Cluster Hull:
A Technique for Summarizing Spatial Data Streams

John Hershberger   Nisheeth Shrivastava   Subhash Suri[*]

Mentor Graphics Corp.
8005 SW Boeckman Road
Wilsonville, OR 97070, USA.
john_hershberger@mentor.com

Department of Computer Science
University of California
Santa Barbara, CA 93106, USA.
{nisheeth,suri}@cs.ucsb.edu

1 Introduction

Recently there has been a growing interest in detecting patterns and analyzing trends in data that are generated continuously, often delivered in some fixed order and at a rapid rate, in the form of a *data stream* [5, 6]. When the stream consists of spatial data, its geometric "shape" can convey important qualitative aspects of the data set more effectively than many numerical statistics. In a stream setting, where the data must be constantly discarded and compressed, special care must be taken to ensure that the compressed summary faithfully captures the overall shape of the point distribution. We propose a novel scheme, ClusterHulls, to represent the *shape* of a stream of two-dimensional points.

ClusterHull

Given an on-line, possibly unbounded stream of 2-D points, we propose a scheme for summarizing its spatial distribution or *shape* using a small, bounded amount of memory m. Our scheme, called *ClusterHull*, represents the shape of the stream as a dynamic collection of convex hulls, with a total of at most m vertices. The algorithm dynamically adjusts both the number of hulls and the number of vertices in each hull to represent the stream using its fixed memory budget. Thus, the algorithm attempts to capture the shape by decomposing the stream of points into groups or clusters and maintaining an approximate convex hull of each group. Depending on the input, the algorithm adaptively spends more points on clusters with complex (potentially more interesting) boundaries and fewer on simple clusters.

We implemented ClusterHull and experimented with both synthetic and real data to evaluate its performance. In all cases, the representation by ClusterHull appears to be more information-rich than those by other clustering schemes such as CURE [4], k-medians, or LSEARCH [3], even when the latter are enhanced with some simple ways to capture cluster shape. For example, in Figure 1 below, we compare the output of our ClusterHull algorithm with those produced by k-median and CURE. The top row shows the input data (left), and output of ClusterHull (right) with memory budget set to $m = 45$ points. The middle row shows two possible outputs of k-median, one with 45 cluster centers and another with 5 cluster centers each with 9 sample points. The bottom row shows the similar outputs of CURE: 45 cluster centers on the left, and 5 cluster centers each with 9 sample points on the right. One can see that both the boundary shape and the densities of the point clusters are quite accurately summarized by the cluster hulls. Thus, our general conclusion is that ClusterHull can be a useful tool for summarizing geometric data streams.

Related Work

Inferring shape from an unordered point cloud is a well-studied problem that has been considered in many fields, including computer vision, machine learning, pattern analysis, and computational geometry, and most extensively as *clustering* in databases [1, 2]. However, the classical algorithms from these areas require full access to data, and are generally based on minimizing some statistical function, making them unsuited for our problem of extracting the geometric shapes in data stream setting.

Among the clustering schemes that work for data streams, BIRCH [8] appears to work well for spherical-shaped clusters of uniform size, but performs poorly when the data are clustered into groups of unequal sizes and different shapes [4]. The CURE clustering scheme proposed by Guha et al. [4] is better at identifying non-spherical clusters; in addition, because it maintains sample points for each cluster, one can also infer some information about the

[*]Subhash Suri's research was supported in part by the National Science Foundation under grant IIS-0121562.

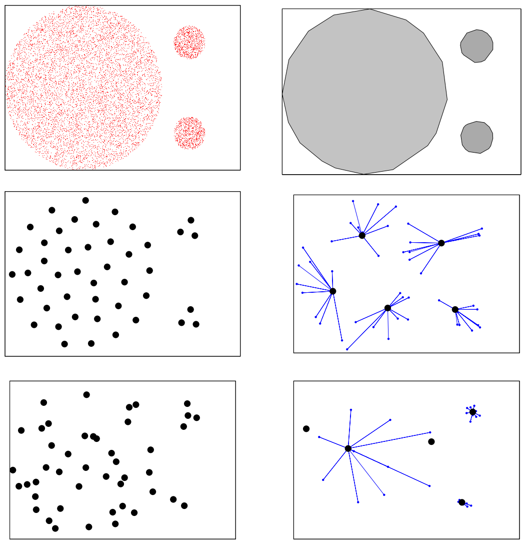

Figure 1. Illustration of ClusterHull.

geometry of the cluster. Therefore, CURE is one of the schemes we use to compare against ClusterHull.

In [3], Guha et al. propose two stream variants of k-center clustering: stream k-median and LSEARCH (local search). Through experimentation, they argue that the stream versions of these algorithms produce better quality clusters than BIRCH, although the latter is computationally more efficient. Since we are more concerned with the quality of the shape, we also report on experimental results comparing ClusterHull with k-median and LSEARCH.

2 Shape as a Cluster of Hulls

Our algorithm represents the point stream as a collection of convex hulls. The convex hulls need not be disjoint, but together these hulls are limited to m vertices, which is our memory budget. We divide this memory budget into two pools: a fixed pool of k groups, each with a constant number of vertices; and a shared pool of $O(k)$ points, from which different cluster hulls draw additional vertices. The number k has the same role as the parameter fed to k-medians clustering—it is set to some number at least as large as the number of native clusters expected in the input.

At a high level, our algorithm works as follows. Suppose that the current point set S is partitioned among k convex hulls $H_1, \ldots, H_k$. Each convex hull H is evaluated using a cost function $w(H)$, which has the following general form:

$$w(H) = \texttt{area}(H) + \mu \cdot (\texttt{perimeter}(H))^2, \quad (1)$$

where μ is a constant, chosen empirically.[1] The total cost of the partition $\mathcal{H} = \{H_1, \ldots, H_k\}$ is $w(\mathcal{H}) = \sum_{H \in \mathcal{H}} w(H)$. The algorithm attempts to maintain a partition of the point stream with *minimum* total cost, using its fixed $O(k)$ memory budget.

When a new point p arrives in the stream, the algorithm checks if p either lies inside a hull H_i or is *close enough* to it; please see the expanded version of this paper for a formal definition of "close enough," which is defined using a ring of uncertainty triangles. If so, then we simply discard p. Otherwise, we create a new hull, containing just p, and add it to $\mathcal{H}$. If $\mathcal{H}$ now has more than k hulls, then we merge two of the hulls in $\mathcal{H}$, where the merging hulls are chosen so as to *minimize* the total *increase* in the cost function.

Since the size (number of vertices) of the convex hulls can be very large, we use an approximate convex hull representation [5]. For each hull, the algorithm uses extreme points along certain sampling directions as the vertices of the approximate hull. After the merge, we carry out a *refinement* process to ensure that the new hull representation is within our error bounds. Finally, if the total memory usage exceeds m, then we carry out an *unrefinement* process that discards some sampling directions.

3 Experimental Analysis

We now briefly report on our experimental evaluation of ClusterHull. When comparing our scheme with k-median clustering [3], we used an enhanced version of the latter. The algorithm is allowed to keep a constant number of sample points per cluster, which can be used to deduce the approximate shape of that cluster. We ran k-medians clustering using k clusters and total memory (number of samples) equal to m. CURE already has a parameter for maintaining samples in each cluster, so we used that feature.

For these experiments, we used the area-perimeter cost (Equation 1) to compute the hulls, with $\mu = .05$ for all data sets. To visualize the output, we also shade the hulls generated by our algorithm according to their densities (darker regions are more dense).

3.1 West Nile virus spread

Our first data set, *westnile* (Figure 2 (a)), contains about $68,000$ points corresponding to the locations of the *West Nile* virus cases reported in the US, as collected by the CDC and the USGS [7]. We randomized the input order to eliminate any spatial coherence that might give an advantage to

[1]A comprehensive discussion of the intuition behind the cost function is provided in the full version of the paper. We point out that the perimeter is squared in this expression to match units: if the perimeter term entered linearly, then simply changing the units of measurement would change the relative importance of the area and perimeter terms, which would be undesirable.

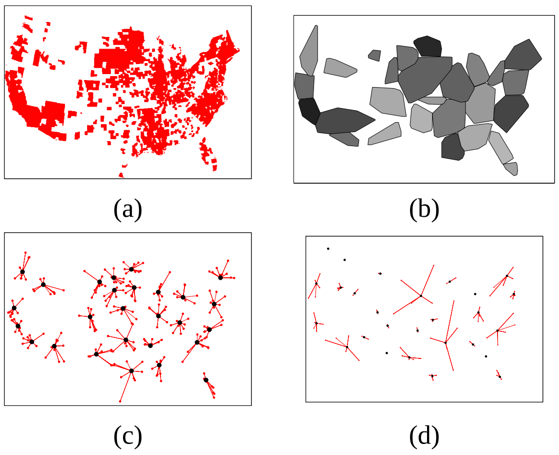

Figure 2. Results for *westnile* data set.

our algorithm. We ran ClusterHull to generate output of total size $m = 256$ (Figure 2 (b)). The clustering algorithms k-medians and CURE were used to generate clusters with the same amount of memory. Both algorithms were ran with $k = 27$ clusters (same as ClusterHull) and 10 sample points for each cluster (total $m > 256$). Figures 2 (c) and 2 (d) show the corresponding outputs for k-medians and for CURE. Visually the output of ClusterHull looks strikingly similar to the input set, offering the analyst a faithful yet compact representation of the geometric shapes of important regions.

3.2 The circles and the ellipse data sets

In this experiment, we compared ClusterHull with k-median and CURE on two synthetic data sets. The circles set contains $n = 10,000$ points generated inside 3 circles of different sizes. We ran the three algorithms with a total memory $m = 64$ (for k-median and CURE, $k = 5$, 14 samples per hull). The output of ClusterHull is shown in Figure 3 (a); the output of k-median is shown in (b); and the output of CURE is shown in (c).

The ellipse data set contains $n = 10,000$ points distributed among ten ellipse-shaped clusters. Figures 3 (d), (e), and (f), respectively, show the outputs of ClusterHull, k-median, and CURE on this set with memory $m = 128$ (for k-median and CURE, $k = 10$, 13 samples per hull).

In all cases, ClusterHull output is more accurate, visually informative, and able to compute the boundary of clusters with remarkable precision. The outputs of other schemes are ambiguous, inaccurate, and lacking in details of the cluster shape boundary. For the circles data, the k-median does a poor job in determining the true cluster structure. For the ellipse data, CURE does a poor job in separating the clusters. (CURE needed a much larger memory—a window size of at least 500—to correctly separate the clusters.)

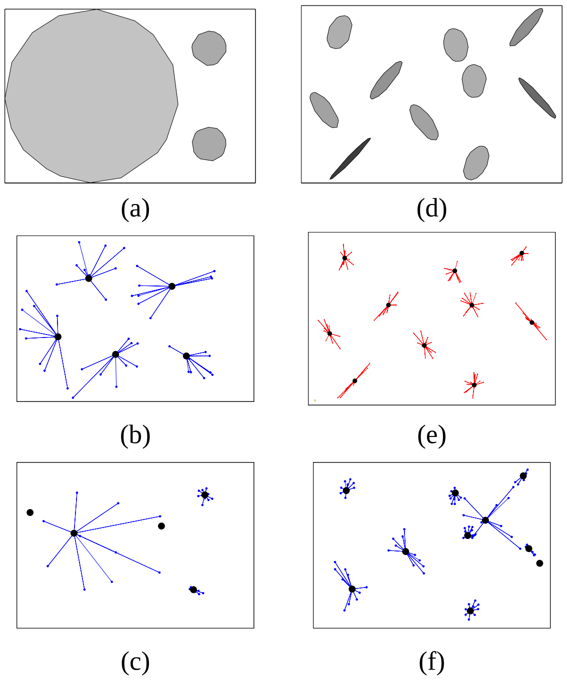

Figure 3. Results for *circles* and *ellipse* datasets.

References

[1] P. S. Bradley, U. Fayyad, and C. Reina. Scaling Clustering Algorithms to Large Databases. Proc. 4th International Conf. on Knowledge Discovery and Data Mining (KDD-98), 1998.

[2] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proc. 29th Annu. ACM Symp. Theory Computing*, pages 626–635, 1997.

[3] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. *Clustering data streams: Theory and practice.* IEEE Trans. Knowl. Data Engineering, Vol. 15, pages 515–528, 2003.

[4] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In Proc. of International Conference on Management of Data (SIGMOD), 1998.

[5] J. Hershberger and S. Suri. Adaptive sampling for geometric problems over data streams. In *Proc. 23rd ACM Sympos. Principles Database Syst.*, pages 252–262, 2004.

[6] S. Muthukrishnan. Data streams: Algorithms and applications. Preprint, 2003.

[7] USGS West Nile Virus Maps - 2002. `http://cindi.usgs.gov/hazard/event/west_nile/`

[8] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In Proc. of the International Conference on Management of Data (SIGMOD), 1996.