Incremental Shared Nearest Neighbor Density Based Clustering

Sumeet Singh, Amit Awekar S.SUMEET,AWEKAR@IITG.ERNET.IN
Indian Institute of Technology, Guwahati, Assam, 781039

Abstract

Shared Nearest Neighbor Density-based (SNN-DBSCAN) clustering is a robust graph-based clustering algorithm and has wide applications from climate data analysis to network intrusion detection. We propose an incremental extension to this algorithm IncSNN-DBSCAN capable of finding clusters on a dataset to which frequent inserts are made. Computations are done only on those data points which undergo one of the three types of changes in properties that we have identified. IncSNN-DBSCAN achieves speed-up up to 360 times by avoiding redundant computations, while generating exact same clustering as the non-incremental algorithm. We experimentally verify our claim on a dataset with 1000 records upon which we make up to 5000 inserts.

1. Introduction

Cluster analysis organizes similar points in the data into groups called *clusters*. Popular clustering algorithms use the stationary nature of data to find a globally optimal solution. When changes are made to the dataset, these algorithms have to be run on the entire dataset to update possible changes in clustering, involving significant redundant computations. We specifically tackle this problem by restricting redundant computations and deliver a clustering identical to the original clustering. The speed-up gained will have tremendous impact in scenarios where changes to dataset are rather frequent like network intrusion detection or web-crawling.

Appearing in Proceedings of the 1st Indian Workshop on Machine Learning, IIT Kanpur, India, 2013. Copyright 2013 by the author(s).

2. Incremental Clustering

Definition 1: Given a data set A, along with its initial clustering $f : A \to \mathbb{N}$ and an update sequence of n data points. After $k \leq n$ updates let A' be the new data set, then an incremental clustering is defined as a mapping $g : f, A' \to \mathbb{N}$ isomorphic to the one-time clustering $f(A')$ by the non-incremental algorithm.

Output of online clustering algorithm can change depending on the order of insertions. However, incremental clustering requires algorithms to produce output which is independent of insertion order. Just like IncDBSCAN (Ester et al., 1998), our algorithm is order-independent.

3. Shared Nearest Neighbor Similarity

3.1. SNN-DBSCAN

Beyond direct similarity between points, similarity in surroundings of points is used as a more robust similarity measure - **Shared Nearest Neighbor** similarity. Two points are called shared nearest neighbors (SNN) if they are present in each others nearest neighbor lists (NN-list). If two SNNs p_1 and p_2, have more common neighbors in their NN-lists then shared link $p_1 \leftrightarrow p_2$ also strengthens. We chose an approach in (Ertöz et al., 2003), to measure shared link strengths, which is, the sum of the product of the ranks assigned by p_1 and p_2 to their common neighbors.
Two points p and q are said to have **SNN connection**, if they have a shared link with strength above a threshold called **merge threshold** and at least one of the points is a **topic point**. A point is a topic if its **total connection strength** (TCS) is greater than or equal to a threshold *topic_t*. TCS is the number of **strong links** incident on a point. Strong links are shared links with strength above another threshold called **strong threshold**, where strong threshold $\geq$ merge threshold.
SNN reachability exists between two points p and q if we can find a sequence of SNN connected points starting from p to q (or vice-versa). SNN reachability is symmetric and transitive. A cluster is maximal set of pairwise SNN reachable points.

3.2. Incremental SNN-DBSCAN

Due to an insertion three types of changes can be observed (please refer Figure 1) ($p \rightarrow q$ means q is in $p's$ NN-list, $p \leftrightarrow q$ means p, q have shared link)
1. (Type I) Changes in NN-list, shared link strengths, total connection strength and topic property.
2. (Type II) Changes in shared link strengths, total connection strength and topic property.
3. (Type III) Changes only in cluster membership.
Nodes not SNN-reachable from any of the affected points would not be involved in any computation, restricting redundancy. First, the inserted point *new_pt's* NN-list is constructed by scanning all other points in the data set. Existing data points compete to appear in NN-list of *new_pt* and vice versa. Points that accept *new_pt* into their NN-list fall in Type I. Such points have to delete one of the older members from their NN-list and degrade rank of others who rank below *new_pt*. This changes shared links, total connection strength and topic property of Type I points. NN-lists of non-Type I neighbors of Type I points is not changed by *new_pt*. However, the rank of non-Type I points could be degraded in NN-list of Type I points. This might result in weakening of shared link strengths between a Type I point and its Type II neighbors. As a result, total connection strength and topic property of Type II points will also change.
Non-Type I and non-Type II neighbors of Type II points are labeled as Type III points. For all Type III points, NN-list and shared links are unchanged. However, a Type II neighbor of a Type III point can change from a topic to a non-topic point. As a result, an earlier SNN connection between them could be lost. This opens up possibility of splitting the existing cluster that contains these points.
After sequentially updating the properties for *new_pt*, Type I , Type II and Type III points and marking all SNN-links to be removed, we perform **merge** and **split** operations on the clusters. Merging involves union of all clusters that are SNN connected through new SNN-connections with *new_pt* or Type I points. Merging need not extend to Type II points or beyond because Type II points can only degrade from a topic to non-topic. A cluster split between two points is decided by a simple SNN reachability test between them.

NN-list computed for each data point by our algorithm is same as the non-incremental algorithm. Every other property of data points and the final clustering is based on NN-lists of all the data points. Therefore, output of our algorithm is order-independent and matches the non-incremental algorithm.

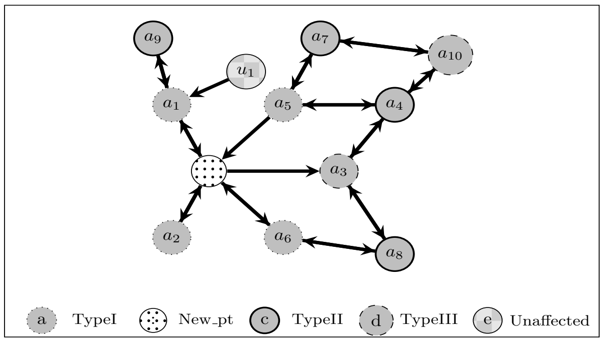
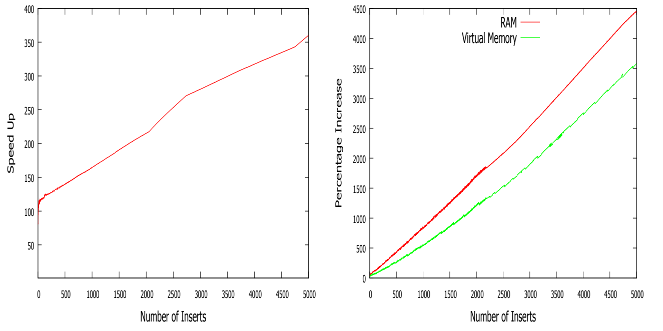


Figure 1. Layers Involved in the Algorithm



(a) Speed-up (b) Percentage rise in Memory

Figure 2. Performance vs Number of Inserts

3.3. Experiments and Results

Our experiments were performed on the Network Intrusion Data from KDD Cup 1999. Filtering out 10,000 unique records with 41 attributes, we measured the speed-up and percentage rise in memory usage from 20 to 5000 inserts (please refer Figure 2). Speed-up is defined as the ratio of time taken by SNN-DBSCAN to the time taken by our algorithm. On 5000 inserts we got 360 times speed-up over the SNN-DBSCAN algorithm which took 15 hours. However, our algorithm requires 45 times more memory.

References

Ertöz, L., Steinbach, M., and Kumar, V. Finding topics in collections of documents: A shared nearest neighbor approach. *Clustering and Information Retrieval*, 11:83–103, 2003.

Ester, M., Kriegel, H.P., Sander, J., M.Wimmer, and Xu, X. Incremental clustering for mining in a data warehousing environment. *Proceedings of the International Conference on Very Large Data Bases, IEEE*, 24:323–333, 1998.