Evaluation of Monte Carlo Subspace Clustering with OpenSubspace

David C. Hunn, Clark F. Olson[1]
[1]Computing and Software Systems, University of Washington, Bothell, WA, U.S.A.

Abstract - *We present the results of a thorough evaluation of the subspace clustering algorithm SEPC using the OpenSubspace framework. We show that SEPC outperforms competing projected and subspace clustering algorithms on synthetic and some real world data sets. We also show that SEPC can be used to effectively discover clusters with overlapping objects (i.e., subspace clustering).*

Keywords: subspace clustering, projected clustering, OpenSubspace

1 Introduction

Clustering algorithms attempt to divide objects in a data set into groups such that objects in a group are more similar to one another than to other objects in the data set. Similarity is usually based on distance. For data sets with few attributes, approaches such as k-means can be used to perform clustering in the full feature space. However, in many applications, data sets contain large numbers of attributes per object. For example, in some text processing applications each object is a term frequency vector whose length is equal to the number of terms under analysis. Such frequency vectors can have thousands of attributes depending on the size of the dictionary. As the dimensionality of data sets increases, traditional approaches that look for clusters in the full feature space start to fail. In part, the difficulty is that the longest and shortest distances in a data set will approach one another as the number of dimensions increases [1]. Thus, increasing dimensionality erodes the usefulness of distance metrics in determining the relative similarity of objects in a data set. A common solution is to use dimensionality reduction tools like principal component analysis (PCA) [2] to project the data set onto fewer dimensions. The resulting data set can then be clustered using traditional techniques. However, applying PCA produces a single subspace and in many applications, clusters exist in different subspaces of the full feature space. Thus, applying PCA may mask clusters and hide interesting results. One solution to these problems is subspace clustering. Subspace clustering aims to identify clusters of objects and their associated subspaces.

The most general aim of subspace clustering is to find all clusters in arbitrarily aligned subspaces. Unfortunately, this form of the problem has an infinite search space and finding clusters under these conditions is difficult. Instead, most approaches rely on heuristics to reduce the search space to something more practical. For example, in many applications, it is reasonable to assume that the attributes of the data are not correlated with one another. This enables one to restrict the search for clusters to only those that are axis-aligned—reducing the number of possible subspaces from infinite to 2^d, where d is the number of dimensions in the data set.

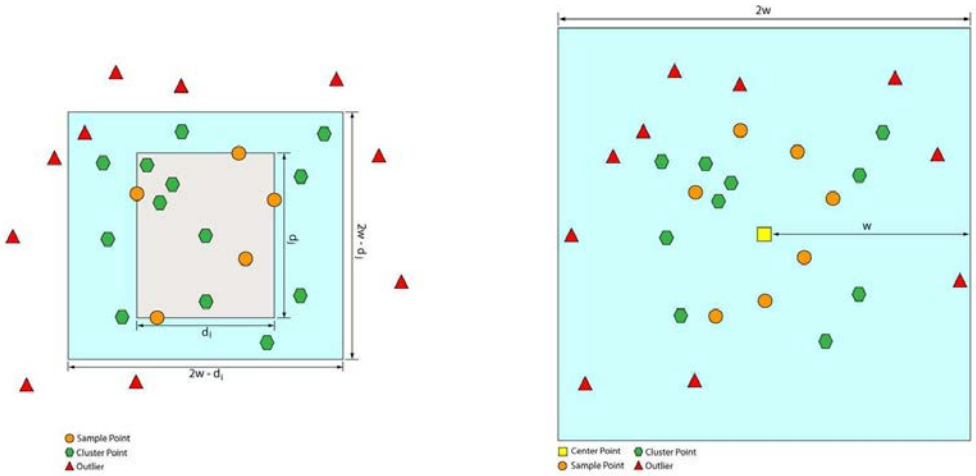
In this paper we evaluate an algorithm for performing projective clustering called Simple and Efficient Projective Clustering (SEPC) [3] using the OpenSubspace framework [4]. We present the results of a thorough evaluation of SEPC with both synthetic and real-world data sets including comparisons with competing approaches. We will also show that SEPC can be used to effectively discover clusters with overlapping objects (i.e., subspace clustering).

2 Related Work

For a thorough review of the current state of subspace and projected clustering, see [7]. We provide a brief overview of some historical and closely related algorithms to the current work.

CLIQUE [8] is often cited as the earliest subspace-clustering algorithm. In CLIQUE, the data set is discretized into ξ intervals of equal length. Units containing a sufficient number of points are considered dense. Adjacent dense cells are joined together to create clusters. The algorithm first discovers all one-dimensional dense units and then in a priori fashion searches for subspace clusters. The algorithm is made more efficient by leveraging the downward closure property of subspace clusters to prune the search space.

PROCLUS [9] is another approach to the problem of subspace clustering. Where CLIQUE is a bottom-up approach, PROCLUS builds clusters in a top-down fashion. PROCLUS is a k-medoid-like clustering algorithm. It partitions the data into k clusters with an average number of dimensions equal to l. In the first stage of the algorithm, a set of candidate medoids (M) is sampled from the data set. From M, k medoids are selected and the subspaces for each are determined by minimizing the standard deviation of the distances of the points in the neighborhood of the medoids to the corresponding medoid along each dimension. Then points are assigned to the medoid they are closest to using a distance metric that only considers the relevant subspaces for each medoid. In a refinement phase, medoids may be switched out for other members of M—the pool of medoids. The result is a strict partitioning of the data set into k parts along with a set of outlier points.



(a) (b)

Fig. 1. A comparison between the SEPC and DOC algorithms for determining cluster dimensions and cluster points using a discriminating set. (a) The SEPC algorithm uses a sheath of width $\boldsymbol{w}$ to determine if a discriminating set congregates in a dimension. When the set congregates, a larger sheath with width between $\boldsymbol{w}$ and $\boldsymbol{2w}$ is used to determine additional data points that are added to the cluster. (b) The DOC and FastDOC algorithms use a sheath with width $\boldsymbol{2w}$ both to determine if the discriminating set congregates in a dimension and to find additional data points that are added to the cluster.

DOC [5] defines a global measure of cluster quality to determine an optimal cluster. Given a subspace cluster that contains a set of objects C and set of attributes D, the function $\mu(|C|,|D|) = |C|/\beta^{|D|}$ determines the quality of the cluster. β is a user defined constant that determines the tradeoff between objects and attributes in a cluster with the restriction $0 < \beta < 0.5$. The user must also specify a cluster width w, that is used to determine both the relevant attributes and object membership in the cluster. To discover clusters, the algorithm iteratively samples the data set in Monte Carlo fashion. DOC uses two loops to find an optimal cluster: an inner and outer loop. In the outer loop, it randomly samples medoids from the data set. Then in the inner loop, it randomly samples a small number of points from the data set. This small set of points is referred to as the discriminating set. The relevant dimensions of the hypothesized cluster are determined by calculating the distance between the medoid and the points in the discriminating set. This process is repeated many times and the highest quality cluster found is reported.

A closely related algorithm to DOC is MineClus [6]. MineClus uses a similar Monte Carlo framework to sample medoids from the data set. However, in MineClus the search for relevant subspaces is transformed into a frequent pattern tree growth method. This replaces the inner loop of DOC turning it into a determinative step. This replacement results in a significant decrease in running time compared to the DOC algorithm.

3 Approach

SEPC [3] is an iterative Monte Carlo algorithm inspired by DOC [5]. It inherited the cluster model as well as the quality function used by DOC. However, in SEPC, the outer loop in which DOC randomly samples the data set to determine a cluster medoid has been discarded. Instead, clusters are hypothesized using just the discriminating set.

In each trial, a small set of data points (the discriminating set) is sampled randomly from the data set. The minimum and maximum in each dimension of the discriminating set is determined. If the difference between the minimum and maximum value in a given dimension is less than a fixed width w, then a congregating dimension has been discovered. This results in a sheath of width w for determining the congregating dimensions. In contrast, the DOC algorithm uses two loops to generate a hypothetical cluster. In the first loop, seed points are sampled from the data set and in the second loop, a set of discriminating points is sampled from the data set. The distance along each dimension from the seed points to the points in the discriminating set is determined. The hypothesized cluster is said to congregate in the dimensions for which the distance is less than w. This results in a sheath of width $2w$ for determining congregating dimensions, twice as large in each dimension as the SEPC sheath. The narrower sheath of SEPC improves the detection of truly congregating dimensions. It also makes it possible to use values of $\beta > 0.5$ (DOC is limited to $\beta < 0.5$).

Once all congregating dimensions have been determined, the hypothesized cluster is populated with points from the data set. However, the distances calculated to determine the congregating dimensions cannot be used to determine which points belong to the hypothesized cluster, since they are generally too narrow to capture the extremal points in the cluster. Instead, the span of the sheath is determined in each congregating dimension by subtracting w from the maximum value and adding w to the minimum value. The length of the

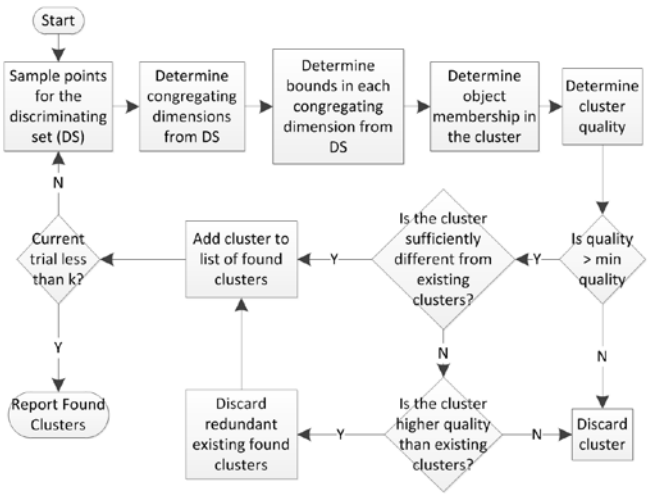


Fig. 2. Finding overlapping clusters with SEPC.

resulting range is $2w - d_i$, where d_i is the absolute difference between the minimum and maximum value in the discriminating set along dimension i. See Fig. 1 (a). In contrast, the DOC algorithm uses the same sheath width $(2w)$ for determining congregating dimensions and point membership in the cluster. See Fig. 1 (b).

After point or object membership has been determined for a hypothesized cluster, its quality can be determined. If the quality is high enough, then it is retained. In disjoint mode, the points in the newly discovered cluster will be removed from consideration for membership in subsequently discovered clusters. Alternatively, if the user wishes to discover clusters with overlapping points, the hypothesized cluster will only be kept if it is qualitatively different from clusters that have already been discovered or if it is of higher quality than an existing cluster that it significantly overlaps with.

3.1 Soft Cluster Equality

Using the algorithm in non-disjoint mode is problematic if we do not remove duplicate clusters. Since points are not removed from consideration when they are assigned to clusters, the algorithm needs to check that each newly found cluster is unique and has not been previously discovered. When a new cluster is discovered, it is compared to existing found clusters. However, using a strict test for equality will result in a large number of clusters being discovered that are not very different from one another. To solve this problem, the algorithm allows the user to loosen the criteria for equality between clusters. This allows the user to tune the algorithm to yield only clusters that are truly unique.

Our test for cluster equality involves both the set of objects in each cluster as well as the subspaces spanned by each cluster. This dual check is necessary since a purely object-based method for determining cluster equivalence would be error prone. Consider two subspace clusters $C_1(O_1, S_1)$ and $C_2(O_2, S_2)$, with $O_1 \subseteq O_2$. Without considering the subspaces of the two clusters we may come to the conclusion that C_1 is redundant with respect to C_2. However, consider the case where $S_1 \neq S_2$. In this case, we have discovered not one but two conceptually distinct clusters. Despite sharing objects, the two clusters describe a different relationship among those objects, thus the two clusters would not be equivalent.

The equivalence check is performed in two steps: first we determine if the two clusters span roughly the same subspace based on a user-specified tolerance (e.g. if two clusters share 90% of the same attributes then they span roughly the same subspace). Then if the two clusters are determined to span roughly the same subspace, we determine the amount of overlap between their respective sets of objects. If the object overlap between the two clusters exceeds a user-specified tolerance then the two clusters are considered to cover roughly the same set of objects (e.g. if two clusters share 90% of the same objects then they cover roughly the same set of objects). Formally, given two clusters $C_i(O_i, S_i)$ and $C_j(O_j, S_j)$, $C_i = C_j$ if

$$\frac{|S_i \cap S_j|}{|S_j|} \geq Min\ Subspace\ Overlap \qquad (1)$$

and

$$\frac{|O_i \cap O_j|}{|O_j|} \geq Min\ Object\ Overlap. \qquad (2)$$

Where $Min\ Subspace\ Overlap$ and $Min\ Object\ Overlap$ are user specified values between zero and one. Note that this check is not commutative, since it determines the percent overlap of attributes and objects by dividing by the cardinality of one of the clusters.

3.2 Using Soft Cluster Equality

Since our check for subspace cluster equality is not commutative, we perform the check in both directions in the following way: each newly hypothesized cluster is checked against existing clusters using itself as the index. Therefore, each new cluster must be sufficiently unique with respect to existing clusters in order to be considered for inclusion in the clustering results. In other words, a user-specified percentage of a new cluster's subspace must be unique. Failing that, a user-specified percentage of the new cluster's set of objects must be unique. If a newly hypothesized cluster is determined to be redundant with respect to an existing cluster by this metric, then we discard the new cluster if its quality is lower than the existing cluster. If a newly hypothesized cluster is sufficiently unique (or of higher quality than redundant existing clusters) then we perform the equivalence check in the other direction. In this case, all existing clusters that are determined to be redundant with respect to the new cluster are removed from the clustering results. Fig. 2 depicts the process used by SEPC to discover overlapping clusters in a data set.

4 Quality Metrics

Some subspace clustering metrics used in OpenSubspace are object-based [4]. That is, they ignore the congregating dimensions of clusters in evaluating cluster quality. Instead, they rely entirely on how objects have been allocated into found clusters compared to the "true" allocation. This approach works sufficiently well when points belong to only one cluster. However, in some instances, it is advantageous to allow points to belong to multiple clusters that span different subspaces. In such cases, the above metrics will yield misleading results. For example, the synthetic data sets provided with the OpenSubspace framework typically have one or two clusters that, point-wise, are subsets of other clusters. However, the super- and sub-clusters span a different set of dimensions (typically, the larger cluster spans fewer dimensions). This causes problems for purely object-based metrics, because, the sub-clusters are not unique on a purely object basis. Thus, metrics like clustering error (CE) [7], account for the subspace spanned by each object.

For the following discussion, it is useful to define two types of clusters: *found clusters* and *hidden clusters*. A found cluster is returned as part of the results of running a clustering algorithm on a given data set. In contrast, a hidden cluster is known within a data set. In the case of synthetic data sets both the objects and subspace of hidden clusters are known, however, for real world data, we typically are limited to information about object membership in hidden clusters.

4.1 F1

In OpenSubspace, F1 is an object-based metric computed with respect to each hidden cluster. It is composed of two sub-metrics called recall and precision. A high recall corresponds to a high coverage of objects from a hidden cluster while a high precision denotes a low coverage of objects from other clusters. Prior to calculating F1, found clusters are mapped to the hidden cluster they overlap with the most. Then recall and precision are determined for each hidden cluster. The F1 scores for the hidden clusters are determined by taking the harmonic mean of their recall and precision scores. The overall F1 score is the average of the F1 scores of each hidden cluster.

Problems arise with the F1-measure when there are overlapping hidden clusters that span different subspaces. For F1, the trouble arises when found clusters are mapped to hidden clusters. Since, the mapping is done purely on a point or object basis, when one hidden cluster is a subset of another, there is a high likelihood that the smaller cluster will "capture" found clusters overlapping both the super- and sub-hidden clusters. This happens, because the overlap between a hidden and found cluster is normalized by the cardinality of the hidden cluster. This results in a bias towards smaller hidden sub-clusters.

4.2 Clustering Error

Clustering error (CE) [7] addresses the problem of overlapping clusters by taking into account the dimensions spanned by objects in a cluster. It does this by using *subobjects* in place of objects in its calculation. A subobject is the combination of an object together with the dimensions spanned by the cluster to which it belongs. This allows an object to belong to multiple clusters and still be unique for the purpose of quality measurement. CE measures the extent to which the subobjects in hidden clusters overlap with the subobjects in found clusters. However, before determining overlap between the two sets of clusters, an optimal mapping of found clusters to hidden clusters is performed, which may result in excess found clusters that are not mapped to any hidden cluster and vice versa. This solves a problem common to many subspace clustering quality metrics. Namely, many metrics fail to distinguish between the case where many found clusters overlap a single hidden cluster and, the case where only a single found cluster overlaps a hidden cluster. The second case is more desirable than the first.

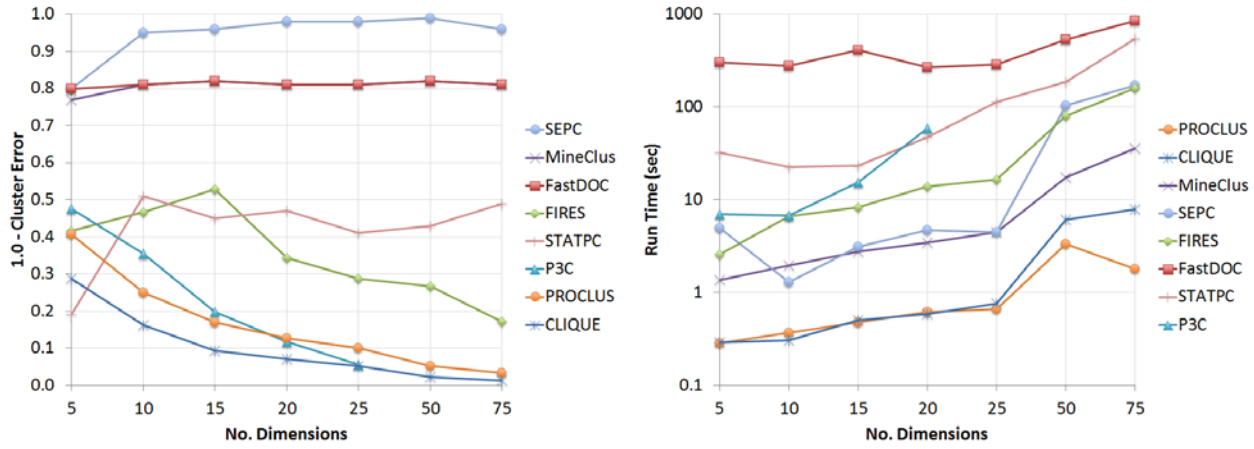
One drawback to CE, is that in real world data sets, the congregating dimensions are often unknown, limiting this method's ability to judge a cluster's quality with respect to its subspace.

5 Experiments

We have reproduced the experiments performed in [8] using the SEPC algorithm and several competing algorithms. We have focused on examining the general properties of SEPC compared to several other subspace clustering algorithms. In [8], each experiment was conducted by trying many different parameter settings for each algorithm in an attempt to obtain the maximum possible performance for each. In addition, each run of an algorithm was limited to 30 minutes. For this evaluation, we used the same strategy. The following algorithms, as implemented in OpenSubspace, were used in this comparison: CLIQUE [9], DOC [5], MineClus [6], FIRES [10], PROCLUS [11], P3C [12], and STATPC [13]. Each algorithm has been tuned using the parameter settings provided by Mueller et al. [8]. The experiments were run on machines with 1.8GHz Dual-Core AMD Opteron™ 2210 processors and 2GB memory running Red Hat Linux 5.9.

5.1 Synthetic Data

OpenSubspace is packaged with three synthetic data sets, each intended to explore a different aspect of algorithm performance. These data sets enable evaluation over increasing dimensionality (number of attributes), over increasing data set size (number of objects), and over increasing amounts of noise (irrelevant objects). Additionally, all of the data sets contain overlapping hidden clusters. That is, they contain clusters that share objects, but span different subspaces. Thus, we applied SEPC in non-disjoint mode to maximize its possible achievable performance.

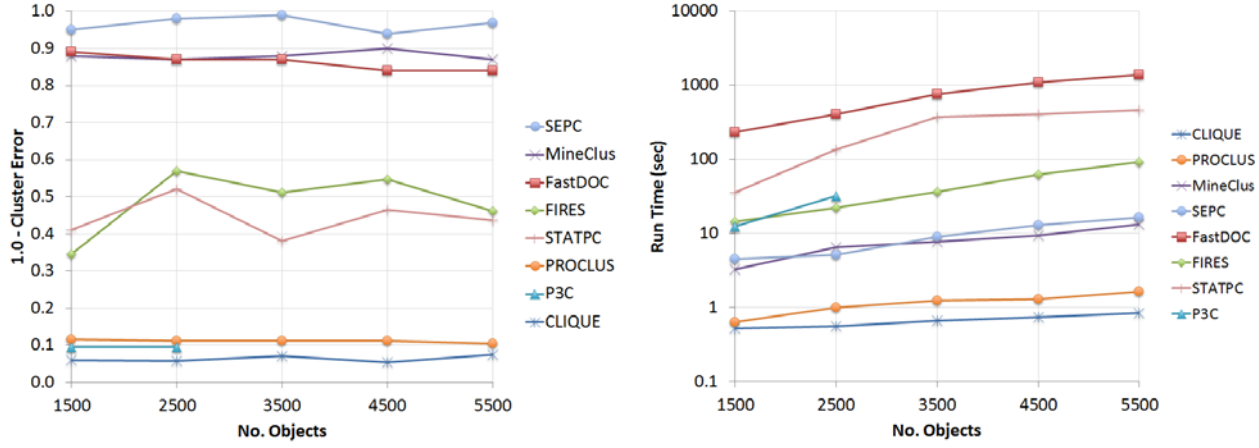


(a) (b)

Fig. 3. Algorithm performance over an increasing number of dimensions measured by (a) clustering error and (b) run time.

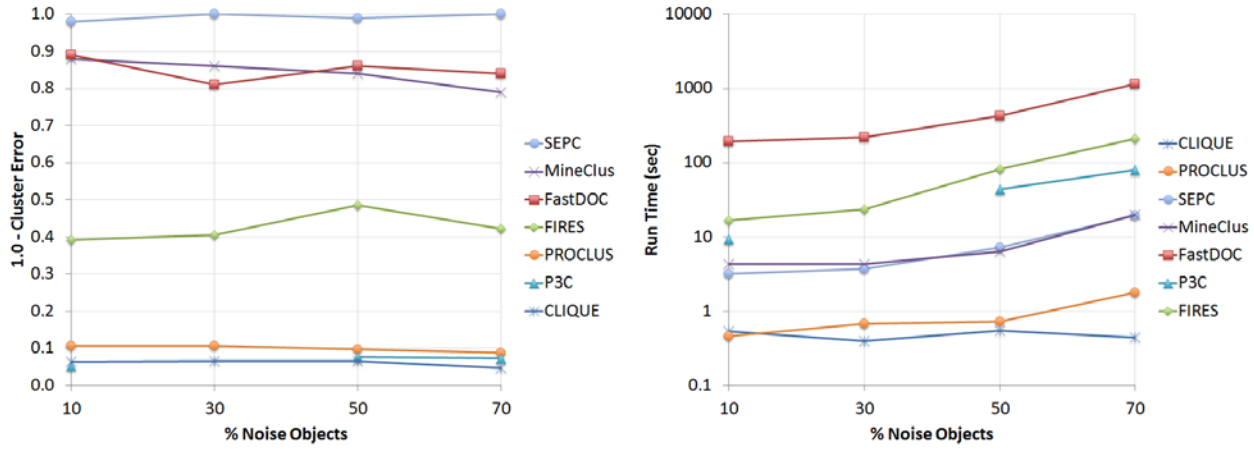
As in [8], we used CE to examine the relative quality of the clustering results generated on these synthetic data by each algorithm. Since we have information about the relevant subspaces of the hidden clusters in the synthetic data; we can fully leverage the power of CE. Recall that the CE metric not only indicates that objects have been correctly assigned to clusters, but also penalizes redundancy (e.g. multiple found clusters covering the same hidden cluster) and splitting hidden clusters (many small found clusters covering the objects of a single hidden cluster). This also allows us to evaluate an algorithm's ability to discover clusters with overlapping objects. Since each of the synthetic data sets include overlapping clusters, algorithms that perform a strict partitioning of objects into clusters will not be able to score a perfect CE. For example, the maximum partitioned CE score for the object-count data sets is about 0.88. Recall that CE is determined using subobjects (objects combined with the subspace of the cluster to which it belongs). This means that according to the CE metric, an object may be assigned to multiple clusters, as long as those clusters span different subspaces. This also means that there can be more subobjects than objects in a data set. For example, the 1500-object data set has 1462 objects in its 10 hidden clusters. However, since some of these objects belong to more than one cluster, the data set contains 1663 subobjects. To determine the maximum possible CE results with single object assignment, we simply divide the number of objects by the number of subobjects, which yields approximately 0.88. Therefore, a CE score exceeding 0.88 on the 1500-object data set would indicate that the algorithm was successfully discovering overlapping clusters. The maximum disjoint CE value varies between the synthetic data sets. It is about 0.88 for all of the object-count and noise data sets, and about 0.8 for the dimension-count data sets.

We also examined the running times for each algorithm for each data set. For some algorithms, parameter settings may significantly affect running time. In such cases, we used consistent parameter settings to gather run time data even if it did not yield optimal CE results. In addition, some of the algorithms never completed within 30 minutes on some of the data sets for any parameter settings. For example, P3C did not finish within 30 minutes on any of the dimension data sets



(a) (b)

Fig. 4. Algorithm performance over an increasing number of objects measured by (a) clustering error and (b) run time.



(a) (b)

Fig. 5. Algorithm performance over increasing noise measured by (a) clustering error and (b) run time.

above 20 dimensions. This accounts for the missing data in Fig 3 (b), 4 (b) and 5 (b).

To evaluate the scalability of algorithms as the dimensionality of a data set increases, OpenSubspace includes data sets with dimensions varying from 5 to 75. Each data set includes ten subspace clusters that span 50%, 60%, and 80% of the full feature space. Fig. 3 shows the results of our evaluations of each algorithm on the dimension-based data sets. Our evaluation agreed closely with [8], in which Mueller and his team observed the best CE results for the cell-based approaches—particularly DOC and MineClus. In our evaluation, DOC and MineClus scored a CE value of approximately 0.8 across all dimensionalities. However, as can be seen in Fig 3 (a), SEPC exceeded these results for dimensionality of 10 or greater. At dimensionality 5, SEPC performs about as well as DOC or MineClus. However, as dimensionality increases, the CE score achieved by SEPC improves.

OpenSubspace also includes a set of synthetic data where the number of objects in each cluster varies, but the number of dimensions is constant. All of these data sets contain 20-dimensional objects, but they vary in size from about 1500 points up to about 5500 points. We used these data sets to evaluate algorithm performance over increasing data set size. The best results for DOC and MineClus varied between CE values of about 0.85 and 0.9. SEPC exceeded these results with a CE value of at least 0.94 (on the data set containing 4500 data points) and achieved a CE value of 1.0 for the data set containing 3500 data points. See Fig. 4 (a).

For noise-based experiments, OpenSubspace includes data sets where the percentage of noise objects increases from 10% noise up to 70% noise. These data sets were built by adding noise to the 20-dimensional data set from the first scalability experiments. For noise-based experiments, Mueller et al. reported CE results for DOC and MineClus of about 0.79 to 0.89. We saw similar results in our evaluation. We can see in Fig. 5 that the DOC and MineClus results exhibit a slight downward trend as the amount of noise in the data set increases. In contrast, the CE results for SEPC are consistent ranging from 0.95 to 0.97, with no degradation in performance with increasing amounts of noise.

Running time is an important factor to consider when evaluating subspace clustering algorithms. In addition to the CE data, we collected run time data on all of the synthetic data sets to compare SEPC to the other algorithms. See Fig. 3 (b), Fig. 4 (b), and Fig. 5 (b). From these graphs of run time data, we see that SEPC is significantly faster than DOC on all data sets. We also see that MineClus and SEPC have similar run times across all data sets. CLIQUE and PROCLUS are the fastest algorithms across all data sets. However, they also score among the lowest CE values.

SEPC consistently performed better than the other algorithms on each of the synthetic data sets with respect to the CE metric. Recall that the maximum disjoint CE scores for the object and noise data sets was 0.88 and 0.8 for the dimension data sets. SEPC scored CE results near 1.0 on all of these data sets (with the exception of the 5-dimensional data set). The high CE scores achieved by SEPC across all data sets indicate that it effectively discovered overlapping clusters. Overall, it appears SEPC's performance increases with scale. This was illustrated in the experiments with the dimensionality data sets. At 5 dimensions, SEPC performed as well as DOC and MineClus, but as the dimensionality of the data sets increased, SEPC's performance also increased. It appears that for very low dimensional data (less than 5), SEPC is comparable in performance to DOC and MineClus. However, the performance of DOC and MineClus stays the same for large dimensional data, while SEPC improves. SEPC runs much faster than DOC and in similar time to MineClus.

6 Real World Data

In addition to synthetic data, we used the real world data packaged with OpenSubspace to evaluate SEPC against other subspace clustering algorithms. These publicly available data sets from the UCI archive [14] have typically been used in classification tasks and thus the data have class labels. The class labels are assumed to describe natural clusters in the

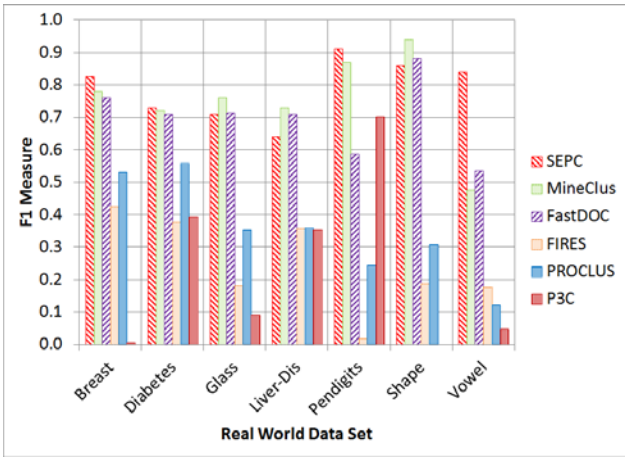


Fig. 6. F1 results with real world data sets.

data. However, no information about the subspaces of the clusters is known. This limits the usefulness of CE, since it can only be applied at the object level. Thus, we have followed the lead of [8] and optimized each algorithm with respect to F1. Also, since all of the clusters in the real world data sets are disjoint, SEPC was run in disjoint mode.

SEPC was compared with MineClus, DOC, PROCLUS, FIRES, and P3C. See Fig. 6 for a chart summarizing the F1 results obtained by each algorithm for each of the seven real world data sets. SEPC yielded the highest F1 score on four out of the seven data sets.

7 Conclusions

SEPC outperformed all other algorithms on synthetic data in terms of clustering quality measured by CE. The high CE scores achieved on the synthetic data sets show that the algorithm effectively identifies clusters even when they share objects. Thus, demonstrating SEPC can be rightly called a subspace clustering algorithm.

The experiments using the synthetic data sets reveal some possible areas where differences in algorithm performance might be more visible. For example, both DOC and MINECLUS, exhibit steady CE results of about 0.8 over an increasing number of dimensions, while the CE results for SEPC started at about 0.8, then increased to values closer to 1.0. Similar results were observed for data containing significant amounts of random noise. Experiments with larger data sets (both in the number of objects and in the number of dimensions), as well as with noisier data, would likely yield more interesting comparisons of performance between algorithms. We also demonstrated that SEPC can be used to achieve high quality results on real world data.

8 References

[1] C. Aggarwal, A. Hinneburg, and D. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Space," *Database Theory—icdt 2001*, pp. 420–434, 2001.

[2] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemom. Intell. Lab. Syst.*, vol. 2, no. 1, pp. 37–52, 1987.

[3] C. F. Olson and H. J. Lyons, "Simple and Efficient Projective Clustering," *Proc. Int. Conf. Knowl. Discov. Inf. Retr.*, pp. 45–55, Oct. 2010.

[4] E. Müller, I. Assent, S. Günnemann, P. Gerwert, M. Hannen, T. Jansen, and T. Seidl, "A Framework for Evaluation and Exploration of Clustering Algorithms in Subspaces of High Dimensional Databases," 2011.

[5] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali, "A Monte Carlo Algorithm for Fast Projective Clustering," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, 2002, pp. 418–427.

[6] M. L. Yiu and N. Mamoulis, "Frequent-Pattern Based Iterative Projected Clustering," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, 2003, pp. 689–692.

[7] A. Patrikainen and M. Meila, "Comparing Subspace Clusterings," *Knowl. Data Eng. Ieee Trans.*, vol. 18, no. 7, pp. 902–916, 2006.

[8] E. Müller, S. Günnemann, I. Assent, and T. Seidl, "Evaluating Clustering in Subspace Projections of High Dimensional Data," *Proc. Vldb Endow.*, vol. 2, no. 1, pp. 1270–1281, 2009.

[9] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," in *Proceedings ACM SIGMOD International Conference on Management of Data*, Seattle, WA, 1998, vol. 27.

[10] H. P. Kriegel, P. Kroger, M. Renz, and S. Wurst, "A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data," in *Data Mining, Fifth IEEE International Conference on*, 2005, p. 8–pp.

[11] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast Algorithms for Projected Clustering," *Acm Sigmod Rec.*, vol. 28, no. 2, pp. 61–72, 1999.

[12] G. Moise, J. Sander, and M. Ester, "P3C: A Robust Projected Clustering Algorithm," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*, 2006, pp. 414–425.

[13] G. Moise and J. Sander, "Finding Non-Redundant, Statistically Significant Regions in High Dimensional Data: A Novel Approach to Projected and Subspace Clustering," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 533–541.

[14] "UCI Machine Learning Repository." [Online]. Available: http://archive.ics.uci.edu/ml/. [Accessed: 05-Mar-2013].