Levelwise Cluster Mining under a Maximum SSE Constraint

Jeroen De Knijf, Bart Goethals, and Adriana Prado

University of Antwerp, Belgium

Abstract. In this work we cast the problem of finding clusters in the dataset into a frequent itemset mining problem. We show that the SSE measures is monotone with respect to set inclusion, and use this property in an Apriori style algorithm to find optimal clusters in the data. Some preliminary experiments are reported on real life data.

1 Introduction

Clustering is identified as one the core problems in data mining [1]. Moreover, clustering plays a prominent role in many different disciplines in science and engineering, including bio-informatics, linguistics, pattern recognition, earth science and so on. The past decades a waste number of clustering algorithms have been proposed, for example [2–6]. These algorithms mainly differ in required level of domain knowledge, type of cluster that can be found and suitability for high dimensional data. In general, a clustering algorithm uses a similarity measure between objects in order to find an optimal grouping. That is, each group consists of objects that are similar to themselves and dissimilar to objects in other groups. The definition of similarity varies between the different clustering algorithms and depends upon the application domain. Principally, all possible groupings of the data points needs to be considered, which result in an exponential search space. Because of the exponential search space an exact, optimal solution is often infeasible to compute, even for dataset of small size. Therefore, clustering algorithms make use of heuristics to approximate an optimal solution under the similarity measure used. One of the most popular clustering methods for points in Euclidean space is called k-means clustering. Given a set P of n data points in d-dimensional space $\mathcal{R}^d$, and an integer k, the problem is to determine a set of k points in $\mathcal{R}^d$, usually referred to as centers, that minimizes the mean squared Euclidean distance from each data point to it nearest center. This measure is often referred to as Sum of Squared Error, or distortion.

The problem of finding groups of data under given constraints, is a well studied problem in the area of frequent pattern mining. For example, in frequent itemset mining [7] the goal is to derive all sets that occur more often than a user-specified threshold. To accomplish this, a level wise algorithm is used that employs the monotonicity of the constraint to limit the search space. Recent involvements in frequent itemset mining is to derive the most 'interesting' sets that best represent the input data [8–14].

In this paper we use the constraints to find an optimal clustering in an Apriori-style algorithm. We show that the SSE constraint is monotone under set inclusion, and use this fact in our mining algorithm, to perform a level wise search towards the optimal solution. The resulting sets represents groups of points that are more close towards each other than to any other points in the input data. Finally, the post-processing step takes as input all derived sets and combine the ones that best partition the data. Contrary to k-means, we do not use a parameter that specify the number of desired clusters in the data. Instead, our algorithm produces several clusterings, where the kth iteration produces in general far less clusters than the $k-1$ iteration. As a result, a user is not faced with cumbersome process of apriori determining the exact clusters and hence our algorithm can be used for truly explorative data analysis tasks.

The rest of this paper is organized as follows: in the next section we introduce the basic concepts and the notation used. In section 3 we present the mining algorithm Maximum SSE Miner (MSSEM). The following section describes the experiments performed on several real life datasets. Finally, in the last section we discuss the results, draw conclusions and give some pointers for further research.

2 Preliminaries

Given x, y, with $x, y \in \mathcal{R}^d$, let $\Delta(x, y)$ denote the Euclidean distance between x and y, i.e.

$$\Delta(x, y) = \sqrt{\sum_{i}^{d} (x_i - y_i)^2}.$$

Moreover, let $\Delta^2(x, y)$ denotes the squared distance between x and y, i.e.

$$\Delta^2(x, y) = \sum_{i}^{d} (x_i - y_i)^2.$$

For a finite set $X \subset \mathcal{R}^d$ and a point $y \in \mathcal{R}^d$ the SSE of X relatively to y, denoted as $\mathrm{SSE}(X, y)$, equals the sum of the squared distances between any point in X and y, i.e.

$$\mathrm{SSE}(X, y) = \sum_{x \in X} \Delta^2(x, y).$$

In this paper, we only use the SSE of X relatively to the mean of X, which we abbreviate to $\mathrm{SSE}(X)$.

2.1 Problem Statement

Given a finite set $P \subset \mathcal{R}^d$ of size n, the goal is to derive k sets $\{P_1, \ldots, P_k\}$ such that the k sets are a partition of P with minimal SSE, more formally:

1. $P_i \cap P_j = \varnothing$ for $1 \leq i, j \leq k$ and $i \neq j$.

2. $\bigcup_{i=1}^{k} P_i = P$.
3. $\nexists\{M_1, \ldots, M_k\}$ such that:
   (a) $M_i \cap M_j = \varnothing$ for $1 \leq i, j \leq k$ and $i \neq j$.
   (b) $\bigcup_{i=1}^{k} M_i = P$.
   (c) $\sum_{i=1}^{k} \text{SSE}(M_i) < \sum_{i=1}^{k} \text{SSE}(P_i)$.

As stated earlier, in this work we don't apriori fix the k value. Instead we derive a whole range of k values and the corresponding clusterings. Further noteworthy, the trivial solution for the problem stated above is to take the k value equal to the number of points in the input data, i.e. $k = |P|$. However, this valid solution is from a user perspective worthless.

3 Algorithm

In this section we describe the different components of MSSEM in more detail. But first we show that the SSE measure is monotone under set inclusion.

It is well known from literature (see for example [2]) that the best point to minimize the SSE of a set X is the mean or center of X. Given this observation, we can state our theorem on which this algorithm is base.

Theorem 1 (Monotonicity of the SSE measure w.r.t. set inclusion) *For any two finite set $X, Y \in \mathcal{R}^d$ it holds that:*

$$X \subseteq Y \Rightarrow \text{SSE}(X) \leq \text{SSE}(Y).$$

Proof. Let Z contain the points of Y which are not part of X, i.e. $Z = Y \setminus X$. Moreover, let c_1 be the center of X and c_2 be the center of Y. Because, the center of a set is the points that minimizes the SSE of the set we have that $\text{SSE}(X, c_2) \geq \text{SSE}(X, c_1)$. Moreover, because $\text{SSE}(Z, c_2) \geq 0$ it follows that $\text{SSE}(X, c_2) + \text{SSE}(Z, c_2) \geq \text{SSE}(X)$. Hence, $\text{SSE}(X) \leq \text{SSE}(Y)$.

Given the previous theorem, a straightforward approach would be to derive all maximal sets for a user defined maximum SSE threshold. Next, a post-processing step is applied to extract a cover of the data. However, such a simple approach result in multiple problems. First, for realistic dataset it is unlikely that in the ideal clustering, all clusters have more or less the same SSE value. This, because some regions are more dense than other and the number of points in the true clusters can vary drastically. Hence, the maximum SSE value needed to find interesting clusters in one region of the input space, results in an explosion of patterns in another region. The other way around, the maximum SSE value needed in a dense region would overlook possible clusters in sparse regions. Second, determining a reasonable value for the maximum SSE is an almost impossible task for a user.

To overcome these problems we applied a mining algorithm for every point in the data and its region. A region of a point is defined as the points that are not further away than the average distance between all point in the data. Moreover, the support threshold for a region depends on the density of the region. Finally, we define an order among the points depending on the number of points in its region. Points in more dense regions are processed before points in sparser regions, moreover the points in dense regions are removed from all other regions after they have been processed. As a result, for every point $p \in P$ and its region denoted as p_r the mining algorithm enumerate sets $\{i, \ldots, j\}$ where $\{i, \ldots, j\} \subseteq p_r$ and $\forall x \in \{i, \ldots, j\} : \mathrm{ord}(x) > \mathrm{ord}(p)$. Where $\mathrm{ord}(p)$ denotes the order assigned to p which depends on $|p_r|$.

Algorithm 1 Derive all sets that satisfy the constraints

```
Input: $P$, npoints
Output: all set that satisfy the constraint
 1: OUT $\leftarrow \varnothing$
 2: $mean_{\Delta}(P) \leftarrow$ mean distance between all pair of point in $P$
 3: for all $p \in P$ do
 4:     $p_r \leftarrow p_r \cup \{q | q \in P \wedge \Delta(p, q) \leq mean_{\Delta}(P)\}$
 5: end for
 6: sort($P$) according to $|p_r|$ in descending order
 7: $\forall p \in P$, ord($p$) equals the position of $p$ in array
 8: remove of all occurrences of point $p$ in $q_r$ if $\mathrm{ord}(p) < \mathrm{ord}(q)$
 9: for all $p \in P$ do
10:     OUT $\leftarrow$ OUT $\cup$ mine($\{p\}, \frac{p_r, mean_{\Delta}(P) \times mean_{\Delta}(P) \times \mathrm{npoints}}{|p_r|}$)
11: end for
12: return OUT
```

The next question to answer is, what is a good maximum SSE threshold for a region p_r and perhaps even more important how does the threshold for p_r relates to a threshold for q_r, with $q \in P$. Let $mean_{\Delta}(P)$ denote the average distance between all pair of points in P. Obviously, if we take $mean_{\Delta}(P) \times mean_{\Delta}(P) \times |p_r|$ as maximum SSE then $\{p\} \cup p_r$ is the maximal set that satisfy the SSE constraint. Clearly, taking this value as maximum SSE is senseless. Instead, we let a user specify how many points should approximately be in the largest set. Given this parameter, we derive the maximum SSE as follows: $\frac{mean_{\Delta}(P) \times mean_{\Delta}(P) \times \#\text{ of points}}{|p_r|}$. Note that, we use the same input parameter (# of points) for all different regions. The pseudo-code of these pre-processing steps is given in Algorithm 1. The input arguments are the set of points P and the used defined parameter npoints. The lines $2 - -8$ are used to compute the region for each point in the dataset. On line 10, the levelwise mining algorithm is called (Algorithm 2), for each point in the dataset and its previously computed region.

The pseudo-code of the actual mining algorithm, is given in Algorithm 2. As stated earlier, this algorithm is a straightforward levelwise algorithm that uses Theorem 1 to limit the search space. Again, we assume an ordering on the points

Algorithm 2 **level wise search algorithm**

```
Input: $\{p_1, \ldots, p_n\}, p_r, \text{maxsse}$
Output: all subsets of $p_r$ with a lower SSE than maxsse
 1: $mean_\Delta(P) \leftarrow$ mean distance between all pairs of points in $P$
 2: for all $q \in p_r$ do
 3:     if $max(\{\text{ord}(p_1), \ldots, \text{ord}(p_n)\}) < \text{ord}(q) \wedge SSE(\{p_1, \ldots, p_n\} \cup \{q\}) \leq \text{maxsse}$
        then
 4:         $\text{OUT} \leftarrow \text{OUT} \cup \{\{p_1, \ldots, p_n\} \cup \{q\}\}$
 5:         $\text{OUT} \leftarrow \text{OUT} \cup \textbf{mine}(\{p_1, \ldots, p_n\} \cup \{q\}, p_r, \text{maxsse})$
 6:     end if
 7: end for
 8: return OUT
```

in the region. However, this order can be the same as determined in Algorithm 1. In general, the output of the levelwise search algorithm consists of overlapping sets that satisfy the constraint. Hence, the output is not an partition of the input space.

Algorithm 3 **Post-processing the results**

```
Input: A List $L$ of all sets that satisfy the max SSE constraint
Output: RES a clustering of the input
 1: while $L \neq \emptyset$ do
 2:     sort $L$ first according to cluster size, then according to SSE
 3:     $\text{RES} \leftarrow \text{RES} \cup top(L)$
 4:     for all $X \in L$ do
 5:         if $X \cap top(L) \neq \emptyset$ then
 6:             $L \leftarrow L \setminus X$
 7:         end if
 8:     end for
 9: end while
10: return RES
```

To select a partition from all the sets that best describe the input data we use a simple heuristic to take the cover of the input data. The pseudo-code of this post-processing step is given in Algorithm 3. First we order the output sets on the number of points they contain (line 2). In case two sets have equal size they are ordered on SSE value. Then the first set in the ordered list is taken and added to the results (line 3). Next, all sets that contain one or more points of the set that is previously added to the results are removed (line $4--8$). We repeat this procedure till all output sets are either removed or in the results sets.

Finally, to deliver a whole range of number of clusters we repeat the algorithms on the output derived. More precisely, let $\{X_1, \ldots, X_k\}$ be the k clusters derived on input P. Then, for each X_i with $1 \leq i \leq k$ we take the center of

Algorithm 4 Max SSE Miner

```
Input: $P$, npoints
1: RES $\leftarrow \varnothing$
2: while $|\text{RES}| \neq |P|$ do
3:     $L \leftarrow$ Pre-process($P$, npoints)
4:     RES $\leftarrow$ Post-process($L$)
5:     DISPLAY(RES)
6:     $P \leftarrow \varnothing$
7:     for all $X \in RES$ do
8:         $P \leftarrow P \cup \text{mean}(X)$
9:     end for
10: end while
```

X_i and add these points to a new input set P'. We repeat the previously described algorithms on the newly created input set. This repetition stops when the number of clusters obtained stays the same. The pseudo-code of the MSSEM algorithm is given in Algorithm 4. The input parameters of this algorithm is the set of points P and the user specified parameter *npoints*, the output is range of clusterings of the input set.

We implemented a prototype of the described MSSEM algorithm. However, for practical reasons more optimizations could be applied. One obvious optimization is to derive only maximal sets that satisfy the user defined threshold. Note that in this case it is, because of the post-processing step used, highly likely that we need to recompute the support of one or more of the subsets of a maximal set. Further, note that concepts as closed sets are not interesting because of the continuous scale of the support.

4 Experiments

In this section, we report on the preliminary results obtained of the MSSEM mining algorithm on three real life datasets. The goal of the experiments is to qualitatively compare our results, with the results obtained by k-means as implemented in Weka [15]. For the evaluation of the clusters, we use the sum of the SSE values for the different clusters. To compare the results to k-means, we performed for each obtained clustering of MSSEM a run of k-means with as input parameter exactly the same number of clusters as the number of clusters that obtained by MSSEM. Another criterion mentioned is, if MSSEM is able to derive the true number of clusters, where the true number of clusters means the actual classes in the dataset.

We used three real-word datasets, all available in the UCI repository [16]. The characteristics of the datasets are presented in Table 4. We only consider the descriptive dimensions for clustering, while the class attribute is used in order to evaluate the result.

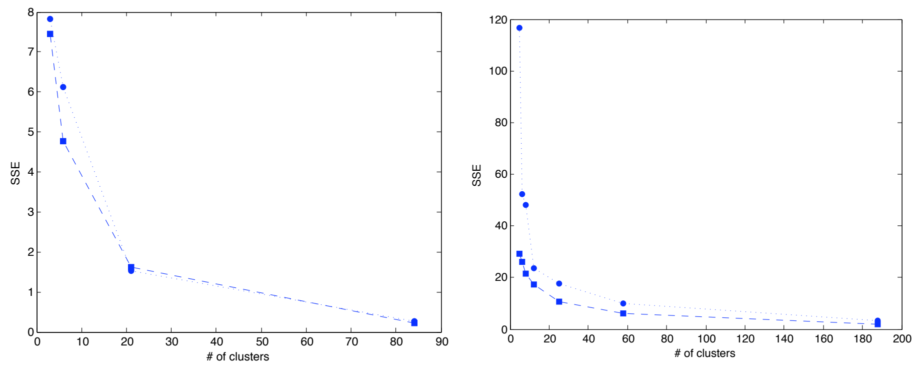

Fig. 1. Results for the Iris dataset (left) and the Ecoli dataset (right). The dashed line with square markers states the results obtained by the MSSEM approach. The dotted line with circle markers denote the results obtained by the k-means algorithm.

Dataset	#points	#dimensions	#classes
Iris	150	5	3
Ecoli	336	8	8
Sonar	208	61	2

Table 1. Characteristics of the dataset used.

The results obtained for the Iris dataset and the Ecoli dataset are displayed in Figure 1. For both datasets MSSEM obtained the true number of clusters. The results shows that for almost all clusterings obtained by MSSEM the corresponding SSE value is better than the SSE value obtained by k-means. For the Ecoli dataset the difference is considerable. For different support values the results were more or less the same, except that MSSEM did not always succeed in finding the true number of clusters. The results for the Sonar dataset (displayed in Figure 2) show a somewhat different pictures. The results obtained for MSSEM are, in all but one case, slightly worse than those obtained by k-means. Moreover, MSSEM was not able to find the true number of clusters.

5 Related Work

Although a lot of clustering algorithms have been proposed, for example [2–6], almost no approaches uses local patterns. In this section we briefly review some clustering approaches that does uses local patterns.

For market basket data, Wang et al. [17] uses the frequency of itemsets to form clusters. With this approach they avoid pairwise comparison between the different items. Recently, Van Leeuwen et al. [18] uses an MDL based approach to partition transactional data. First, a so called code table is constructed from

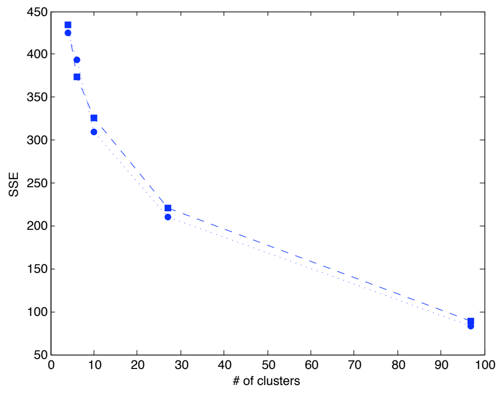

Fig. 2. Results for the Sonar dataset. The dashed line with square markers states the results obtained by the MSSEM approach. The dotted line with circle markers denote the results obtained by the k-means algorithm.

local patterns, then they search for an optimal decomposition of the code tables that partition the data. One major difference between MSSEM and the two previously described approaches is that MSSEM operates on numerical data while the other approaches require transactional data.

In subspace clustering, the goal is to find clusters in one or more axis-parallel subspaces of the input space. CLIQUE [5] is a grid based subspace mining algorithm. For all dimensions, the data space is partitioned by a grid of equisized bins. All bins that contains more than a user defined threshold number of point are considered dense. A cluster is defined as a maximal set of adjacent dense bins. Starting with all one-dimensional dense units, the algorithms uses a level wise search to enumerate $(k+1)$ dimensional dense unit. In general the number of clusters found by CLIQUE can be enormous. The difference with our approach is that CLIQUE find clusters in all different axis-parallel subspaces and that the similarity measure is density instead of SSE.

6 Discussion and Conclusion

In this paper, we showed that the SSE measure is monotone with respect to set inclusion and proposed an algorithm that uses the SSE measure as monotone constraint in an Apriori style algorithm. We implemented a prototype of MSSEM and conducted some preliminary experiments on real datasets. Besides implementation issues, there are some more points of consideration:

1. While MSSEM reports a whole range of possible clusterings, it is not guaranteed that the true number of clusters will be derived. Moreover, the number of clusters derived is rather sensitive for the user defined input parameter.

One possible direction to solve these problems is to perform more advanced post-processing, i.e. instead of our simple heuristic algorithm to cover the input space a more fundamental approach might be needed. Another direction is to take a large number of derived clusters as input of another clustering algorithm (for example an hierarchical clustering algorithm). In such an approach MSSEM could be considered as a pre-processing step for a clustering algorithm.

2. An issue prevalent in most clustering algorithms are outliers. Obviously, MSSEM is likely to be sensitive for outliers. A straightforward approach to handle theses cases might be to disregard point that have a larger mean distance to other points than the average mean distance between any pair of points.

However, besides the open issues the results also show that MSSEM is able to find in case of the Ecoli dataset far better solutions than k-means algorithm. Moreover, for the other two datasets the difference in obtained results was minimal. Concluding, our work shows local patterns can be used to capture essential clusters in a dataset. However, more research is needed to fine-tune the algorithm.

References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2006)
2. Macqueen, J.: Some methods of classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. (1967) 281–297
3. Ester, M., Kriegel, H., Sander, J., Xu, J.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (1996) 226–231
4. Guha, S., Rastogi, R., Shim, K.: Cure: an efficient clustering algorithm for large databases. In: Proceedings ACM SIGMOD International Conference on Management of Data. (1998) 73–84
5. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data. Data Min. Knowl. Discov. **11**(1) (2005) 5–33
6. Domeniconi, C., Gunopulos, D., Ma, S., Yan, B., Al-Razgan, M., Papadopoulos, D.: Locally adaptive metrics for clustering high dimensional data. Data Min. Knowl. Discov. **14**(1) (2007) 63–97
7. Agrawal, R., Mannila, H., Srikant, R., H.Toivonen, Verkamo, A.: Fast discovery of association rules. In Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., eds.: Advances in Knowledge Discovery and Data Mining. (1996) 307–328
8. Geerts, F., Goethals, B., Mielikinen, T.: Tiling databases. In: Discovery Science, Springer (2004) 278–289
9. Heikinheimo, H., Seppänen, J., Hinkkanen, E., Mannila, H., Mielikäinen, T.: Finding low-entropy sets and trees from binary data. In Berkhin, P., Caruana, R., Wu, X., eds.: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2007) 350–359

10. Knobbe, A., Ho, E.: Maximally informative k-itemsets and their efficient discovery. In Eliassi-Rad, T., Ungar, L., Craven, M., Gunopulos, D., eds.: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2006) 237–244
11. Bringmann, B., Zimmermann, A.: The chosen few: On identifying valuable patterns. In: IEE International Conference on Data Mining. (2007) 63–72
12. Tatti, N., Vreeken, J.: Finding good itemsets by packing data. In: IEE International Conference on Data Mining. (2008) 588–597
13. Siebes, A., Vreeken, J., van Leeuwen, M.: Item sets that compress. In Ghosh, J., Lambert, D., Skillicorn, D., Srivastava, J., eds.: SIAM International Conference on Data Mining. (2006)
14. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery **15**(1) (2007) 55–86
15. Witten, I., E., F.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
16. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
17. Wang, K., Xu, C., Liu, B.: Clustering transactions using large items. In: Proceedings of the eighth international conference on Information and knowledge management. (1999) 483–490
18. van Leeuwen, M., Vreeken, J., Siebes, A.: Identifying the components. Data Mining and Knowledge Discovery **19**(2) (2009) 176–193