A New Shared Nearest Neighbor Clustering Algorithm and its Applications

Levent Ertöz, Michael Steinbach, Vipin Kumar
{ertoz, steinbac, kumar}@cs.umn.edu
University of Minnesota

Abstract

Clustering depends critically on density and distance (similarity), but these concepts become increasingly more difficult to define as dimensionality increases. In this paper we offer definitions of density and similarity that work well for high dimensional data (actually, for data of any dimensionality). In particular, we use a similarity measure that is based on the number of neighbors that two points share, and define the density of a point as the sum of the similarities of a point's nearest neighbors. We then present a new clustering algorithm that is based on these ideas. This algorithm eliminates noise (low density points) and builds clusters by associating non-noise points with representative or core points (high density points). This approach handles many problems that traditionally plague clustering algorithms, e.g., finding clusters in the presence of noise and outliers and finding clusters in data that has clusters of different shapes, sizes, and density. We have used our clustering algorithm on a variety of high and low dimensional data sets with good results, but in this paper, we present only a couple of examples involving high dimensional data sets: word clustering and time series derived from NASA Earth science data.

1 Introduction

Cluster analysis tries to divide a set of data points into useful or meaningful groups, and has long been used in a wide variety of fields: psychology and other social sciences, biology, statistics, pattern recognition, information retrieval, machine learning, and data mining. Cluster analysis is a challenging task and there are a number of well-known issues associated with it, e.g., finding clusters in data where there are clusters of different shapes, sizes, and density or where the data has lots of noise and outliers. These issues become more important in the context of high dimensionality data sets.

For high dimensional data, traditional clustering techniques have sometimes been used. For example, the K-means algorithm and agglomerative hierarchical clustering techniques [DJ88], have been used extensively for clustering document data. While K-means is efficient and often produces "reasonable" results, in high dimensions, K-means still retains all of its low dimensional limitations, i.e., it has difficulty with outliers and does not do a good job when the clusters in the data are of different sizes, shapes, and densities. Agglomerative hierarchical clustering schemes, which are often thought to be superior to K-means for low-dimensional data, also have problems. For example, the single link approaches are very vulnerable to noise and differences in density. While group average or complete link are not as vulnerable to noise, they have trouble with differing densities and, unlike single link, cannot handle clusters of different shapes and sizes.

Part of the problems with hierarchical clustering approaches arise because of problems with distance in high dimensional space. It is well-known that Euclidean distance does not work well

in high dimensions, and typically, clustering algorithms use distance or similarity measures that work better for high dimensional data e.g., the cosine measure. However, even the use of similarity measures such as the cosine measure does not eliminate all problems with similarity. Specifically, points in high dimensional space often have low similarities and thus, points in different clusters can be closer than points in the same clusters. To illustrate, in several TREC datasets (which have class labels) that we investigated [SKK00], we found that 15-20% of a points nearest neighbors were of a different class. Our approach to similarity in high dimensions first uses a *k* nearest neighbor list computed using the original similarity measure, but then defines a new similarity measure which is based on the number of nearest neighbors shared by two points.

For low to medium dimensional data, density based algorithms such as DBSCAN [EKSX96], CLIQUE [AGGR98], MAFIA [GHC99], and DENCLUE [HK98] have shown to find clusters of different sizes and shapes, although not of different densities. However, in high dimensions, the notion of density is perhaps even more troublesome than that of distance. In particular, the traditional Euclidean notion of density, the number of points per unit volume, becomes meaningless in high dimensions. In what follows, we will define the density at a data point as the sum of the similarities of a point's nearest neighbors. In some ways, this approach is similar to the probability density approach taken by nearest neighbor multivariate density estimation schemes, which are based on the idea that points in regions of high probability density tend to have a lot of close neighbors.

While "better" notions of distance and density are key ideas in our clustering algorithm, we will also employ some additional concepts which were embodied in three recently proposed clustering algorithms, i.e., CURE [GRS98], Chameleon [KHK99], and DBSCAN. Although, the approaches of these algorithms do not extend easily to high dimensional data, they algorithms outperform traditional clustering algorithms on low dimensional data, and have useful ideas to offer. In particular, DBSCAN and CURE have the idea of "representative" or "core" points, and, although our definition is somewhat different from both, growing clusters from representative points is a key part of our approach. Chameleon relies on a graph based approach and the notion that only some of the links between points are useful for forming clustering; we also take a graph viewpoint and eliminate weak links. All three approaches emphasize the importance of dealing with noise and outliers in an effective manner, and noise elimination is another key step in our algorithm.

To give a quick preview, our clustering approach first redefines the similarity between points by looking at the number of nearest neighbors that points share [JP73]. Using this similarity measure, we then define the notion of density based on the sum of the similarities of a point's nearest neighbors. Points with high density become our representative or core points, while points with low density represent noise or outliers and are eliminated. We then find our clusters by finding all groups of points that are strongly similar to representative points.

Any new clustering algorithm must be evaluated with respect to its performance on various data sets, and we present a couple of examples. For the first example, we find clusters in NASA Earth science data, i.e., pressure time series. For this data, our shared nearest neighbor (SNN) approach has found clusters that correspond to well-known climate phenomena, and thus we

have confidence that the clusters we found are "good." Using these clusters as a baseline, we show that the clusters found by Jarvis-Patrick clustering [JP73], an earlier SNN clustering approach, and K-means clustering are not as "good." For the second example, we cluster document terms, showing that our clustering algorithm produces highly coherent sets of terms. We also show that a cluster consisting of a single word can be quite meaningful.

The basic outline of this paper is as follows. Section 2 describes the challenges of clustering high dimensional data: the definition of density and similarity measures, and the problem of finding non-globular clusters. Section 3 describes previous clustering work using the shared nearest neighbor approach, while Section 4 introduces our new clustering algorithm. Section 5 presents a couple of examples: the first example Section finds clusters in NASA Earth science data, i.e., pressure time series, while the second example describes the results of clustering document terms.

2 Challenges of Clustering High Dimensional Data

The ideal input for a clustering algorithm is a dataset, without noise, that has a known number of equal size, equal density, globular. When the data deviates from these properties, it poses different problems for different types of algorithms. While these problems are important for high dimensional data, we also need to be aware of problems which are not necessarily important in two dimensions, such as the difficulties associated with density and measures of similarity and distance. In this section, we take a look at these two problems and also consider the importance of representative points for handling non-globular clusters.

2.1 Behavior of similarity and distance measures in high dimensions

The most common distance metric used in low dimensional datasets is Euclidean distance, or the L_2 norm. While Euclidean distance is useful in low dimensions, it doesn't work as well in high dimensions. Consider the pair of ten-dimensional data points, 1 and 2, shown below, which have binary attributes.

Point	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10
1	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1

If we calculate the Euclidean distance between these two points, we get $\sqrt{2}$. Now, consider the next pair of ten-dimensional points, 3 and 4.

Point	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10
3	1	1	1	1	1	1	1	1	1	0
4	0	1	1	1	1	1	1	1	1	1

If we calculate the distance between point 3 and 4, we again find out that it's $\sqrt{2}$. Notice that points 1 and 2 do not share any common attributes, while points 3 and 4 are almost identical. Clearly Euclidean distance does not capture the similarity of points with binary attributes. The problem with Euclidean distance is that missing attributes are as important as the present

attributes. However, in high dimensions, *the presence of an attribute is a lot more important than the absence of an attribute*, provided that most of the data points are sparse vectors (not full), and in high dimensions, it is often the case that the data points will be sparse vectors, i.e. they will only have a handful of non-zero attributes (binary or otherwise).

Different measures, such as the cosine measure and Jaccard coefficient, have been suggested to address this problem. The cosine similarity between two data points is equal to the dot product of the two vectors divided by the individual norms of the vectors. (If the vectors are already normalized the cosine similarity simply becomes the dot product of the vectors.) The Jaccard coefficient between two points is equal to the number of intersecting attributes divided by the number of spanned attributes by the two vectors (if attributes are binary). There is also an extension of Jacquards coefficient to handle non-binary attributes. If we calculate the cosine similarity or Jaccard coefficient between data points 1 and 2, and 3 and 4, we'll see that the similarity between 1 and 2 is equal to zero, but is almost 1 between 3 and 4.

Nonetheless, even though we can clearly see that both of these measures give more importance to the presence of a term than to its absence, there are cases where using such similarity measures still does not eliminate all problems with similarity in high dimensions. We investigated several TREC datasets (which have class labels), and found out that 15-20% of the time, for a data point A, its most similar data point (according to the cosine measure) is of a different class. This problem is also illustrated in [GRS99] using a synthetic market basket dataset.

Note that this problem is ***not*** due to the lack of a good similarity measure. Instead, the problem is that direct similarity in high dimensions cannot be trusted when the similarity between pairs of points are low. In general, data in high dimensions is sparse and the similarity between data points, on the average, is very low.

Another very important problem with similarity measures in high dimensions is that, *the triangle inequality doesn't hold*. Here's an example:

Point	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10
A	1	1	1	1	1	0	0	0	0	0
B	0	0	1	1	1	1	1	1	0	0
C	0	0	0	0	0	1	1	1	1	1

Point A is close to point B, point B is close to point C, and yet, the points A and C are infinitely far apart. The similarity between A and B and C and B comes from different sets of attributes.

2.2 Dealing with Non-globular Clusters using Representative Points

Non-globular cluster cannot be handled by centroid-based schemes, since, by definition, such clusters are not represented by their centroid. Single link methods are most suitable for capturing clusters with non-globular shapes, but these methods are very brittle and cannot handle noise properly. However, representative points are a good way of finding clusters that are not

characterized by their centroid and have been used in several recent clustering algorithms, e.g., CURE and DBSCAN.

In CURE, the concept of representative points is used to find non-globular clusters. The use of representative points allows CURE to find many types of non-globular clusters. However, there are still many types of globular shapes that CURE cannot handle. This is due to the way the CURE algorithm finds representative points, i.e., it finds points along the boundary, and then shrinks those points towards the center of the cluster.

The notion of a representative point is also used in DBSCAN, although the term "core point" is used. In DBSCAN, the density associated with a point is obtained by counting the number of points in a region of specified radius around the point. Points with a density above a specified threshold are classified as core points, while noise points are defined as non-core points that don't have a core points within the specified radius. Noise points are discarded, while clusters are formed around the core points. If two core points are neighbors of each other, then their clusters are joined. Non-noise, non-border points, which are called boundary points, are assigned to the clusters associated with any core point within their radius. Thus, core points form the skeleton of the clusters, while border points flesh out this skeleton.

While DBSCAN can find clusters of arbitrary shapes, it cannot handle data containing clusters of differing densities, since its density based definition of core points cannot identify the core points of varying density clusters. Consider Figure 1. If the user defines the neighborhood of a point by a certain radius and looks for core points that have a pre-defined number of points within that radius, then either the tight left cluster will be picked up as one cluster and the rest will be marked as noise, or else every point will belong to one cluster.

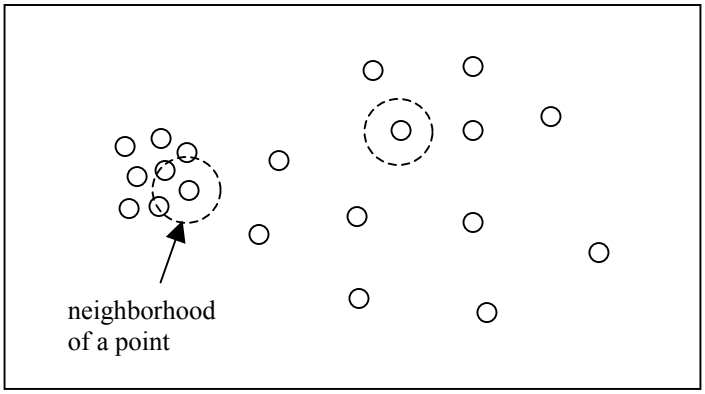

Figure 1. Density Based Neighborhoods

2.1.1 Density in High Dimensional Space

In high dimensional datasets, the traditional Euclidean notion of density, which is the number of points per unit volume, is meaningless. To see this, consider that as the number of dimensions increases, the volume increases rapidly, and unless the number of points grows exponentially with the number of dimensions, the density tends to 0. Thus, in high dimensions, it is not

possible to use a (traditional) density based method such as DBSCAN which identifies core points as points in high density regions and noise points as points in low density regions.

However, there is another notion of density that does not have the same problem, i.e., the notion of the probability density of a point. In the k-nearest neighbor approach to multivariate density estimation [DHS01], if a point that has a lot of close near neighbors, then it is probably in a region which has a relatively high probability density. Thus, when we look at the nearest neighbors of a point, points with a large number of close (highly similar) neighbors are in more "dense" regions than are points with distant (weakly similar) neighbors.

In practice, we take the sum of the similarities of a points nearest neighbors as a measure of this density. The higher this density, the more likely it is that a point is a core or representative points. The lower the density, the more likely, the point is a noise point or an outlier.

3. Shared Nearest Neighbor Based Algorithm

An alternative to a direct similarity is to define the similarity between a pair of points in terms of their shared nearest neighbors. That is, the similarity between two points is "confirmed" by their common (shared) near neighbors. If point A is close to point B and if they are both close to a set of points C then we can say that A and B are close with greater confidence since their similarity is "confirmed" by the points in set C. This idea of shared nearest neighbor was first introduced by Jarvis and Patrick [JP73]. A similar idea was later presented in ROCK [GRS99].

In the Jarvis – Patrick scheme, a shared nearest neighbor graph is constructed from the proximity matrix as follows. A link is created between a pair of points p and q if and only if p and q have each other in their closest k nearest neighbor lists. This process is called k-nearest neighbor sparsification. The weights of the links between two points in the snn graph can either be simply the number of near neighbors the two points share, or one can use a weighted version that takes the ordering of the near neighbors into account. Let i and j be two points. The strength of the link between i and j is now defined as:

$$str(i, j) = \sum (k + 1 - m) * (k + 1 - n), \quad \text{where } i_m = j_n$$

In the equation above, k is the near neighbor list size, m and n are the positions of a shared near neighbor in i and j's lists. At this point, all edges with weights less than a user specified threshold are removed and all the connected components in the resulting graph are our final clusters [JP73].

Figures 2 and 3 illustrate two key properties of the shared nearest neighbor graph in the context of a 2-D point data set. In Figure 2, links to 5 most similar neighbors are drawn for each point. In Figure 3 shows unweighted shared nearest neighbor graph. In the graph, there is a link between points A and B, only if A and B had each other in their near neighbor lists.

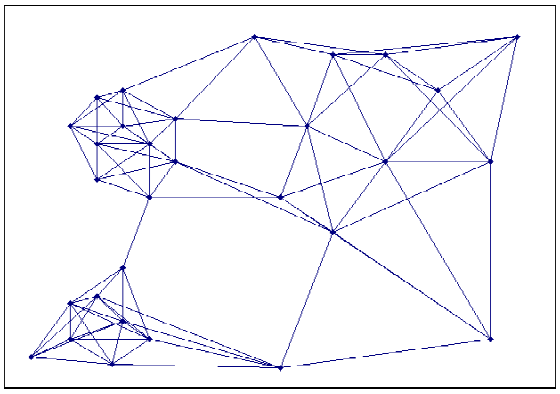

Figure 2. Near Neighbor Graph

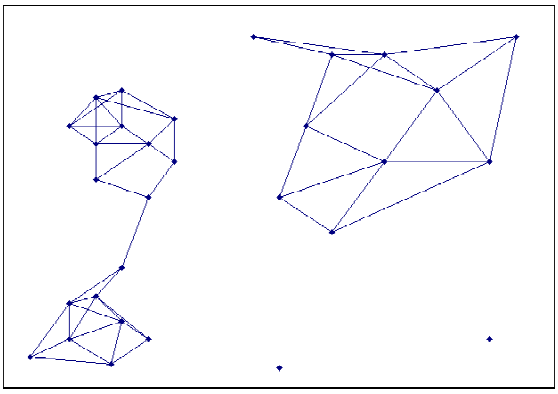

Figure 3. Unweighted Shared Near Neighbor Graph

There are two important issues to note in this 2-D point set example.

First, noise points and outliers end up having most of their links broken if not all. The point on the lower right corner ended up losing its entire links, because it wasn't in the nearest neighbor lists of its own near neighbors. By just looking at the number of surviving links after constructing the snn graph, we can get rid of considerable amount of noise.

Second, shared nearest neighbor graph is "density" independent, i.e. it will keep the links in uniform regions and break the ones in the transition regions. This is an important property, since widely varying tightness of clusters is one of the harder problems for clustering.

A major drawback of the Jarvis – Patrick scheme is that, the threshold needs to be set high enough since two distinct set of points can be merged into same cluster even if there is only one link across them. On the other hand, if the threshold is too high, then a natural cluster may be split into too many small clusters due to natural variations in the similarity within the cluster. As a matter of fact, there may be no right threshold for some data sets. This problem is illustrated in the following example that contains clusters of points sampled from Gaussian distributions of two different means and variance.

In Figure 4, there are two Gaussian samples. (Note that these clusters cannot be correctly separated by k-means due to the different sizes and densities of the samples). Figure 5 shows clusters obtained by Jarvis – Patrick method using the smallest possible threshold (any threshold smaller than this puts all points in the same cluster). Even this smallest possible threshold breaks the data into many different clusters. In Figure 5, different clusters are represented with different shapes and colors, where the discarded points / background points are shown as tiny squares. We can see that, even with a better similarity measure, it is hard to obtain the two apparent clusters.

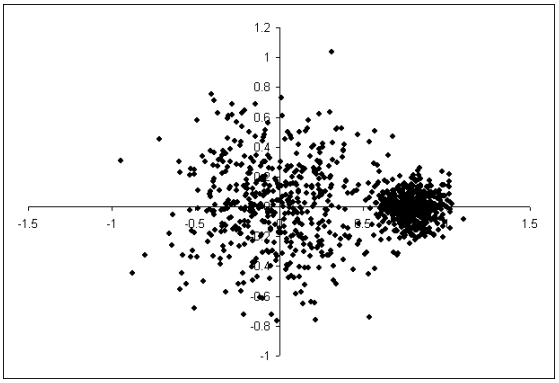

Figure 4. Gaussian Dataset

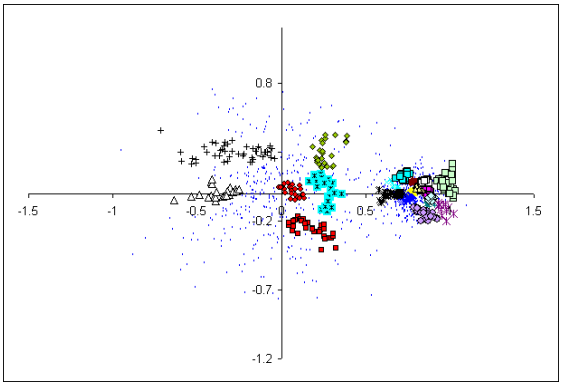

Figure 5. Connected Components - JP Clustering

4. Shared Nearest Neighbor based Clustering Algorithm using Representative Points and Noise Removal

In Section 3 we showed how Jarvis – Patrick method would fail on the gaussian dataset where transition between regions is relatively smooth. In this section, we present an algorithm that builds on Jarvis – Patrick method, and addresses the problems discussed in section 2. This algorithm uses a density based approach to find core / representative points. However, this approach will be based on the notion of density introduced in Section 2.3, which is based on the idea of probability density. However, since we will be using similarity based on a shared nearest neighbor approach, which automatically compensates for different densities (see Section 3), this density approach will not be subject to the same problem illustrated in Figure 1.

4.1 Noise Removal and Detection of Representative Points

Figures 6-9 illustrate how we can find representative points and effectively remove noise using the snn graph. In this 2D point dataset, there are 8000 points. A near neighbor list size of 20 is used. Figure 7 shows all the points that have 15 or more links remaining in the snn graph. In Figure 8, all points have 10-14 links surviving and Figure 9 shows the remaining points. As we can see in these figures, the points that have high connectivity in the snn graph are candidates for representative / core points since they tend to be located well inside the natural cluster, and the points that have low connectivity are candidates for noise points and outliers as they are mostly in the regions surrounding the clusters. Note that all the links in the snn graph are counted to get the number of links that a point has, regardless of the strength of the links. An alternative way of finding representative points is to consider only the strong links in the count. Similarly we can find the representative points and noise points by looking at the sum of link strengths for every point in the snn graph. The points that have high total link strength then become candidates for representative points, while the points that have very low total link strength become candidates for noise points.

Figure 6. Initial Set of Points

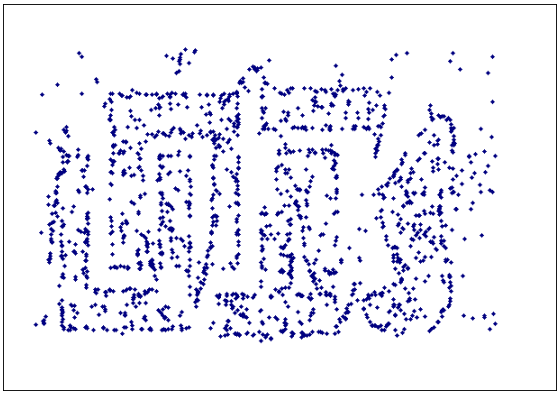

Figure 7. Medium Connectivity Points

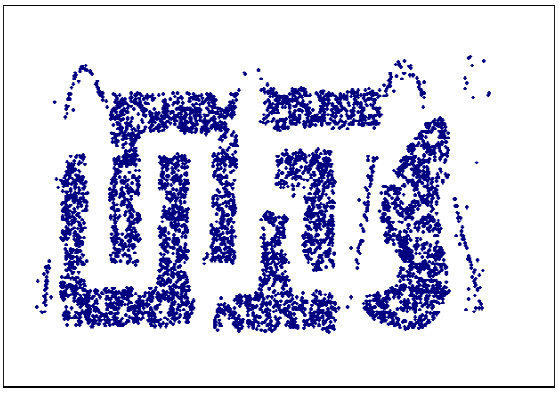

Figure 8. High Connectivity Points

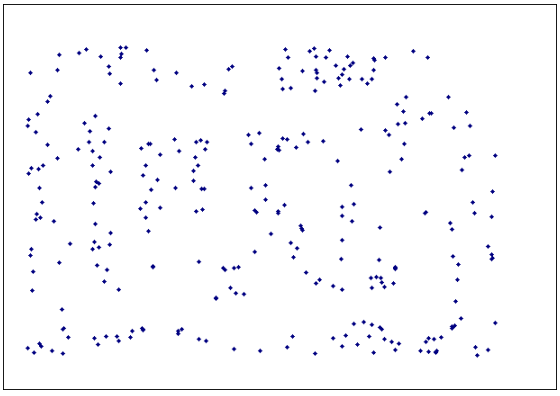

Figure 9. Low Connectivity Points

4.2 The Algorithm

1. Construct the similarity matrix.
2. Sparsify the similarity matrix using k-nn sparsification.
3. Construct the shared nearest neighbor graph from k-nn sparsified similarity matrix.
4. For every point in the graph, calculate the total strength of links coming out of the point. (Steps 1-4 are identical to the Jarvis – Patrick scheme.)
5. Identify representative points by choosing the points that have high total link strength.
6. Identify noise points by choosing the points that have low total link strength and remove them.
7. Remove all links that have weight smaller than a threshold.
8. Take connected components of points to form clusters, where every point in a cluster is either a representative point or is connected to a representative point.

The number of clusters is not given to the algorithm as a parameter. Depending on the nature of the data, the algorithm finds "natural" clusters. Also note that not all the points are clustered using out algorithm. Depending on the application, we might actually want to discard many of the points.

Figure 10 shows the clusters obtained from the Gaussian dataset shown in Figure 4 using the method described here. We can see that by using noise removal and the

representative points, we can obtain the two clusters shown below. The points that do not belong to any of the two clusters can be brought in by assigning them to the cluster that has the closest core point.

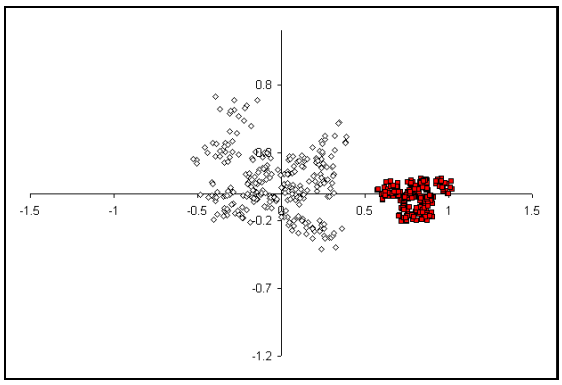

Figure 10. SNN Clustering

5 Applications of SNN clustering

5.1 Earth Science Data

In this section, we consider an application of our SNN clustering technique to Earth science data. In particular, our data consists of monthly measurements of sea level pressure for grid points on a 2.5° longitude-latitude grid (144 horizontal divisions by 72 vertical divisions) from 1950 to 1994, i.e., each time series is a 540 dimensional vector. These time series were preprocessed to remove seasonal variation. For a more complete description of this data and the clustering analysis that we have performed on it, please see [Ste+01] and [Ste+02].

Briefly, Earth scientists are interested in discovering areas of the ocean, whose behavior correlates well to climate events on the Earth's land surface. In terms of pressure, Earth scientists have discovered that the difference in pressure between two points on the Earth's surface often yields a time series that correlates well with certain weather phenomena on the land. Such time series are called Ocean Climate Indices (OCIs). For example, the Southern Oscillation Index (SOI) measures the sea level pressure (SLP) anomalies between Darwin, Australia and Tahiti and is associated with El Nino, the anomalous warming of the eastern tropical region of the Pacific that has been linked to climate phenomena such as droughts in Australia and heavy rainfall along the Eastern coast of South America [Tay98]. Our goal in clustering SLP is to see if the difference of cluster centroids can yield a time series that reproduces known OCIs and to perhaps discover new indices.

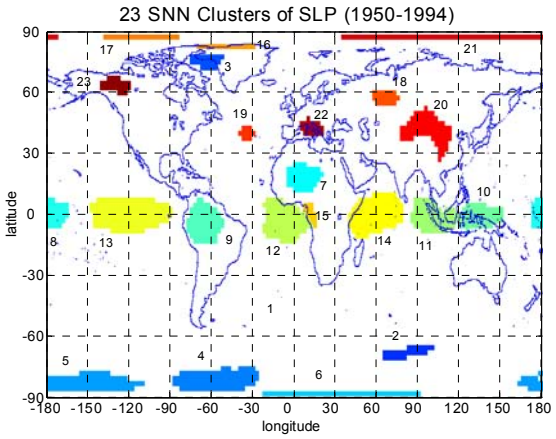

Figure 11. SNN clustering

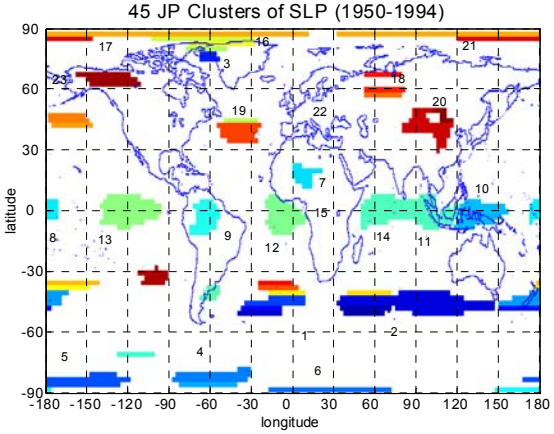

Figure 12. JP clustering with optimal parameters

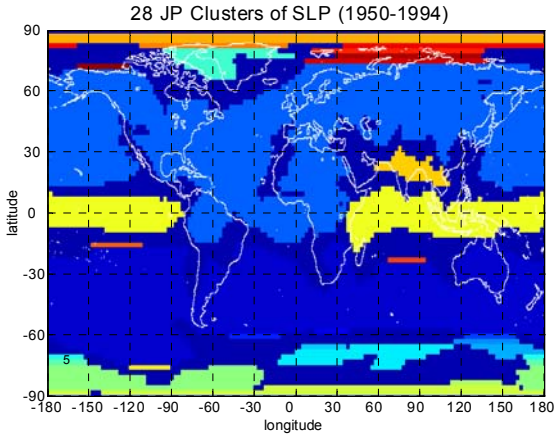

Figure 13. JP clustering with sub-optimal parameters

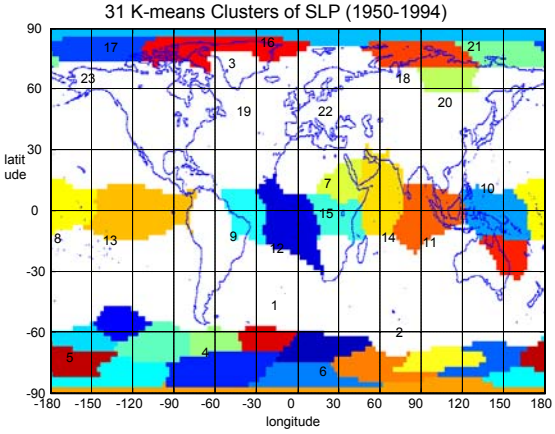

Figure 14. K-means cluster after discarding "loose" clusters.

Our SNN clustering approach yielded the clusters shown in Figure 11. These clusters have been labeled (cluster "1" is the background or "junk" cluster) for easy reference. (Note that we cluster pressure over the entire globe, but here we focus on the ocean.) While we cluster the time series independently of any spatial information, the resulting clusters are typically geographically contiguous, probably because of the underlying spatial autocorrelation of the data.

Using these clusters, we have been able to reproduce SOI as the difference of the centroids of clusters 13 and 10. We have also been able to reproduce another well-known OCI, i.e., NAO, which is the Normalized SLP differences between Ponta Delgada, Azores and Stykkisholmur, Iceland. NAO corresponds to the differences of the centroids of clusters 16 and 19. For more details, see [Ste+02]. This success gives us confidence that the clusters discovered by SNN have real physical significance.

However, it is reasonable to ask whether other clustering techniques could also discover these clusters. To answer that question, we clustered the same data using the Jarvis-

Patrick technique and K-means. The best results for JP clustering are shown in Figure 12, while Figure 13 shows the results for a less optimal parameter choice for JP clustering. Note that the colors have no meaning except to distinguish the different clusters and are not coordinated between any of the different figures. The cluster labels of Figure 11 have been carried forward to Figure 12 to allow for easier comparison. The JP and SNN clustering approaches find roughly the same set of clusters. However, the JP approach tends to break the SNN clusters into a number of smaller clusters. Also, it took a considerable amount of fiddling with the JP parameters to get this figure, while the current implementation of our algorithm produced these clusters on the first try.

For K-means, we attempted to find roughly the same number of clusters. However, since K-means naturally clusters all the data, we generated 100 clusters and then threw out all but the 30 most cohesive clusters (as measured by their average cluster correlation). Still, the clusters shown in Figure 13 are not a very good match for clusters of Figure 11, although they do show the same pattern of clusters around the poles and along the equator. Note that cluster 19, which is key to reproducing the NAO index, is missing from the K-means clusters that we kept. Interestingly, one of the discarded K-means clusters did correlate highly with cluster 19 (corr = 0.95), but it was the least cohesive cluster with respect to average cluster correlation, and thus, the last cluster that we would have considered interesting from a K-means viewpoint.

5.2 Word Clustering

One interesting application comes from the text domain. Much work has been done in document clustering, but relatively little on term clustering. Term clustering (clustering of words) is technically exactly same as document clustering; instead of clustering the rows in the document – term matrix, the columns are clustered. Term clustering can be very useful when trying to identify topics in a given corpus. If we look at the task of clustering words, we will immediately find out that most of the words are generic words that wouldn't give us any clue about the different themes in a given corpus. It is very useful to be able to cluster the words that could give us hints about the different themes. Therefore, we would need to use a clustering algorithm that can handle a serious amount of noise in the data.

On a popular search engine, the query "Beatles" was issued and the words in the resulting 661 snippets were clustered using out algorithm. Note that each snipped contained only one or two lines of text, since we did not download the actual documents. The vocabulary of this snippet set was approximately 1500 words. Every row in

Table 1 represents a cluster obtained from "Beatles" query. Note that the words are in their stemmed form. The snippets can be found at www.cs.umn.edu/~ertoz/word_clusters/.

Here are the highlights of the results. Most of the words in each cluster make sense and seem to belong together. Even the smalles size clusters (e.g., singletons and size two clusters) are meaningful. For example, there was a "pictures" theme in the snippets represented by cluster #2. Similarly, cluster #1 (beatles), and cluster #8 (rock'n roll) make sense. Since these clusters are obtained in the web context, Cluster 3, which talks

about "related files" makes sense. The ability to find small but meaningful clusters is a unique feature of our algorithm. To the best of our knowledge, there is no clustering algorithm that can attach meaning to a singleton cluster or to very small clusters. Note that not all the singletons are clusters. Singleton clusters consist of a word that is a representative point. We can say that the word "pictures" is chosen to be a representative word since there was a coherent neighborhood around it. If we take a look at some of the remaining clusters, cluster #4 contains the members of the band; John Lennon, Paul Mc Cartney, George Harrison and (Ringo) Starr, while cluster #13 contains different albums by the band; Sgt. Pepper's Lonely Hearts Club Band, Hard Day's Night, Magical Mystery Tour, Yellow Submarine, Revolver and White (Album). "Abbey Road", which is another album by the band is in cluster #9.

Table 1. Word Clusters from "beatles" query

1	beatl
2	pictur
3	relat file
4	georg harrison john paul mccartnei lennon starr
5	histori art artist
6	net sale result card postcard
7	fan includ comput perform guid robert
8	rock roll
9	abbei road
10	month event daili januari eight week forget
11	valu christma martin produc
12	trivia jukebox quiz
13	band club dai night tour hard white magic mysteri bbc revolv sgt pepper yellow submarin lone heart

6 Conclusion

In this paper we have introduced a new clustering algorithm which combines a number of ideas to overcome many of the challenges traditionally plague clustering algorithms, e.g., finding clusters in the presence of noise and outliers and finding clusters in data that has clusters of different shapes, sizes, and density. In addition, our clustering approach works well for high dimensional data, where the concepts of distance and density are often ill-defined. To overcome the problems with distance in high dimensionality, we use a distance measure which is based on the number of neighbors that two points share. To handle problems with density, we define the density of a point as the sum of the similarities of a point's nearest neighbors.

The key aspects of our clustering algorithm, besides its use of more effective notions of density and distance, are that it eliminates noise (low density points) and builds clusters by associating non-noise points with representative or core points (high density points). The use of representative points allows us to find arbitrarily shaped clusters, while our

ability to handle noise and outliers make our clustering algorithm tolerant of poor data quality.

Two examples were given that illustrated the performance of our algorithm on real data: documents and NASA Earth science data consisting of time series. In the latter case, we also compared our approach to the Jarvis-Patrick approach and to K-means. While JP clustering produced comparable results, it was hard to parameterize, and in other cases we have observed, impossible to parameterize to yield good clusters.

References

[AGGR98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopoulos, and Prabhakar Raghavan, (1998), Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, Technical Report, IBM Almaden Research Center.

[DHS01] Richard Duda, Peter Hart, and David Stork, (2001), *Pattern Classification*, Wiley-Interscience.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, (1996), *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, pp. 226-231.

[GHC99] Sanjay Goil, Harasha Nagesh, and Alok Choudhary, (1999), *MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets*, Technical Report Number CPDC-TR-9906-019, Center for Parallel and Distributed Computing, Northwestern University, June 1999.

[GK78] K. C. Gowda and G. Krishna, (1978), "Agglomerative Clustering Using the Concept of Mutual Nearest Neighborhood", Pattern Recognition, Vol. 10, pp. 105-112.

[GRS98] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim, (1998*), CURE: An Efficient Clustering Algorithm for Large Databases*, ACM SIGMOD Conference, 1998, pp. 73-84.

[GRS99] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim, (1998*), ROCK: A Robust Clustering Algorithm for Categorical Attributes*, In Proceedings of the 15th International Conference on Data Engineering, 1999.

[HK98] Alexander Hinneburg Daniel. A. Keim and: *An Efficient Approach to Clustering in Large Multimedia Databases with Noise*, Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, AAAI Press.

[JP73] R. A. Jarvis and E. A. Patrick, "Clustering Using a Similarity Measure Based on Shared Nearest Neighbors," IEEE Transactions on Computers, Vol. C-22, No. 11, November, 1973.

[KHK99] George Karypis, Eui-Hong Han, and Vipin Kumar, (1999) *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*, IEEE Computer, Vol. 32, No. 8, August, 1999. pp. 68-75.

[SKK00] Michael Steinbach, George Karypis, and Vipin Kumar, "A Comparison of Document Clustering Algorithms," KDD-2000 Text Mining Workshop, 2000.

[Ste+01] M. Steinbach, P. N. Tan, V. Kumar, C. Potter, S. Klooster, A. Torregrosa, "Clustering Earth Science Data: Goals, Issues and Results", In *Proc. of the Fourth KDD Workshop on Mining Scientific Datasets* (2001).

[Ste+02] M. Steinbach, P. N. Tan, V. Kumar, C. Potter, S. Klooster, "Data Mining for the Discovery of Ocean Climate Indices", Submitted to Workshop on Mining Scientific Datasets, Second SIAM International Conference on Data Mining (2002).

[Tay98] G. H. Taylor, "Impacts of the El Niño/Southern Oscillation on the Pacific Northwest" (1998) http://www.ocs.orst.edu/reports/enso_pnw.html