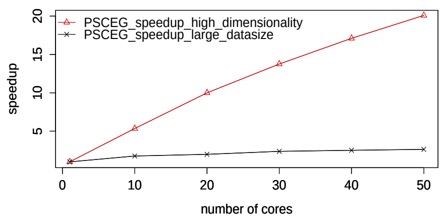


Fig. 3: Speedup

for different values of α. Experimentally we observed that the lower the value of α, the higher the accuracy but the lower the efficiency. PSCEG finds the correct clusters with a relatively large value of $\alpha = 1.35$, in linear time w.r.t. dimensionality. pMafia needs to decrease α to 0.45 to find the 10-D cluster, showing dramatically increased runtime and suggesting at least quadratic scalability (note the different y axes) and failing to find the 4-D cluster. Figure 1b shows scalability w.r.t. cluster dimensionality on 100-D datasets with one cluster of varying dimensionality and one 4-D cluster. PSCEG found all clusters efficiently, while pMafia only found one higher-dimensional cluster after decreasing α to 0.45, incurring dramatically increased computational costs. Figure 2 shows scalability w.r.t. the number of objects on 20-D datasets containing one 2-D and one 4-D cluster. The labels at each point represent accuracies for the found clusters. pMafia failed to find the 2-D cluster, while PSCEG found both with remarkable accuracy. Figure 3 shows the performance of our algorithm for (a) one 100-D dataset of 30,000 objects (high dimensionality) where one 14-D cluster was hidden in the whole feature space, and (b) one 20-D dataset of 1,000,000 objects (large datasize) that contains one 4-D and one 2-D clusters. We define $speedup = \frac{T1C}{TNC}$ where $T1C$ is the execution time using 1 core, and TNC is the execution time using N cores. Near-linear speedup is achieved in both cases.

Accuracy. We applied PSCEG and pMafia to two public available datasets with ground truth information, in which the correlations between dimensions remain unknown, but all objects are classified with class tags. *Sset1* is a public synthetic 2-D dataset widely used as benchmark for testing accuracy. It contains 15 2-D clusters with different shapes, positions and densities. With $\alpha = 0.25, \theta = 0.1$, our algorithm succeeded in finding all 15 clusters, with average accuracy around 90%. We repeated pMafia with different values for each of the three parameters, and the best result showed 13 clusters with a big cluster that consisted of 2 real clusters, with average accuracy below 50%. We also compared both algorithms on *glass*, used previously in an experimental evaluation [7]. PSCEG reported two meaningful subspace clusters: one 8-D cluster that contains 82.2% of all 163 objects of class "window", and one 6-D cluster that successfully classified 72.4% of objects with the "headlamps" class tag. Classes "containers" and "tableware" were not found because there are too few objects of each class (less than 14) to reach the density thresholds. pMafia only found a set of 9-D small clusters each containing no more than 10 objects. This result separated the "window" class into pieces and was difficult to interpret.

4 Conclusions

We presented PSCEG, a novel parallel subspace clustering algorithm that can generate an exact grid of proper size without the need to specify its size and capture the positions of dense units. We also propose a novel adaptive density estimator that is not biased to a certain dimensionality. Experimental results on both synthetic and real datasets show that PSCEG has much better accuracy, scalability and efficiency compared to existing grid-based subspace clustering algorithms, especially for detecting overlapping subspace clusters.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[2] S. Goil, H. Nagesh, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. In *Proceedings of the 5th ACM SIGKDD*, pages 443–452, 1999.

[3] C-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM SIGKDD*, pages 84–93, 1999.

[4] K. Sequeira and M. Zaki. Schism: A new approach for interesting subspace mining. In *Proceedings of the 4th ICDM*, pages 186–193, 2004.

[5] G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *Proceedings of the 23th ICDE*, pages 1250–1254, 2007.

[6] H-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.

[7] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. *VLDB Endowment*, 2(1):1270–1281, 2009.

[8] L. Parsons et al. Evaluating subspace clustering algorithms. In *Workshop on Clustering High Dimensional Data and its Applications, SIAM*, pages 48–56, 2004.

[9] I. Assent, R. Krieger, E. Muller, and T. Seidl. Dusc: Dimensionality unbiased subspace clustering. In *Proceedings of the 7th ICDM*, pages 409–414, 2007.

[10] M. Zaharia et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX*, pages 2–2, 2012.

[11] M. Ester, H-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd SIGKDD*, pages 226–231, 1996.

[12] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.