HIERARCHICAL GIS CLUSTERING USING PRINCIPAL COMPONENTS

Abhinav Dayal

IDV Solutions Inc. 5913 Executive Drive, Suite 320, Lansing, MI 48911

1. INTRODUCTION

In any interactive mapping application to visualize point data, point features tend to overlap on map, causing redundancy of view, slowness of rendering and too much information making it difficult to infer. Clustering of points reduces point features by grouping for better visualization, better user experience and faster display. A good clustering algorithm in GIS should be efficient and produce informative results, following the trends in the underlying data.

There are several clustering techniques available as described in [1]. Partitioning methods depend highly on the heuristics of initial choice or are of quadratic complexity. Most density based methods are of quadratic complexity. Hierarchical and grid based methods seems more promising for GIS application. Methods like Wavecluster [2], produce natural clusters more suited for data mining and not in GIS. In this paper we present a hierarchical approach using BSP trees that splits data along the least significant dimension. [3] uses principle components to generate Voronoi diagram for a set of points for data reduction and approximation. [4] uses PCA to determine discrete cluster membership of cluster points in K-means method.

The algorithm we present is of logarithmic complexity in time and linear complexity in space giving a truly remarkable performance. Moreover since grouping is based on the characteristics of the underlying data the resulting cluster points give an intuitive user experience. The algorithm easily identifies overlapping points and has ability to group such points separately. For example apartments in the same building may be represented by a single point on the map instead of overlapping points that never separate apart no matter how detailed a map you choose. Finally we also allow the user to control how much to cluster based on the map extents and a custom factor between 0 (no clustering) and 1 (cluster all points to a single point).

2. METHODOLOGY

Figure 1 shows a simple demonstration. We use a top down or divisive approach. Beginning from an array of all the input points, we partition object space hierarchically. At each level, we find the mean location $\mu = (x, y)$ and the principal component vector$\vec{V}$. A perpendicular of the principal component $\overrightarrow{V_\perp}$ indicates the least significant dimension. A line in the direction of $\overrightarrow{V_\perp}$passing through the mean μ indicates a natural boundary in the data set. We use this to split the points into two halves. In each part we recursively apply the same approach till a stopping condition, c becomes true.

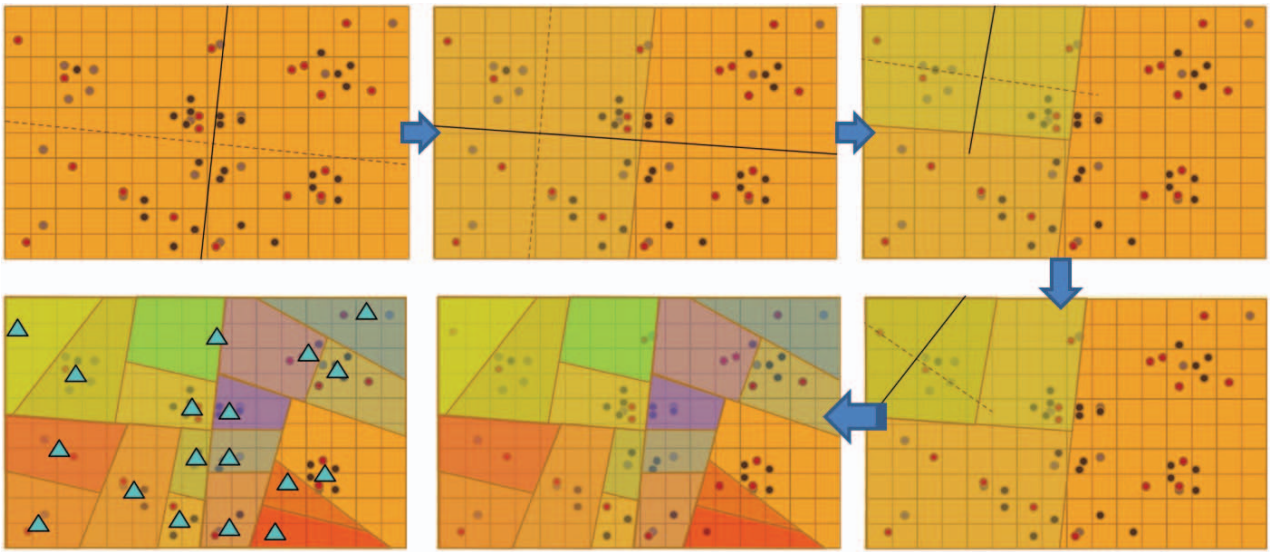

Figure 1. PCA based hierarchical GIS clustering demonstration.

Stopping Condition: Let Δx be the x or longitude span on points in dataset and let Δy be the latitude or y span. Given a user controlled factor, $\delta \in [0,1]$, and dimensions of the current view, $(W°, H°)$, where $W°$is the width and $H°$ is the height of the view in longitude difference and latitude difference respectively, we define the span ratio, $\tau = \begin{cases} \Delta x/W°, if\ \Delta x > \Delta y \\ \Delta y/H°, otherwise \end{cases}$. We then define our stopping condition $c = \tau \leq \delta$. For overlapping points $\tau = 0$ and thus $c = true\ \forall \delta > 0$.

Cluster Point Location: For each terminal node in the BSP tree, we choose the location of a cluster point as that of an existing point in the node, whose position is closest to the arithmetic mean of all the points in the cluster. Choosing the actual mean could cause poor visualization, for example a land feature may be shown as a cluster point in water body instead.

Computational Complexity: Since this method builds a binary tree the number of splits is of the order of $O[\log_2 n]$, where n is the size of the input. Finding the principle component of a dataset of size n is of the order of $O[n]$. Therefore overall complexity of execution of this algorithm is $O[n \log_2 n]$. Actually this is also the worst case scenario.

Space Complexity: The memory complexity can be made linear or $O[n]$, by doing the splits in the input list by mere shuffling of the points in the array. Figure 2 shows a sample demonstration.

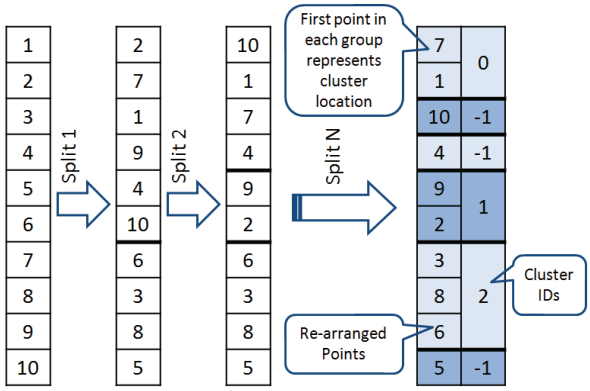

Figure 2. Achieving linear(O[n]) space complexity.

3. RESULTS AND CONCLUSION

We choose a huge data set of 150,765 point features in entire world as shown in Figure 3. On a Centrino 2.0 GHz with 3GB Ram, it took 546ms to generate entire BSP tree with the stopping condition of a single point in the leaf node (total 601,881 nodes). The actual runtime varies based on view dimensions and user controlled factor (125ms in this example). Notice how clusters are not regular grid like points but vary based on the local density of the points.

Figure 3. Clustering results. 1265 clusters in 125 ms with threshold, δ of 0.05. (Left) shows actual points in purple and cluster points as dark blue dots. (Right) shows polygon extents of each cluster point.

4. REFERENCES

[1] Spatial Clustering Methods in Data Mining: A Survey. Han et.al, School of Computing Science, Simon Fraser University, Burnaby, BC Canada.

[2] WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. Gholamhosein Sheikholeslami, Surojit Chatterjee and Aidong Zhang, Proceedings of the 24th VLDB Conference, New York, USA, 1998.

[3] Clustering for Data Reduction and Approximation. T. Schreier, Computer Graphics and Geometry Journal 1999-3. Dept. of Computer Science, University of Kaiserslautern, Germany

[4] K-means Clustering via Principal Component Analysis. Chris Ding, Xiaofeng He, Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720. Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.