Cluster Pruning: Finding a Better Cluster Representative Object by Dimension Reduction

Amy Wagaman *

Abstract
Cluster analysis is a significant research area with many applications. While new clustering methods and cluster validation have been a focus of substantial research work, extracting a good representative cluster object appears to have received less attention. In this article, we propose a method to prune clusters obtained from any non-fuzzy clustering method, eliminating unusual cluster members in order to obtain a better representative object from the cluster. The method uses dimension reduction to identify the unusual cluster objects which are then removed. We demonstrate the method via simulations and with applications. One application is extracting protein potential native structures after clustering frames from a molecular dynamics simulation. In this application, averaging frames in a cluster to produce a representative frame could result in a nonsensical protein structure, so extracting a representative frame is important.

Key Words: clustering, dimension reduction, multidimensional scaling, representative object

1. Introduction

The statistical community has invested significant research time in the area of clustering, dividing a set of objects into groups based on object similarities. A variety of algorithms for performing clustering exist, divided into several types: hierarchical, partitioning, density-based, model-based, and graph-based (1). Researchers in many fields use clustering methods. Clustering algorithms may be fuzzy (allowing objects to be assigned to multiple clusters based on computed probabilities) or non-fuzzy, and may be able to determine that some observations are outliers (5) (8) . Researchers have tackled many aspects of the clustering problem, developing new clustering methods, extending methods to deal with large data sets (especially data sets of images) (8) (18), examining ways to determine the number of clusters to search for, and determining ways to validate cluster solutions (13) (14). In many applications, especially when looking at clusters of (visual) graphs or images, cluster "representatives" are chosen from each cluster to enable visualizations of the solutions that quickly display the differences between the clusters. There has been some work on finding good representatives using different methods. In this work, we propose a new method to "prune" an existing non-fuzzy clustering solution using dimension reduction methods to remove outliers before extracting a representative object from the cluster. Briefly, we consider relevant literature, focusing our literature review on applications where representative objects are selected from clusters or applications where some objects are removed from the clustering solution.

There are many ways that the term "representative object" arises in the clustering literature. For example, for large data sets, data prototypes that are "representative objects" of the data set may be selected from a large set of observations and used to cluster the data. However, these points are not selected in advance to represent a particular cluster found via a clustering solution, which is what we

*Department of Mathematics, Amherst College, P.O. Box 5000, Amherst, MA 01002

mean by "representative object". An application where clustering is performed in order to select objects to preserve the information in the original data set is provided in (6). In that application, DBSCAN (8) is used to find a clustering solution for a large data set. Then, representative objects are selected from each cluster in proportion to the cluster's size to create a subset of the original data set that can be used for further calculations. To find other examples of objects being used to represent a cluster from a clustering solution, one can easily look at literature on image analysis. When clustering a set of images, it is natural to want a representative image for each cluster. For a simple example, if you returned from a vacation and wanted to put together a scrapbook, you would want to use a representative set of images. Clustering to find groups of similar images and then extracting one from each cluster to use as a representative image makes sense. Several researchers have produced work in this area of image analysis, and have come up with several ways of obtaining (or constructing) cluster representatives (4) (27).

The classic representative objects for clusters that are considered for comparison with new methods are the centroid and medoid object. The centroid is the average of all observations for a cluster, and as a result, it does not have to be an actual data point. The medoid object is the data point that has the smallest average distance to every other data point within its cluster. For examples where new cluster representatives are proposed, the reader may refer to (3) (4) (7) (16) (27). Only one of these applications addresses the question of outliers and their impact on the representative objects examined (7). It is interesting to note that in that application dealing with outliers, the authors work with a true data set of 1024 images, where 100 images are to be selected as representatives, and add outlier images (up to the number of true images). So, at the maximal number of data points, over 2000 images (half true and half outlier), they are still selecting one representative for every 20 images. In our motivating application of selecting representative frames from a protein molecular dynamics simulation, we have thousands of data points and wish to select very few representative objects.

The apparent disconnect between proposing representative objects and not considering the issue of outliers occurs despite the fact that cluster analysis and outlier detection are closely related problems (22) (23). Statisticians have undertaken work to remove or "trim" outliers and to develop methods to do this even as a part of exploratory analysis or as part of a clustering solution. For example, Garcia-Escudero and co-authors (11) propose some trimming methods in exploratory data analysis based on a trimmed k-means framework (5). However, the restrictions of being based on a k-means approach could cause issues with non-spherical clusters. Extensions of these techniques led to development of other models, including the spurious-outlier model for clustering analysis (10) where a proportion of outliers are assumed to be present. This model was further developed by Garcia-Escudero and co-authors in (12) to deal with more heterogeneous clusters. In these applications, however, the problem of selecting a representative object is not addressed.

We want to consider methods that can "prune" unusual objects from clusters from an existing non-fuzzy clustering solution whether or not that technique removes outliers itself. In our examples we use classic clustering methods which do not treat with the outlier problem. We tackle the problem by considering that clusters should be groups of similar objects, and use dimension reduction methods to examine where the clusters are tightly packed and where they are not, setting criteria related to the density of cluster objects in the projected space that objects must meet to be retained in the cluster. The idea of using projection methods to examine clustering

solutions is not new, but using it in this particular way is novel. For example, Pena and Prieto (19) examine applications of univariate projection methods to find clustering solutions. We use projection methods to examine a clustering solution and find objects that should be removed before a representative object is extracted. We examine the success of our method both in terms of detecting and removing outliers and the impact their subsequent removal has on attaining good classical cluster representatives. Thinking in a DBSCAN-style framework, we project the clusters down to a low-dimensional space, and then attempt to remove points that are not "core" points (8). While this may seem counter to DBSCAN's approach (which relies on the concept of directly density reachable points to retain border points), recall that we are interested in extracting cluster representatives, and hence, we are making an assumption that border points will not be the best representatives, so it is appropriate to remove them as well as traditional outliers. Future extensions of the method may attempt to adapt the DBSCAN approach, but we want to analyze this approach first.

In this article, we propose a method to "prune" clusters obtained from any non-fuzzy clustering method, eliminating outliers and slightly unusual cluster members in order to obtain a better classical representative object from the cluster. We propose our new method in Section 2. Then we introduce simulations and results in Section 3. Finally, we conclude with some discussion and future work in Section 4.

2. Proposed Method for Cluster Pruning

The main idea we use in pruning and trying to improve the selection of classical representative objects is that a projection of the cluster into lower-dimensional space that is distance-preserving should put objects in the cluster very near the other objects in the cluster because the cluster is designed to be a collection of similar objects. The projections are obtained via dimension reduction methods, and then projected points are examined to see if they meet a density-based criteria that we set in the projected space. The dimension reduction method we use is multidimensional scaling (MDS). MDS is solved by minimizing an optimization problem with a stress criterion (2). Many other dimension reduction methods could be used to obtain the projections. For example, nonlinear MDS may be used, which only preserves ranking not distance values, or methods such as the ISOMAP (26) or local linear embedding (LLE) (24). However, MDS has fewer tuning parameters than most of these methods (one just has to choose the dimension d of the projection), and seems to work well for our purposes. Other manifold projection algorithms may produce similar results.

Our algorithm has two inputs and several tuning parameters that can be adjusted. The inputs come from an existing clustering analysis. A cluster allocation vector (the assignment of observations to clusters from a clustering algorithm) as well as the data or distance matrix (pairwise distances) is needed. For our algorithm, the distance matrix does not have to match what was used for the clustering, but it should reflect how the points should be compared for pruning (perhaps some covariates are left out). For tuning parameters, either two or three values are needed. For each application of the algorithm, the dimension d is needed for the MDS projection to low-dimensional space. The other two possible tuning parameters deal with the pruning process itself. We call these tuning parameters *trim* and *NN*. Trim (trim factor) is either a multiplication factor (for projections to dimensions greater than

1) or the percentage of points you want to remove (for dimension equal to 1). NN is the number of neighbors that need to occur in the neighborhood of points in the low-dimensional space for the point to be retained in the cluster (only used for dimensions greater than 1). The clusters are pruned using the inputs and supplied tuning parameter values.

The algorithm loops through all clusters from the cluster allocation vector, one at a time. For each cluster, the distance matrix for the subset of observations in the cluster is extracted and the points projected using MDS onto the dimension specified. For projections to 1D, the trim factor is used to prune an even percentage of points from each end of the projection (this can be modified to be uneven, depending on the distribution of coordinates). For example, by default, in 1D, a trim factor of 10 would prune the top 5 percent of the observations with high coordinate values and bottom 5 percent with low coordinate values based on their coordinates in the projected space. For projections to 2D or higher, the distance matrix for the subset of observations in the cluster is computed, the points projected down, and the standard deviation (denoted SD)of all the pairwise distances for points within the cluster is computed. The trim factor is multiplied by SD, and an "epsilon" ball with radius equal to the trim factor times the SD is placed around each point in the low-D space, defining a neighborhood around each point. Points must have NN neighbors within their neighborhoods or they are pruned out of the cluster. The algorithm returns the subset of the original cluster that was retained after pruning, so that analysis may continue. Alternatively, the indices of the observations that were pruned can be obtained. In this way, outliers in the individual clusters can be identified. We explored projections to $d = 1, 2$, and 3, but higher values are possible. Many other variants on the algorithm are possible and are discussed in future work below.

3. Simulations and Results

In order to demonstrate the improvement in obtaining representative cluster centers, we performed simulations comparing the cluster representative objects obtained from several clustering methods to the representative objects obtained after applying our pruning algorithm from generated cluster data. In this section, we describe the simulations and our results. Three main sets of simulations were used to study the performance of the pruning algorithm. The results discussed here are the results of only one set of simulations. (Details on the other simulation sets will be available in a future publication). The simulations were comprehensive, and designed to study the effectiveness of the algorithm at removing outliers and aiding in the extraction of classical representative objects, as well as the impact of choice of tuning parameters (dimension, NN, and trim settings), and even the sensitivity of the algorithm to: choice of number of clusters to find compared to the truth, number of variables (noisy and nonnoisy), size of clusters, separation of clusters, and most importantly, fraction of outliers present. We also share a short toy example with visuals to see how the pruning algorithm works.

3.1 Clustering Algorithms Used in Simulations

In order to demonstrate our cluster pruning algorithm, we need to provide (as inputs) solutions from clustering algorithms. In our simulations, we used three different clustering algorithms which each have pros and cons. These algorithms are

traditional K-means, PAM (robust K-means using medoids), and affinity propagation (AP). All algorithms were implemented in R with necessary packages (25). For affinity propagation, the function apclusterK was used to obtain the number k clusters specified for that simulation.

One of the oldest and most traditional clustering methods, K-means is well studied and easily implemented in any statistics package. K-means seeks to split the data points into k clusters (k must be specified in advance) such that the sum of the squared distances of the points to their assigned cluster centers is minimal. The algorithm is iterative, and starting cluster centers must be supplied or randomly determined. Using several random starts is suggested. For additional insights into the K-means algorithm, see (15). PAM is a more robust version of K-means, where the data points are used to provide the starting cluster centers. In K-means, the starting cluster centers do not have to be existing data points. PAM tries to find representative objects (existing data points) in the cluster, and then assigns points to the nearest representative object to create clusters. For more details on the PAM algorithm, see (17). Finally, affinity propagation is a relatively new technique that obtains clusters by considering the clustering problem as a graph partitioning problem. The method is iterative and does not require that a number of clusters be specified to be found. Instead, a starting tuning parameter is supplied based on the dissimilarities of the data points. Variants of the algorithm exist that allow a specified number of clusters to be found by searching through values of the starting tuning parameter. For more information on affinity propagation, see (9).

3.2 Data Generation for Simulations

In order to generate appropriately clustered data for simulations, we used the clusterGeneration R package (20) (21). This R package was written to generate random clusters, and the genRandomClust function is able to create clusters of many sizes and shapes, and has a parameter that governs the level of cluster separation. The genRandomClust function has a variety of inputs useful for testing new procedures allowing the user to specify the number of clusters, separation index for clusters, number of noisy and non-noisy variables, fraction of outliers, and providing several ways to specify cluster sizes. For cluster size, we chose to supply an interval of values from which each cluster size was randomly drawn. Generated data sets have the true cluster labels available. Outliers are added but are not assigned to a cluster by the generation code.

For our toy example, we set sepValInput (which governs the cluster separation) at 0.1, making the clusters more separated than in the other simulations reported here. The fraction of outliers was set at 20 percent. As a toy example for visualization, we only had two noisy and two nonnoisy variables, and we view the clusters from the nonnoisy variable space for illustration. Four clusters were generated, and cluster sizes ranged from 50 to 200 objects.

For the main simulation set described here, our data generation settings were as follows. We set sepValInput (which governs the cluster separation) at 0.01 or -0.1. The negative value results in clusters that are less separated than when sepValInput is set at 0.01. We refer to the .01 setting as medium separation and -0.1 setting as minor separation for easier reference (though those adjectives are subjective). The fraction of outliers was set at either 10 or 20 percent. Five clusters were requested with sizes ranging from a minimum of 100 to maximum of 300 objects (cluster size was randomly drawn from that interval). For number of variables, two settings were

used - 4 nonnoisy and 2 noisy variables or 5 nonnoisy and 8 noisy variables.

For each data set generated, the three different clustering algorithms (K-means, PAM, and affinity propagation) were applied to obtain cluster solutions. In addition, in different runs of the simulation, we told the algorithms to find four, five, or six clusters (remember that the data was generated with five clusters), and thus, only one-third of the runs were trying to find the correct number of clusters. For each clustering solution, the clusters found were compared to the true clusters. The centroids and medoids of the clusters were compared due to our focus on improving cluster representative objects. For each cluster, the cluster centroid and medoid distances from the true cluster centroids and medoids were computed, and the found clusters were matched with the true cluster that their centers were closest to. The number of the matching cluster and distance between the true and found centroid or medoid were reported. Note that even if the algorithms found five clusters, this does not mean all true clusters would be captured. For the toy example, the algorithms were told to search for six clusters, where four is the correct number of clusters.

3.3 Pruning Settings

For this set of simulations, we examined the following pruning algorithm choices. For the dimension of the MDS projection, we tested values of 1, 2, and 3. For the trim factor, we used 10 and 20 percent for the one-dimensional projections, and .5 and 1 as the multipliers to determine the size of the neighborhoods around each point in the projected space for dimensions two and three. These are referred to as trim levels 1 and 2 respectively in the results below. We fixed NN at 3, based on previous simulation work. Ten data sets were analyzed at each combination of simulation settings. In the simulations not reported here, the fraction of outliers was pushed to 30 percent, number of clusters to 15, number of variables past 20 (with more than half noisy) with 100 data sets at each combination of settings. In addition, we assessed the impact of choice of NN on the results. For the toy example, we used a dimension of two, NN=3, and trim=.5 for pruning.

3.4 Toy Example

Our toy example with only 4 clusters and 4 variables (2 noisy and 2 nonnoisy) serves to demonstrate how well the pruning algorithms can remove outliers. Figure 1 shows the true cluster labels, K-means solution, and K-means pruned solution in the nonnoisy variable space. The PAM and AP solutions are not shown but are very similar. In the Figure, you can see the true clusters are somewhat separated in the nonnoisy variable space. Outliers appear as black circles in the true cluster plot. In the K-means solution, due to asking for six clusters, and the presence of noisy variables, the true clusters are not exactly picked up (most are split in two, at least). Additionally one cluster is identified (red triangles pointing up) that actually consists of points that are mostly outliers. In the K-means pruned solution, we see the outlier cluster has basically been removed, and other outliers cleaned up largely as well. Some real data points have also been removed (an inevitable tradeoff). The solution is clearly improved, but applying pruning does not rectify the issue with true clusters being split though that is not the purpose of this methodology. In the toy example, there were 87 outliers total added to four clusters (sizes 80, 109, 143, and 103, respectively) with combined size of 435 real points. In the K-means pruned solution, 77 percent of outliers were removed, while only 7.4 percent of real data points were removed. In the PAM pruned solution, 73.6 percent of outliers were

removed, while 9.4 percent of real data points were removed. Finally, in the affinity propagation pruned solution, 78.2 percent of outliers were removed, and only 6.4 percent of real data points were removed.

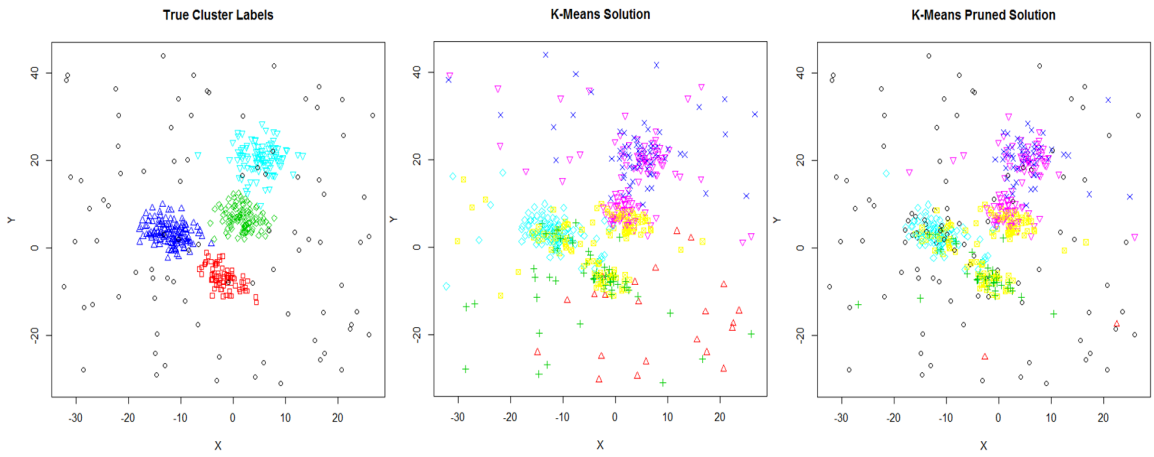

Figure 1: Toy Example with Four Clusters showing True Cluster Allocations, K-means Solution, and K-means Pruned Solution in the 2D nonnoisy variable space. Outliers are shown as black circles in the true allocation. Pruned points are shown as black circles in the pruned solution.

3.5 Simulation Results

Next, we turn to our results for the main simulations described above. Tables 1 and 2 display the average fraction of outliers versus true points removed over all 72 simulations for their respective cluster separation values. It is clear that the pruning method performs about equally well even with different levels of cluster separation, and removed over 50 percent of outliers on average while not removing many data points attributed to clusters.

Table 1: Outlier and True Point Removal Rates in the Medium Separation Setting. These are the average fractions of points removed over all simulations at the medium separation setting.

Clustering Method	Avg. Outliers Removed	Avg. True Points Removed
K-means	.5938	.0703
PAM	.6252	.0729
AP	.6239	.0721

We next consider results when broken down by dimension and fraction of outliers added. Results are similar for both the medium and minor separation cases, so we show results for the harder case of minor separation here. From Tables 3 and 4 we can see that projecting to three dimensions had the best results for removing outliers, but also resulted in the most real data loss. Similarly, when projecting to two dimensions, fewer outliers and fewer real points were removed. When looking at the results broken down by fraction of outliers, we see a similar tradeoff. We can

Table 2: Outlier and True Point Removal Rates in the Minor Separation Setting. These are the average fractions of points removed over all simulations at the minor separation setting.

Clustering Method	Avg. Outliers Removed	Avg. True Points Removed
K-means	.5953	.0707
PAM	.6277	.0718
AP	.6261	.0715

remove a higher fraction of outliers if we are willing to remove a higher fraction of real data points.

Table 3: Outlier and True Point Removal Rates in the Minor Separation Setting By Dimension. These are the average fractions of points removed over all simulations with that dimension in the minor separation setting.

	Avg. Outliers Removed			Avg. True Points Removed		
Method	Dim=1	Dim=2	Dim=3	Dim=1	Dim=2	Dim=3
K-means	.5553	.4869	.7444	.0912	.0173	.1039
PAM	.5573	.5207	.8059	.0908	.0187	.1060
AP	.5592	.5178	.8022	.0903	.0195	.1050

Table 4: Outlier and True Point Removal Rates in the Minor Separation Setting By Fraction of Outliers Added. These are the average fractions of points removed over all simulations with that fraction of outliers (FO=1 is 10 percent, FO=2 is 20 percent) in the minor separation setting.

	Avg. Outliers Removed		Avg. True Points Removed	
Method	FO=1	FO=2	FO=1	FO=2
K-means	.6195	.5712	.0863	.0552
PAM	.6791	.5765	.0880	.0557
AP	.6770	.5754	.0882	.0550

Finally, we look at results in terms of how far the centroids and medoids moved closer to the true centroids and medoids in Tables 5 and 6 for the minor separation setting for K-means, broken down by dimension, number of neighbors required to keep a point (NN), trim setting and fraction of outliers.

For both centroids and medoids, we see that pruning is improving the K-means solution resulting in representative objects that are closer to the true values. For centroids, the PAM and affinity propagation results are similar to those observed for K-means. There is substantial improvement with pruning as the pruned centroids move towards the true centroids compared to the unpruned centroids. For the medoids, there are fewer improvements for PAM and affinity propagation compared to K-means because there is often no shift in the medoid from the pruned to unpruned version. This is somewhat intuitive. Pruning removes unusual points,

Table 5: Improvements in Centroids on Average in the Minor Separation Setting for K-means. Distance to true centroid is computed before and after pruning and compared (Unpruned distance minus pruned distance). Positive improvements indicate that the pruned centroid was closer to the truth than the unpruned centroid. Average values of improvement are shown with standard deviations in parentheses.

	Dimension=1			Dimension=2			Dimension=3		
Trim/FO	NN=4	NN=5	NN=6	NN=4	NN=5	NN=6	NN=4	NN=5	NN=6
Trim=1	.71	1.00	1.27	1.28	1.61	2.00	1.56	2.13	2.63
FO=1	(.21)	(.33)	(.28)	(.44)	(.41)	(.44)	(.59)	(.56)	(.59)
Trim=2	.94	1.24	1.56	.61	.95	1.29	1.26	1.73	2.10
FO=1	(.23)	(.30)	(.37)	(.26)	(.34)	(.29)	(.26)	(.37)	(.42)
Trim=1	1.18	1.60	1.66	1.86	2.34	3.14	2.72	3.97	4.46
FO=2	(.19)	(.34)	(.36)	(.62)	(.45)	(.60)	(.44)	(.91)	(.97)
Trim=2	1.63	1.97	2.37	.51	1.05	1.36	1.67	2.69	3.06
FO=2	(.37)	(.28)	(.50)	(.22)	(.28)	(.45)	(.44)	(.48)	(.72)

Table 6: Improvements in Medoids on Average in the Minor Separation Setting for K-means. Distance to true medoid is computed before and after pruning and compared (Unpruned distance minus pruned distance). Positive improvements indicate that the pruned medoid was closer to the truth than the unpruned medoid. Average values of improvement are shown with standard deviations in parentheses.

	Dimension=1			Dimension=2			Dimension=3		
Trim/FO	NN=4	NN=5	NN=6	NN=4	NN=5	NN=6	NN=4	NN=5	NN=6
Trim=1	6.66	5.56	5.70	7.09	6.61	6.25	6.07	6.52	4.83
FO=1	(6.17)	(5.02)	(4.87)	(5.37)	(3.46)	(5.83)	(4.95)	(3.96)	(4.57)
Trim=2	6.72	6.84	4.06	6.73	5.91	5.50	5.14	5.72	4.38
FO=1	(5.65)	(4.83)	(3.99)	(3.79)	(4.17)	(4.54)	(5.00)	(4.91)	(3.52)
Trim=1	5.49	4.78	4.73	5.15	4.18	6.37	5.87	5.73	4.57
FO=2	(4.62)	(3.10)	(4.69)	(3.89)	(3.63)	(5.32)	(5.38)	(5.61)	(7.70)
Trim=2	5.53	7.42	5.27	5.91	5.20	6.27	5.59	8.09	4.32
FO=2	(4.91)	(6.53)	(5.45)	(4.97)	(3.59)	(5.00)	(4.16)	(3.20)	(6.01)

and just as in the univariate case, removing outliers will affect the mean more than the median. PAM and affinity propagation both choose "median" type data points as their representative objects and as a result, not much shifting occurs in the medoids due to pruning. However, we do observe that large fractions of the unusual points are being removed, no matter what clustering method is employed. Overall these results suggest that the pruning method is beneficial for removing outliers and improving the "classical" representative objects of the centroid and medoid of clusters.

4. Discussion and Future Work

In conclusion, we have proposed a new methodology that "prunes" clusters from a clustering solution, removing outlying points and improving the classical cluster representative objects of centroids and medoids. The methodology proved effective at removing over 50 percent of outlying points in simulations and improved centroids more than medoids for all clustering methods in the simulations. Medoid improvements occurred but were not as pronounced as the centroid improvements except for the K-means clustering method. Further work investigating uses of the method and adaptations is necessary.

As an application, we consider the motivation for the method. The pruning method was developed for use in clustering protein structures resulting from molecular dynamics simulations aimed at uncovering native protein structures. These simulations result in thousands of structures, from which a few candidate structures for the native state (folded state) of the protein must be selected. In this setting, pruning was able to remove unusual structures from consideration, however, the scientists developed another methodology to examine structures with a graph-based approach. As part of this approach, unusual frames did not contribute many edges to the graphs, and so were dealt with in that way rather than being removed explicitly. The pruning method was applied to a few test cases, but no definitive improvement for solving the protein structure problem was seen, when coupled with the other approach. However, extensions and adaptations of the pruning procedure may still be useful in that application and in other applications where outlier removal is desired.

Many extensions and adaptations of the procedure are possible and may help in different situations. First, the tuning parameters do not need to be held constant for all clusters. The performance of the procedure should be examined when pruning settings are adapted based on cluster sizes. Additionally, the projection does not need to be performed with MDS. Depending on the underlying structure of the clusters, it may be more useful to prune points that are outliers when the projection is completed using ISOMAP (26) or other nonlinear dimensionality reduction methods. Simulations should be undertaken to examine the performance of the method when points are on selected manifolds under different projection mechanisms. As a last adaptation of the procedure, the criteria for "pruning" a point is based on a crude estimate of density in the projected space. Many other choices for setting that criteria exist and could be explored.

In terms of combining the pruning algorithm with clustering methods, our simulations only coupled the technique with non-fuzzy clustering methods that do not treat with the outlier problem themselves. It would be interesting to see how the algorithm works when coupled with density-based clustering algorithms like DBSCAN and clustering algorithms designed for the spurious outlier model that have built-in mechanisms for dealing with outliers.

Finally, we only examined the performance of the method in improving representative objects for the classical cases of centroids and medoids. Our literature review revealed that there are other ways to select representative objects that are useful in different situations. The pruning algorithm may help in the selection of some of these representative objects but not for others (as was the case for centroid vs. medoid). An investigation here may prove useful but we anticipate the results may be heavily application specific.

Acknowledgements: Collaboration with Laufer Center for Quantitative Biology

at Stonybrook University

References

[1] B. Andreopoulos, A. An, X. Wang, and M. Schroeder, *A roadmap of clustering algorithms: Finding a match for a biomedical application*, Briefings in Bioinformatics, 10 (2009), pp. 297–314.

[2] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer, New York, 2005.

[3] S. Brecheisen, H. Kriegel, P. Krger, and M. Pfeifle, *Visually mining through cluster hierarchies*, in Proc. SIAM Int. Conf. on Data Mining, SIAM, 2004, pp. 400–412.

[4] W. Chu and C. Lin, *Automatic selection of representative photo and smart thumbnailing using near-duplicate detection*, in Proceedings of the 16th ACM International Conference on Multimedia, ACM, 2008, pp. 829–832.

[5] J. Cuesta-Albertos, A. Gordaliza, and C. Matran, *Trimmed k-means: An attempt to robustify quantizers*, The Annals of Statistics, 25 (1997), pp. 553–576.

[6] M. Daszykowski, B. Walczak, and D. Massart, *Representative subset selection*, Analytica Chimica Acta, 468 (2002), pp. 91–103.

[7] E. Elhamifar, G. Sapiro, and R. Vidal, *See all by looking at a few: Sparse modeling for finding representative objects*, in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference, IEEE, 2012, pp. 1600–1607.

[8] M. Ester, H. Kriegel, J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, KDD, (1996).

[9] B. Frey and D. Dueck, *Clustering by passing messages between data points*, Science, 315 (2007), pp. 972–976.

[10] M. Gallegos and G. Ritter, *A robust method for cluster analysis*, The Annals of Statistics, 33 (2005), pp. 347–380.

[11] L. Garcia-Escudero, A. Gordaliza, and C. Matran, *Trimming tools in exploratory data analysis*, Journal of Computational and Graphical Statistics, 12 (2003), pp. 434–449.

[12] L. Garcia-Escudero, A. Gordaliza, C. Matran, and A. Mayo-Iscar, *A general trimming approach to robust cluster analysis*, The Annals of Statistics, 36 (2008), pp. 1324–1345.

[13] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, *Cluster validity methods: part i.*, ACM Sigmod Record, 31 (2002), pp. 40–45.

[14] ——, *Clustering validity checking methods: part ii*, ACM Sigmod Record, 31 (2002), pp. 19–27.

[15] J. Hartigan and M. Wong, *A k-means clustering algorithm*, Applied Statistics, 28 (1979), pp. 100–108.

[16] J. Helfman and J. Hollan, *Image representations for accessing and organizing web information*, in Proceedings of the SPIE International Society for Optical Engineering Internet Imaging II Conference, 2001, pp. 91–101.

[17] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley and Sons, Inc., 1990.

[18] R. Ng and J. Han, *Clarans: A method for clustering objects for spatial data mining*, Knowledge and Data Engineering, IEEE Transactions on, 14 (2002), pp. 1003–1016.

[19] D. Pena and F. Prieto, *Cluster identification using projections*, Journal of the American Statistical Association, 96 (2001), pp. 1433–1445.

[20] W.-L. Qiu and H. Joe, *Generation of random clusters with specified degree of separation*, Journal of Classification, 23 (2006), pp. 315–334.

[21] ——, *Separation index and partial membership for clustering*, Computational Statistics and Data Analysis, 50 (2006), pp. 585–603.

[22] D. Rocke and D. Woodruff, *Identification of outliers in multivariate data*, Journal of the American Statistical Association, 86 (1996), pp. 1047–1061.

[23] ——, *A synthesis of outlier detection and cluster identification*. Preprint, 1999.

[24] S. Roweis and L. Saul, *Nonlinear dimensionality reduction by local linear embedding*, Science, 290 (2000), pp. 2323–2326.

[25] R. D. C. Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, 2009.

[26] J. Tenenbaum, V. de Silva, and J. Langford, *A global geometric framework for nonlinear dimensionality reduction*, Science, 290 (2000), pp. 2319–2323.

[27] A. Varde, E. Rundensteiner, C. Ruiz, D. Brown, M. Maniruzzaman, and R. S. Jr., *Effectiveness of domain-specific cluster representatives for graphical plots*, SIGMOD IQIS, (2006), pp. 24–29.