Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data

Levent Ertöz [†] Michael Steinbach [†] Vipin Kumar [†]

February 20, 2003

Abstract

Finding clusters in data, especially high dimensional data, is challenging when the clusters are of widely differing shapes, sizes, and densities, and when the data contains noise and outliers. We present a novel clustering technique that addresses these issues. Our algorithm first finds the nearest neighbors of each data point and then redefines the similarity between pairs of points in terms of how many nearest neighbors the two points share. Using this definition of similarity, our algorithm identifies core points and then builds clusters around the core points. The use of a shared nearest neighbor definition of similarity alleviates problems with varying densities and high dimensionality, while the use of core points handles problems with shape and size. While our algorithm can find the "dense" clusters that other clustering algorithms find, it also finds clusters that these approaches overlook, i.e., clusters of low or medium density which represent relatively uniform regions "surrounded" by non-uniform or higher density areas. We experimentally show that our algorithm performs better than traditional methods (e.g., K-means, DBSCAN, CURE) on a variety of data sets: KDD Cup '99 network intrusion data, NASA Earth science time series data, and two dimensional point sets. The run-time complexity of our technique is $O(n^2)$ if the similarity matrix has to be constructed. However, we discuss a number of optimizations that allow the algorithm to handle large data sets efficiently.

Keywords: cluster analysis, shared nearest neighbor, time series, network intrusion, spatial data

1 Introduction

Cluster analysis [7, 11] divides data into groups (clusters) for the purposes of summarization or improved understanding. For example, cluster analysis has been used to group related documents for browsing, to find genes and proteins that have similar functionality, or as a means of data compression. While clustering has a long history and a large number of clustering techniques have been developed in statistics, pattern recognition, data mining, and other fields, significant challenges still remain. In part, this is because large, as well as high dimensional, data sets and the computational power to deal with them are relatively recent. However, most of the clustering challenges, particularly those related to "quality," rather than computational resources, are the same challenges that existed decades ago: how to find clusters with differing sizes, shapes and densities, how to handle noise and outliers, and how to determine the number of clusters.

1.1 The Challenges of Cluster Analysis and Related Work K-means [7] is one of the most commonly used clustering algorithm, but it does not perform well on data with outliers or with clusters of different sizes or non-globular shapes. The single link agglomerative clustering method is the most suitable for capturing clusters with non-globular shapes, but this approach is very sensitive to noise and cannot handle clusters of varying density. Other agglomerative clustering algorithms, e.g., complete link and group average, are not as affected by noise, but have a bias towards finding globular clusters. More recently, clustering algorithms have been developed to overcome some of these limitations. In particular, for low dimensional data, DBSCAN [13], CURE [18], and Chameleon [9] have shown good performance.

In DBSCAN, the density associated with a point is obtained by counting the number of points in a region of specified radius, *Eps*, around the point. Points with a density above a specified threshold, *MinPts*, are classified as core points, while noise points are defined as non-core points that don't have a core point within the specified radius. Noise points are discarded, while clusters are formed around the core points. If two core points are within a radius of *Eps* of each other, then

[†]Department of Computer Science
University of Minnesota
{ertoz, steinbac, kumar}@cs.umn.edu

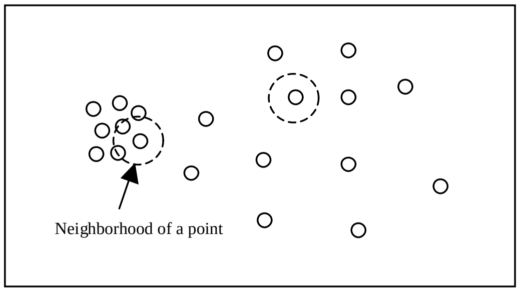


Figure 1: Density based neighborhoods

their clusters are joined. Non-noise, non-core points, which are called border 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 a user defines the neighborhood of a point by specifying a particular radius and looks for core points that have a pre-defined number of points within that radius, then either the tight cluster on the left will be picked up as one cluster, and the rest will be marked as noise, or else every point will belong to a single cluster.

In CURE, the concept of representative points is employed to find non-globular clusters. Specifically, CURE represents a cluster by using multiple representative points from the cluster. These points capture the geometry and shape of the cluster and allow for non-globular clusters. The first representative point is chosen to be the point furthest from the center of the cluster, while the remaining points are chosen so that they are farthest from all the previously chosen points, thus guaranteeing that representative points are well distributed. Once the representative points are chosen, they are shrunk toward the center by a factor, α. This helps to moderate the effect of noise points located at cluster boundaries by moving the representative points belonging to the clusters farther apart.

CURE uses an agglomerative hierarchical scheme to perform the actual clustering. The distance between two clusters is the minimum distance between any two of their representative points (after the representative points are shrunk toward their respective centers). During this hierarchical clustering process, CURE eliminates outliers by eliminating small, slowly growing clusters. Although the concept of representative points does allow CURE to find clusters of different sizes and shapes in some data sets, CURE is still biased towards finding globular clusters, as it still has a notion of a cluster center.

Chameleon is a clustering algorithm that combines an initial partitioning of the data with a novel hierarchical clustering scheme that dynamically models clusters. The first step in Chameleon is to generate a k-nearest neighbor graph. Conceptually, such a graph contains only links between a point and its k-nearest neighbors, i.e., the points to which it is closest or most similar. This local approach reduces the influence of noise and outliers, and provides an automatic adjustment for differences in density. Once a k-nearest neighbor graph has been obtained, an efficient multi-level graph-partitioning algorithm, METIS [10], can be used to partition the data set, resulting in a large number of almost equally sized groups of well-connected vertices (highly similar data points). This produces partitions that are sub-clusters, i.e., that contain points mostly from one "true" cluster.

To recombine the sub-clusters, a novel agglomerative hierarchical algorithm is used. Two clusters are combined if the resulting cluster shares certain properties with the constituent clusters, where the two key properties are the relative closeness and the relative interconnectedness of the points. Thus, two clusters will be merged only if the resulting cluster is "similar" to the original clusters, i.e., if self-similarity is preserved.

Although Chameleon does not explicitly use the notion of core points, all three approaches share the idea that the challenge of finding clusters of different shapes and sizes can be handled by finding points or small subsets of points and then building clusters around them.

While DBSCAN, CURE, and Chameleon have largely been used for solving clustering problems for low dimensional data, high dimensional data brings new challenges. In particular, high dimensionality typically makes its influence felt through its effect on the similarity function. For example, in high dimensional data sets, distances or similarities between points become more uniform, making clustering more difficult. Also, sometimes the similarity between individual data points can be misleading, i.e., a point can be more similar to a point that "actually" belongs to a different cluster than to points in its own cluster. This is discussed in Section 2.

A shared nearest neighbor approach to similarity, as proposed by Jarvis and Patrick in [8], and also later in ROCK [19], is a promising way to deal with this issue. Specifically, the nearest neighbors of each point are found, and then a new similarity between points is defined in terms of the number of neighbors they share. As you will see in sections 3 and 4, this notion of similarity is very valuable for finding clusters of differing densities. While the Jarvis-Patrick approach and ROCK use this notion of similarity, they have serious limitations. In particular, ROCK is similar to the group average agglomerative hierarchical technique mentioned earlier and shares a bias towards globular clusters. Jarvis-Patrick, on the other hand, is similar to single link. It's limitations are discussed further in Section 4.

1.2 Our Contribution We present a clustering approach that can simultaneously address the previously mentioned clustering challenges for a wide variety of data sets. In particular, our algorithm first finds the nearest neighbors of each data point and then, as in the Jarvis-Patrick approach [8], redefines the similarity between pairs of points in terms of how many nearest neighbors the two points share. Using this definition of similarity, our algorithm identifies core points and then builds clusters around the core points. These clusters do not contain all the points, but rather, contain only points that come from regions of relatively uniform density. The use of a shared nearest neighbor definition of similarity removes problems with varying density and the unreliability of distance measures in high dimensions, while the use of core points handles problems with shape and size. Furthermore, the number of clusters is automatically determined.

A novel aspect of our algorithm is that it finds clusters that other approaches overlook. In particular, many clustering algorithms only find "dense" clusters. However, this ignores sets of points that represent relatively uniform regions with respect to their surroundings. Another novel aspect of our approach is that a cluster consisting of a single data point can be significant, since this data point may be representative of a large number of other data points. (Unfortunately , there is not room in this paper to illustrate this idea, and we refer the reader to [4].)

Much of the strength of our approach comes from ideas that are found in several recent clustering algorithms: core or representative points, the importance of noise and outlier removal, and the notion of defining distance in terms of the number of shared nearest neighbors. Our contributions include extending the Jarvis-Patrick shared nearest neighbor clustering technique to encompass the notion of representative points, creating a complete clustering algorithm which incorporates a variety of recent and established ideas, relating this approach to the approaches of other researchers, and importantly, showing that our approach works better than current algorithms for a variety of different types of data.

Note that our algorithm does not cluster all the points. Generally, this is a good thing for many data sets, as often much of the data is noise and should be eliminated. In particular, the ability of our approach to remove noise and outliers is a critical requirement in some of the applications described later. However, if a clustering of all points is desired, then the unclustered data can be added to the core clusters found by our algorithm by assigning them to the cluster containing the closest representative point.

There are parameters that allow for the adjustment of the algorithm, and the effects of these parameters will be discussed in more detail in Section 3. However, in brief, the parameters control the resolution or granularity of the clustering and allow the user to control how many points are clustered versus how many points are classified as noise.

The run-time complexity of our algorithm is $O(n^2)$, where n is the number of points, if the similarity matrix has to be constructed. There are, however, a number of approaches that can be used to reduce the complexity of the algorithm in specific instances, many of them based on approaches successfully proposed and used by other researchers. We discuss the details of these optimizations later, in Section 6.

1.3 Outline of the Paper The rest of the paper is organized as follows. Sections 2 and 3, respectively, describe our approaches to the definition of similarity and density, which are key to our clustering algorithm. The actual clustering algorithm itself is described in Section 4. Section 5 follows up with three case studies: two dimensional point data, NASA Earth Science time series data, and the KDD cup '99 network intrusion data. Section 6 discusses the complexity of our clustering algorithm and strategies for improving the run-time efficiency, while Section 7 presents a short conclusion and directions for future work.

2 An Alternative Definition of Similarity

The most common distance metric used in low dimensional data sets is the Euclidean distance, or the $L2$ 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, $p1$ and $p2$, shown in Table 1. If we calculate the Euclidean distance between these two points, we get 5. Now, consider the next pair of ten-dimensional points, $p3$ and $p4$, which are also shown in Table 1. If we calculate the distance between points $p3$ and $p4$, we again get 5. However, if the points are documents and each attribute is the number of times a word occurs in a document, then,

Point	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
p1	3	0	0	0	0	0	0	0	0	0
p2	0	0	0	0	0	0	0	0	0	4
p3	3	2	4	0	1	2	3	1	2	0
p4	0	2	4	0	1	2	3	1	2	4

Table 1: Four points with ten integer attributes.

intuitively, points $p3$ and $p4$, which share 7 words, seem more similar than points $p1$ and $p2$, which don't share any words. More generally, when data points have a lot of attributes that are often 0, i.e., the data is sparse, similarity (distance) is more usefully defined by ignoring attributes that are zero for both data points. However, as illustrated by this example, Euclidean distance does not capture this notion of closeness. This is further discussed in [14].

Different measures, such as the cosine measure and extended Jaccard coefficient [14], address this problem. In particular, 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, then the cosine similarity simply becomes the dot product of the vectors.) If we calculate the cosine similarity between data points $p1$ and $p2$, and $p3$ and $p4$, we see that the similarity between $p1$ and $p2$ is equal to 0, but is 0.759 between $p3$ and $p4$. (Cosine similarity ranges between 0, no similarity, and 1, completely similar.)

Nonetheless, even though both the cosine and extended Jaccard measures give more importance to the presence of a term than to its absence, there are situations where such similarity measures still have problems in high dimensions. For example, for several TREC data sets which have class labels, we found that $15-20\%$ of the time a document's nearest neighbor, i.e, most similar document according to the cosine measure, is of a different class [15]. The "unreliability" of direct similarity is also illustrated in [19] using a synthetic market basket data set. 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 is low. Typically, data in high dimensions is sparse, and the similarity between data points is, on the average, low.

Another very important problem with similarity measures is that *the triangle inequality doesn't hold.* Consider the following example based on three points shown in Table 2. Using the cosine measure, point $p1$ is close to point $p2$, point $p2$ is close to point $p3$, and yet, points $p1$ and $p3$ are have a similarity of 0. The similarity between points $p1$ and $p2$, and points $p2$ and $p3$ come from different sets of attributes.

Point	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
p1	1	1	1	1	1	0	0	0	0	0
p2	0	0	1	1	1	1	1	1	0	0
p3	0	0	0	0	0	1	1	1	1	1

Table 2: Three points with ten binary attributes.

An alternative to 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) nearest neighbors. If point $p1$ is close to point $p2$ and if they are both close to a set of points, S, then we can say that $p1$ and $p2$ are close with greater confidence since their similarity is "confirmed" by the points in set S. The shared nearest neighbor approach was first introduced by Jarvis and Patrick [8]. A similar idea was later presented in ROCK [19].

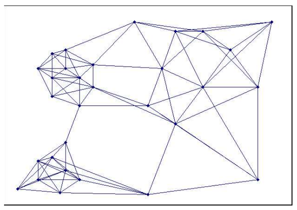
In the Jarvis-Patrick scheme, a shared nearest neighbor (SNN) graph is constructed from the similarity matrix as follows. A link is created between a pair of points, p and q, if and only if p and q have each other in their k-nearest neighbor lists. This process is called k-nearest neighbor sparsification. The weights of the links between two points in the SNN graph can either be simply the number of nearest neighbors the two points share, or can take the ordering of the nearest neighbors into account. Specifically, if p and q be two points. Then, the strength of the link between p and q, i.e., their similarity is defined by the following equation:

$$similarity(p, q) = size(NN(p) \cap NN(q)) \tag{2.1}$$

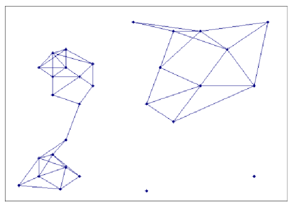
In the above equation, $NN(p)$ and $NN(q)$ are, respectively, the nearest neighbor lists of p and q. At this point, clusters can be obtained by removing all edges with weights (similarities) less than a user specified threshold and taking all the connected components as clusters [8]. We will refer to this as Jarvis-Patrick clustering

Figure 2 illustrates two key properties of the shared nearest neighbor graph in the context of a 2-D point data set. In Figure 2a, links to the five most similar neighbors are drawn for each point. Figure 2b shows the unweighted shared nearest neighbor graph. In this graph, there is a link between points $p1$ and $p2$, only if $p1$ and $p2$ have each other in their nearest neighbor lists.

We make two important observations about this 2-D example that also hold for higher dimensional data. First, noise points and outliers end up having most, if not all, of their links broken. The point in the lower



(a) Near Neighbor Graph.



(b) Unweighted Shared Nearest Neighbor.

Figure 2: Nearest Neighbor Graphs.

right corner of Figure 2b ended up losing all its links, because it wasn't in the nearest neighbor lists of its own nearest neighbors. Thus, construction of the SNN graph removes a lot of noise. Second, the shared nearest neighbor graph keeps the links in regions of uniform density and breaks links in transition regions, i.e., it keeps links in a region of any density, high or low, as long as the region has relatively uniform density. This is an important property, since identifying clusters of widely varying density (tightness) is one of the hardest problems in cluster analysis.

3 An Alternative Definition of Density

In high dimensional data sets, 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, as dimensionality increases, it becomes increasing difficult to use a traditional density based clustering method, such as the one used in DBSCAN, which identifies core points as points in high density regions and noise points as points in low density regions.

A related concept is that of the probability density at a point, which corresponds to measuring the relative density. In the k^{th} nearest neighbor approach to estimating probability density [3], if the k^{th} nearest neighbor of a point is close, then the point is more likely to be in a region which has a relatively high probability density. Thus, the distance to the k^{th} nearest neighbor provides a measure of the density of a point. However, as the dimensionality increases, the Euclidean distance between points becomes increasingly uniform [6] and this approach also works poorly.

Nonetheless, it is possible to use the preceding definition of density, if a "better" similarity (distance) measure can be found, i.e., one that works well for high dimensions. As mentioned in Section 2, the cosine or Jaccard measures work better than Euclidean distance in high dimensional spaces, at least when the data is sparse, e.g., document data. But, as also described, in Section 2, such measures still have problems, and thus, are not suitable for defining a new density measure.

SNN similarity provides us with a more robust measure of similarity, one that works well for data with low, medium and high dimensionality. Consequently, we can use SNN similarity, with the k-nearest neighbor approach to density estimation. More specifically, if the k^{th} nearest neighbor of a point, with respect to SNN similarity, is close, i.e., has high SNN similarity, then we say that there is a high "density" at this point. Since the SNN similarity measure reflects the local configuration of the points in the data space, it is relatively insensitive to variations in density and the dimensionality of the space, and is a promising candidate for a new measure of density.

We remark that a variety of other related measures of density are possible. For example, in previous work [4], we defined the density at a point as the sum of the SNN similarities of a point's k nearest neighbors. We did this for two reasons: a) to reduce the effects of random variation that would result from looking at only one point, and b) to be consistent with a graph-based view of this problem, where we view the density at a vertex as the sum of the weights of the k strongest edges, i.e., as a measure of how strongly the point is connected to other points.

However, in this paper, we will define density as the number of points within a given radius (specified in terms of SNN similarity) since this approach is easier to explain. Results are very similar among different approaches.

Regardless of the exact definition of SNN density adopted, the importance of an SNN approach is that it gives us a method for identifying core or representative points which works regardless of the dimensionality of the data and which is relatively invariant to variations in density. In turn, representative points are the key to handling clusters of differing sizes and shapes.

4 A Shared Nearest Neighbor Clustering Algorithm

We now present an algorithm that uses a density-based approach to find core or representative points. In particular, this approach is based on the notion of "SNN density" introduced in Section 3. This approach automatically compensates for different densities and is not subject to the problem illustrated in Figure 1.

4.1 Identifying Core Points and Removing Noise

Figure 3 illustrates how we can find representative points and effectively remove noise using the SNN graph. In the 2D point data set, shown in Figure 3a, there are $10,000$ points. Subfigures 3b-3d differentiate between these points based on their SNN density. Figure 3b shows the points with the highest SNN density, while Figure 3c shows points of intermediate SNN density, and Figure 3d shows figures of the lowest SNN density.

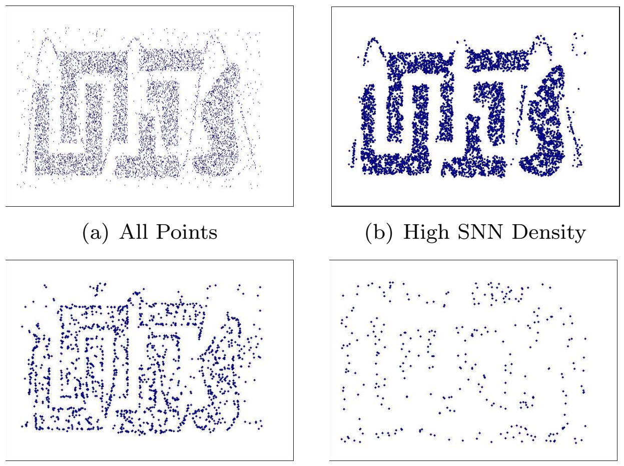

(a) All Points (b) High SNN Density (c) Medium SNN Density (d) Low SNN Density

Figure 3: SNN Density of 2-D Points.

More formally, for this analysis, we used a nearest neighbor list of size of 50. (Therefore, the shared nearest neighbor similarity, i.e., number of nearest neighbors that two points have in common, can range between 0 and 50. While similarity is often measured between 0 and 1, that is not the case here.) We then defined density by counting the number of points that were close to a point, using SNN similarity as our measure of closeness. In particular, we said that two points were close if they shared 20 or more nearest neighbors. Density was then defined as the number of points that were close.

Figure 3b shows all points that have 34 or more points that are close, i.e., share more than 20 nearest neighbors. (A maximum of 50 is possible.). Figure 3c shows all points with $17-33$ points that share 20 or more nearest neighbors, while Figure 3d shows all points that have less than 17 points that share 20 or more nearest neighbors.

In terms of the SNN graph, we only keep links whose similarity is greater than 20, and define density in terms of the number of links that a point has. In DBSCAN terms, we are setting $Eps = 20$. If we regard the points in Figure 3b as core points, then we can, again using DBSCAN terminology, say that $MinPts = 34$.

From these figures, we see that the points that have high density, i.e., high connectivity in the SNN graph, are candidates for being representative or core points since they tend to be located well inside the cluster, while the points that have low connectivity are candidates for being noise points and outliers, as they are mostly in the regions surrounding the clusters.

An alternative way of finding representative points is to consider 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.

4.2 The SNN Clustering Algorithm The steps of the SNN clustering algorithms are as follows:

1. **Compute the similarity matrix.** (This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points.)

2. **Sparsify the similarity matrix by keeping only the k most similar neighbors.** (This corresponds to only keeping the k strongest links of the similarity graph.)

3. **Construct the shared nearest neighbor graph from the sparsified similarity matrix.** At this point, we could apply a similarity threshold and find the connected components to obtain the clusters (Jarvis-Patrick algorithm.)

4. **Find the SNN density of each Point.** Using a user specified parameters, Eps, find the number points that have an SNN similarity of Eps or greater to each point. This is the SNN density of the point.

5. **Find the core points.** Using a user specified parameter, $MinPts$, find the core points, i.e., all points that have an SNN density greater than $MinPts$.

6. **Form clusters from the core points.** If two core points are within a radius, Eps, of each other, then they are placed in the same cluster.

7. **Discard all noise points.** All non-core points that are not within a radius of Eps of a core point are discarded.

8. **Assign all non-noise, non-core points to clusters.** We can do this by assigning such points to the nearest core point. (Note that steps 4-8 are DBSCAN.)

The algorithm, being an extension of Jarvis-Patrick and DBSCAN, will determine the number of clusters in the data automatically. Also note that not all the points are clustered. Depending on the application, we might actually want to discard many of the points.

4.3 Parametrization The neighborhood list size, k, is the most important parameter as it determines the granularity of the clusters. If k is too small, even a uniform cluster will be broken up into pieces due to local variations in the similarity, and the algorithm will tend to find many small, but tight, clusters. On the other hand, if k is too large, then the algorithm will tend to find only a few large, well-separated clusters, and small local variations in similarity will not have an impact. The parameter k, adjusts the focus of the clusters. Once the neighborhood size is fixed, the nature of the clusters that will be produced is also fixed.

In the SNN graph, a point can be similar to at most k other points, therefore the $MinPts$ parameter should be a fraction of the neighborhood list size, k.

4.4 Variations of the SNN Algorithm In our experiments, we observed that minor changes to the DBSCAN algorithm provide more control and stability. DBSCAN uses the *Eps* parameter to define the neighborhood as well as defining the criteria to merge two clusters. We can use a tighter threshold than *Eps* to identify core points, and by doing so; the core points we identify will be more reliable. Furthermore, if we chose the core points more conservatively, then we can also use a looser threshold than *Eps* for merging core points.

We can classify very loosely connected points as noise points, either as a pre-processing step for a clustering algorithm or just for noise removal. From Figure 3d, we can clearly see that noise removal can be performed effectively using the SNN graph.

We used some of these variations in the experiments presented later in the paper.

5 Experimental Results

5.1 2D Data We first illustrate the superiority of our SNN clustering algorithm with respect to the Jarvis-Patrick approach. A major drawback of the Jarvis-Patrick scheme is that it is difficult to find the "right" value for the SNN similarity threshold. If the threshold is too low, then distinct sets of points can be merged into same cluster even if there is only one link between them. On the other hand, if the threshold is too high, then a natural cluster may be split into small clusters due to natural variations in the similarity within the cluster. Indeed, there may be no right threshold for some data sets. This problem is illustrated in the following example. Figure 4 shows Chameleon and Cure clustering results for a data set taken from [9]. Figure 4a shows the clusters found by the Jarvis-Patrick algorithm using the smallest possible threshold that will not cause two genuine clusters to merge. Notice that many of the "natural" clusters are broken. Figure 4b shows the clusters that result when this threshold is decreased by one. In this case, several "natural" clusters are merged Clearly, there is no correct threshold for the Jarvis-Patrick algorithm in this example.

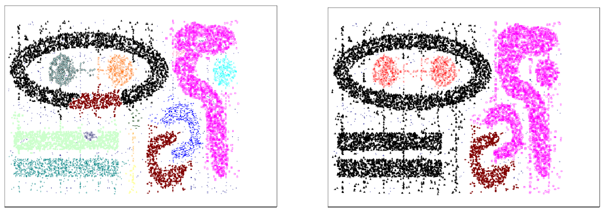

(a) No-merging Threshold (b) No-merging Threshold + 1

Figure 4: Jarvis-Patrick clustering with two thresholds that differ by 1 (best seen in color)

Figure 5a shows the clusters identified by our algorithm on a Chameleon data set [9], while Figure 5b shows the results of our algorithm on a CURE data set [18]. It was shown in [9] that DBSCAN cannot identify the clusters correctly in the CURE data set and CURE cannot identify the clusters correctly in the Chameleon data set. Figures 5a and 5b illustrate that, by combining the strengths of Jarvis-Patrick algorithm with the idea of representative points from CURE and DBSCAN, we can obtain much better results.

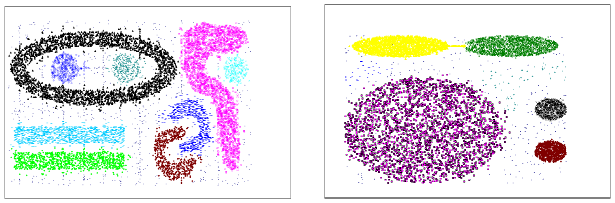

a. Chameleon Data set b. CURE Data set

Figure 5: SNN clustering on two 2D data sets (best seen in color)

5.2 NASA 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 1982 to 1993, i.e., each time series is a 144 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 [16, 17].

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 climate phenomena on the land. Such time series are called Climate Indices (CIs). 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 [14]. Our goal in clustering SLP is to see if the difference of cluster centroids can yield a time series that reproduces known CIs and to perhaps discover new indices.

Our SNN clustering approach yields the clusters shown in Figure 6a. (Notice, that while these clusters are shown in two dimensions, they correspond to time series clusters, which, because of the association between the times series and locations on the globe, can be displayed in a two-dimensional plot.) These clusters have been labeled for easy reference. (Note that cluster "1" is the background or "junk" cluster, and that while we cluster pressure over the entire globe, we focus on the ocean.) Although we cluster the time series independently of any spatial information, the resulting clusters are typically geographically contiguous because of the underlying spatial autocorrelation of the data.

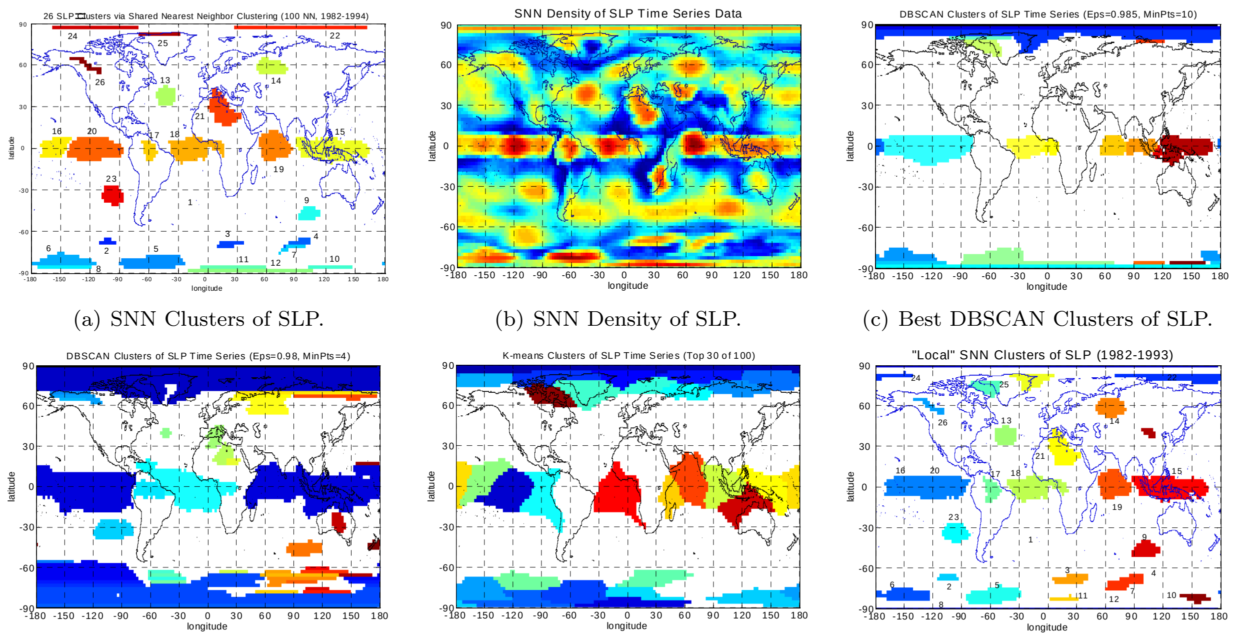
Using these clusters, we have been able to reproduce SOI as the difference of the centroids of clusters 15 and 20. We have also been able to reproduce another well-known CI, i.e., NAO (North Atlantic Oscillation), which is the normalized SLP differences between Ponta Delgada, Azores and Stykkisholmur, Iceland. NAO corresponds to the differences of the centroids of clusters 25 and 13. For more details, see [17]. This success gives us confidence that the clusters discovered by SNN have real physical significance.

The reader might wonder how we know that these are the "real" clusters, i.e., is it possible that the clusters we find might change significantly if we change the parameters for the SNN clustering algorithm? However, that is not the case, and basically the same clusters are produced for different parameterizations. We illustrate why this is so by plotting the SNN density (as discussed in Section 4) at each point - see Figure 6b. The densest (reddest, darkest) locations correspond to time series that will be the core points, although exactly which ones are chosen will depend on a threshold. The time series that belong to the cluster corresponding to a "core point" will typically be the time series associated with the surrounding physical locations, since spatial autocorrelation makes the time series of physically close locations be highly similar. Thus, for this data, the SNN clusters can be picked out visually and are in good agreement with those produced by our clustering algorithm.

It is reasonable to wonder whether other clustering techniques could also discover these clusters. To answer that question, we clustered the same data using DBSCAN and K-means. The best results for DBSCAN clustering are shown in Figure 6c, while Figure 6d shows the results for a less optimal parameter choice for DBSCAN clustering. (Note that the colors have no meaning except to distinguish the different clusters and are not coordinated between any of the different figures.) While there are similarities between the two figures, it seems that to find the less dense clusters, e.g., 13 and 23, which are the clusters not on the equators or at the poles, it is necessary to change the parameterization. These less dense clusters are present in Figure 6d, but it is also clear that we have "blurred" the other clusters in our attempt to find the "weaker" clusters.

For K-means, we attempted to find roughly the same number of clusters as with our SNN approach, i.e., $\approx$ 30. (For the K-means approach, we used the CLUTO package [1] with the bisecting K-means algorithm and the refine parameter.) However, since K-means naturally clusters all the data, in order to be fair to K-means, 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 6e are not a very good match for the clusters of Figure 6a, although they do show the same pattern of clusters around the poles and along the equator. Note that cluster 6b, 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 13 ($corr = 0.99$), but it is the 57^{th} least cohesive cluster with respect to average cluster correlation.

For both DBSCAN and K-means, the main problem is that the "best" clusters are those which are the densest, i.e., tightest. In this Earth science domain, clusters which are relatively uniform, although not as dense, are also important. (We speculate that regions of the ocean that are relatively uniform have more influence on the land, even if their tightness is not as high as that of other regions.) For this Earth


26 SLP Clusters via Shared Nearest Neighbor Clustering (100 NN, 1982-1994)
SNN Density of SLP Time Series Data
DBSCAN Clusters of SLP Time Series (Eps=0.985, MinPts=10)
DBSCAN Clusters of SLP Time Series (Eps=0.98, MinPts=4)
K-means Clusters of SLP Time Series (Top 30 of 100)
"Local" SNN Clusters of SLP (1982-1993)
latitude
longitude

(a) SNN Clusters of SLP. (b) SNN Density of SLP. (c) Best DBSCAN Clusters of SLP.

(d) Suboptimal DBSCAN Clusters of SLP. (e) K-means Clusters of SLP. (f) Local SNN Clusters of SLP.

Figure 6: SNN, K-means and DBSCAN clustering on Earth Science data sets

science domain, it is well known that the equatorial SLP clusters correlate highly to the El Nino phenomenon, which is the strongest CI. However, for K-means and DBSCAN, these "strong" clusters tend to mask the weaker clusters.

5.3 KDD CUP '99 Network Intrusion Data In this section, we present experimental results from the KDD Cup '99 data set which is widely used for benchmarking network intrusion detection systems. The data set consists of TCP sessions whose attributes include TCP connection properties, network traffic statistics using a 2 second window, and several attributes extracted from TCP packets. We also have access to the labels for the TCP sessions, which take one of the 5 possible values: normal, u2r, dos, r2l, or probe.

The original data set is very large, and therefore, we sampled the data using a simple scheme. We picked all the attack sessions from the training and the test data and put a cap of $10,000$ records for each sub-attack type. (There are 36 actual attack types in the data that are grouped into 4 categories: u2r, dos, r2l, and probe.) We also picked $10,000$ normal sessions from both the test and training sets, which resulted in in a data set of approximately $97,000$ records. We then removed duplicate sessions, reducing the data set size to $45,000$ records.

We clustered the data using our SNN clustering algorithm and K-means ($K = 100, 300, 1000$). Tables 3 and 5 show the purity of the clusters for our algorithm and for K-means ($K = 300$). Since our algorithm removes noise, we also constructed a set of "better" K-means results by taking only enough of the tightest K-means clusters so that the size of these clusters added up to the total size of the SNN clusters. These modified K-means results are shown in Table 4. For all of the K-mean results, we used CLUTO [5] with default values to obtain a K-way partitioning since CLUTO does a better job than standard implementations of K-means.

In tables 3, 4, and 5, we can clearly see that the level of impurity for K-means clusters is considerably higher than that of SNN clusters, even when we only consider the tightest K-means clusters. In particular, for the rare class, u2r (user to root attack), K-means picks up only 15 out of 267 correctly (using the majority rule), while our algorithm picks up 101 out of 267. When we look at the tightest K-means clusters, K-means doesn't pick up anything from the rare class. The clusters that our algorithm found were almost always superior to K-means clusters, even for many different values of K. For example, for the tightest clusters of the 1000-way partition, K-means was better only in identifying probe class; the impurity of probe class was 5.63% as opposed to 5.95% for our algorithm.

The size of the largest K-means clusters for $K = 300$ was (coincidentally) 300 and the largest cluster we obtained using our algorithm was 479 (out of 521 clusters). Even though our algorithm had many more clusters in the output, the largest cluster was much bigger than the largest K-means cluster. This demonstrates that our algorithm is able to find more unbalanced clusters when they are present in the data.

Table 3: Purity across classes - K-means (K=300)

	total	correct	incorrect	impurity
normal	18183	17458	725	3.99%
u2r	267	15	252	94.38%
dos	17408	17035	373	2.14%
r2l	3894	3000	894	22.96%
probe	4672	4293	379	8.11

Table 4: Purity across tightest classes - K-means (K=300)

	total	correct	incorrect	missing	impurity
normal	18183	9472	520	8191	5.20%
u2r	267	0	113	154	100.0%
dos	17408	16221	186	1001	1.13%
r2l	3894	2569	471	854	15.49%
probe	4672	3610	302	760	7.72

Table 5: Purity across classes - SNN Clusters

	total	correct	incorrect	missing	impurity
normal	18183	12708	327	5148	2.51%
u2r	267	101	67	99	39.88%
dos	17408	13537	53	3818	0.39%
r2l	3894	2654	257	983	8.83%
probe	4672	3431	217	1024	5.95

6 Complexity Analysis

Our SNN clustering algorithm requires $O(n^2)$ time (n is the number of data points) if all pairwise similarities are computed to find the k nearest neighbors. The space complexity is $O(k*n)$ since only the k nearest neighbors need to be stored. While the k-nearest neighbor list can be computed once and used repeatedly for different runs of the algorithm with different parameter values, the initial computation of the nearest neighbor list can become a bottleneck, especially for high dimensional data. (For low dimensional data, the complexity of computing the nearest neighbor list can be reduced to $n * \log(n)$ by the use of a data structure such as a k-d tree or an R* tree.)

While there is no completely general technique for finding nearest neighbors efficiently (in time less than $O(n^2)$) in high dimensions, the technique of canopies [12] is applicable in many cases. This approach first cheaply partitions a set of data points by repeatedly selecting a point and then forming a group (around the selected point) of all other points that are within a certain similarity threshold. The closest of these points are removed from further consideration, and the process is repeated until the set of points is exhausted. (If the actual similarity measure is expensive, an approximate similarity measure can be used to further increase efficiency.) The result of this process is a set of (possibly overlapping) groups of points which are much smaller than the original set and which can then be processed individually. In our case, canopies would reduce the computation required for finding the nearest neighbors of each point since each point's nearest neighbors will only be selected from its canopy. The rest of the SNN algorithm, which is linear in the number of points, would proceed as before.

While canopies represent a general approach for speeding up nearest neighbor list computation, other, more domain specific approaches are also possible. In the remainder of this section we present a couple of approaches that, respectively, substantially reduce the amount of computation required for documents and for NASA Earth science time series. For documents efficiency is enhanced by using an inverted index and the fact that document vectors are rather sparse. For times series associated with physical points, efficiency can be enhanced by realizing that the most similar time series are almost always associated with physical points that are geographically close.

For our Earth science data the $O(n^2)$ complexity has not been a problem yet, since the current data sets, which consist of data sets of $< 100,000$ records with dimensionality of roughly 500, can be handled in less than a day on a single processor. However, Earth scientists are collecting data using smaller and smaller grid sizes. For example, if the grid size at which data is collected is halved, the number of data points, n, increases by a factor of 4, and the SNN computation increases by a factor of 16. Consequently, in the future computational issues may be more important.

Fortunately it is possible to take advantage of the spatial autocorrelation of this data to significantly reduce the time complexity of SNN to $O(n)$. The basic idea is as follows: because of spatial autocorrelation, time series from two different physical points on the globe that are geographically close tend to be more similar than those from points that are geographically more distant. (Please note the distinction between physical points on the globe and the data points, which are time series associated with those points.) Consequently, if we look at the nearest neighbors (most similar time series) of a particular time series, then they are quite likely to be time series corresponding to points that are spatially very close, and we can compute the nearest neighbor list of a time series—at least to a high accuracy—by looking only at the time series associated with a relatively small number of surrounding locations, e.g., a few hundred. This reduces the complexity of the computation to $O(n)$. Of course, this improvement in complexity is not useful unless the resulting clusters are about the same. However, in preliminary experiments this is the case. Figure 6f shows the clusters resulting for this "local" SNN clustering approach, which are quite similar to those in Figure 6a, which required required the computation of all pairwise similarities. For this experiment, we built the nearest neighbor list (of size 100) of each time series by looking only at the time series for the 200 closest physical locations.

For document clustering, every document contains a fraction of the words from the vocabulary and the document vectors are quite sparse. One of the most widely used similarity measures in document clustering is the cosine measure. According to the cosine measure, for two documents to have a non-zero similarity, they have to share at least one word. Thus, for a given document d, we can find the list of documents that have a non-zero similarity to d by keeping a list of all the documents that contain at least one of the terms in document d. The transpose of the document-term matrix (inverted index) consists of word vectors that are lists of documents that contain each word. If we maintain the inverted index for the data we are trying to cluster, then we can construct the list of similar documents for a given document by taking the union of the word vectors for each word in that particular document. After this list is constructed, we only need to calculate the similarity of the given document to the documents in this list since all other documents will have a zero similarity. This avoids unnecessary computation.

A second optimization can be performed by exploiting the fact that for a given document, only the top (high-weighted) few words contribute substantially to the norm. If we only keep the high-weighted terms that make up most of the norm (90% or so), we can reduce the number of similar documents considerably, but not lose much information. Using this optimization resulted in a speedup of 30 over the case where only the first optimization was used. We clustered 3200 documents in 2 seconds and $100,000$ documents in less than an hour on a Pentium 3, 600MHz desktop computer.

7 Conclusions and Future Work

In this paper we described a novel shared nearest neighbor clustering algorithm that can find clusters of varying shapes, sizes, and densities, even in the presence of noise and outliers. The algorithm can handle data of high dimensionality and varying densities, and automatically determines the number of clusters. Thus, we believe that our algorithm provides a robust alternative to many other clustering approaches that are more limited in the types of data and clusters that they can handle.

In particular, our SNN clustering algorithm can find clusters that represent relatively uniform regions with respect to their "surroundings," even if these clusters are of low or medium density. We presented an example of this in the context of NASA Earth science time series data, where our SNN clustering approach was able to simultaneously find clusters of different densities that were important for explaining how the ocean influences the land. DBSCAN could only find the "weaker" cluster by sacrificing the quality of the "stronger" clusters. K-means could find the weaker cluster, but only as one of a large number of clusters. Furthermore, since the quality of this cluster was low in K-means terms, there was no way to identify this cluster as being anything special.

In addition to the NASA time series data example, we also presented examples of the superior performance of our algorithm with respect to some other well-known clustering algorithms for KDD cup data and two dimensional point data sets. In particular, SNN clustering is consistently able to overcome problems with differing cluster densities that cause other techniques (even Jarvis-Patrick) to fail. For the KDD Cup '99 network intrusion data, SNN clustering was able to produce purer clusters than the clusters produced by K-means.

While our clustering algorithm has a basic time complexity of $O(n^2)$, there are a number of possible optimizations that provide reasonable run-time for specific domains. The basic idea in all cases is to find a cheaper way of computing the nearest neighbors of a points by restricting the number of points considered. The most general approach to accomplish this is to use method of canopies to split the data into smaller sets and find the nearest neighbors of a point only among the points in its canopy. However, for documents and Earth science data, several domain specific approaches are possible that can reduce the required computation by one or two orders of magnitude.

An important goal of our future research will be to investigate the behavior of our SNN clustering approach on other types of data, e.g., transaction data and genomic data. We feel that looking at a wide variety of data sets and comparing the performance of our algorithm to that of other clustering algorithms is the best way to better understand our algorithm and to discover its strengths and weaknesses.

Another major goal will be to explore if the ideas in our SNN clustering algorithm can be extended in the same way that the ideas of DBSCAN have been extended with respect to outlier detection [2], identifying the clustering structure, OPTICS [1], and performing hierarchical clustering [1].

Finally, we have made our SNN clustering algorithm publicly available so that others can try it for themselves. It can be download from http://www.cs.umn.edu/∼ertoz/snn/

8 Acknowledgements

This work was partially supported by Army High Performance Computing Research Center cooperative agreement number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

References

[1] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, Philadelphia, Pennsylvania, USA*, pages 49–60. ACM Press, June 1999.

[2] M. M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. Lof: identifying density-based local outliers. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA*, pages 93–104. ACM Press, May 2000.

[3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Series in Probability and Statistics. John Wiley and Sons, New York, second edition, Novemeber 2001.

[4] L. Ertöz, M. Steinbach, and V. Kumar. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on Clustering High Dimensional Data and its Applications, Second SIAM International Conference on Data Mining, Arlington, VA,*, 2002.

[5] George Karypis. Cluto 2.1: Software for clustering high-dimensional datasets. /www.cs.umn.edu/˜karypis, August 2002.

[6] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In R. Agrawal and P. Stolorz, editors, *KDD '98, Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining, New York City, New York, USA*, pages 58–65. AAAI Press, August 1998.

[7] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series. Prentice Hall, Englewood Cliffs, New Jersey, March 1988.

[8] R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared nearest neighbors. *IEEE Transactions on Computers*, C-22(11), 1973.

[9] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, August 1999.

[10] G. Karypis and V. Kumar. Tr# 98-019: Multilevel algorithms for multi-constraint graph partitioning. Technical report, University of Minnesota, Minneapolis, MN, 1998.

[11] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. John Wiley and Sons, New York, Novemeber 1990.

[12] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Kdd Proceedings 2000: Conference of Knowledge Discovery of Data-Mining, Boston, MA, USA*, pages 169–178. ACM Press, August 2000.

[13] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[14] A. Stehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Workshop of Artificial Intelligence for Web Search, 17th National Conference on Artificial Intelligence*, 2000.

[15] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, August 2000.

[16] M. Steinbach, P.-N. Tan, V. Kumar, C. Potter, S. Klooster, and A. Torregrosa. Clustering earth science data: Goals, issues and results. In *Fourth KDD Workshop on Mining Scientific Datasets, San Francisco, California, USA*, August 2001.

[17] M. Steinbach, P.-N. Tan, V. Kumar, C. Potter, and S. Klooster. Data mining for the discovery of ocean climate indices. In *Mining Scientific Datasets Workshop, 2nd Annual SIAM International Conference on Data Mining*, April 2002.

[18] K. S. Sudipto Guha, Rajeev Rastogi. Cure: An efficient clustering algorithm for large databases. In L. M. Haas and A. Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, Washington, USA*, pages 73–84. ACM Press, June 1998.

[19] K. S. Sudipto Guha, Rajeev Rastogi. Rock: A robust clustering algorithm for categorical attributes. In M. Kitsuregawa, L. Maciaszek, M. Papazoglou, and C. Pu, editors, *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia*, pages 512–521. IEEE Computer Society, March 1999.