Review of NG-DBSCAN: Scalable Density-Based Clustering for Arbitrary Data

GROUP 10 - algo_miners

SAMAY VARSHNEY - samay@iitg.ac.in - 180101097
PULKIT CHANGOIWALA - changoiw@iitg.ac.in - 180101093
SAI SUMANTH MADICHERLA - madicher@iitg.ac.in - 180101068
KOMATIREDDY SAI VIKYATH REDDY - komatire@iitg.ac.in - 180101036

ABSTRACT

NG-DBSCAN is an **approximated** and **distributed** density-based clustering algorithm that operates on **arbitrary data** and **any symmetric distance** measure. In this review, we provide an overview of the steps in the NG-DBSCAN algorithm along with its evaluation criteria and applications. The results are obtained through different experiments with **real and synthetic data**, proving the claims about NG-DBSCAN's performance and scalability.

INTRODUCTION

Clustering algorithms are fundamental in data analysis, providing an unsupervised way to aid understanding and interpreting data by grouping similar objects together.
DBSCAN introduced the idea of density-based clustering: grouping data packed in high-density regions of the feature space. DBSCAN has **two important features**: first, it separates "core points" appearing in dense regions of the feature spaces from outliers (noise points) which are classified as not belonging to any cluster; second, it recognizes clusters having arbitrary shapes rather than being limited to ball-shaped ones. But then also, DBSCAN has many limitations which we have discussed in the next section.
Even though several distributed DBSCAN implementations exist: they partition the feature space, running a single-machine DBSCAN implementation on each partition, and then "stitch" the work done on the border of each partition, all these approaches are effective only when dimensionality is low and also are not able to handle any kind of heterogeneous data.

MAJOR BOTTLENECKS SOLVED BY NG-DBSCAN

The proposed algorithm solves the **DBSCAN scalability problem of handling large databases**, the **inconsistency of working with heterogeneous data sets**: through a modified implementation of DBSCAN which is approximated, scalable and distributed, supporting any arbitrary data and any symmetric distance measure. The modified implementation so-called NG-DBSCAN algorithm, in some of the cases even outperforms competing for DBSCAN implementations while the approximation imposes small or negligible impact on the results.

The problem with DBSCAN was that in high dimensional datasets, it partitions the feature space and then merges the spaces which lead to high computational complexity in large datasets. Also when applied to arbitrary distance measures, it requires retrieving each point's ε-neighborhood, for which the distance between all node pairs needs to be computed, resulting in $O(n^2)$ calls to the distance function. NG-DBSCAN on the other hand follows a vertex centric approach in which computation is partitioned by and logically performed at the vertices of a graph, and vertices exchange messages whereby building a neighbour and ε-graph and clusters are

built based on the neighbour graph content. This approach enables distribution without needing Euclidean spaces to partition. In most of the existing DBSCAN algorithms where $d >= 6$ or n is high, it is computationally infeasible to run the algorithm either due to memory errors or large time complexity. However the algorithm mentioned in the paper named NG-DBSCAN, is **independent of the dimensionality** of the dataset and is able to run in the time **linear** to the number of data points.

NG-DBSCAN ALGORITHM: AN OVERALL DESCRIPTION AND FLOWCHARTS

The NG-DBSCAN algorithm happens in two phases. Dataset is represented in the form of a graph where each node or vertex is a data point and edges represent the similarity distance measure between points. In the algorithm we are having ε, $Minpts$, $M_{\max}$, ρ, T_{n}, T_{r}, k as parameters and these are tuned in accordance with the number of data points and as per most efficient computational complexity. NG notation is used for denoting Neighbour Graph here.

(1) **First Phase:** It is implemented through a neighbour graph which converges to k-nearest neighbour graph. It creates the ε-graph and avoids the ε-neighbourhood queries (which were creating high computational cost in DBSCAN).
The ε-neighborhood of a point p is the set of points within distance ε from p.
ε-neighbourhood queries for a given vertex is finding nodes that are within the ε-distance which can take upto $O(n^2)$ if performed naively.
ε-graph nodes are data points where each node's neighbors are a subset of its ε-neighborhood. At each iteration, all pairs of nodes (x, y) separated by 2 hops in the neighbor graph are considered and if their distance is less than ε, then this edge is added in ε-graph. To speed up the computation, nodes with at least M_{max} neighbors are removed from the neighbour graph.

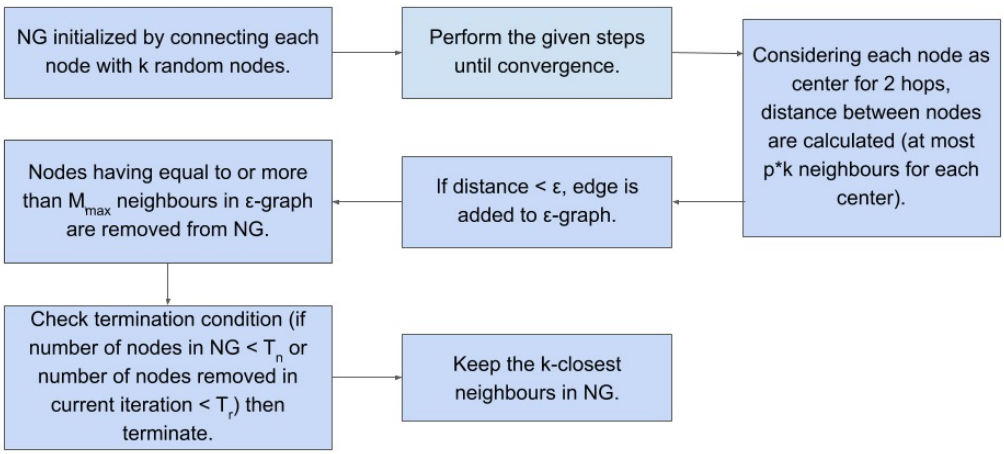

Fig. 1. Overview of Phase 1

(2) **Second Phase:** Second phase takes ε-graph as input to build a clustering and neighbour lookups are performed instead of ε-neighbourhood queries. All nodes are given different roles in the ε-graph.
Core nodes are those having at least $MinPts - 1$ neighbours, Border nodes are those having at least 1 core node as neighbour while Noise nodes are the remaining nodes. Each node coreness is then referred to as $(degree, nodeID)$. This labeling with degree is called coreness dissemination of the graph. Seed of a cluster is called a node with highest coreness. The

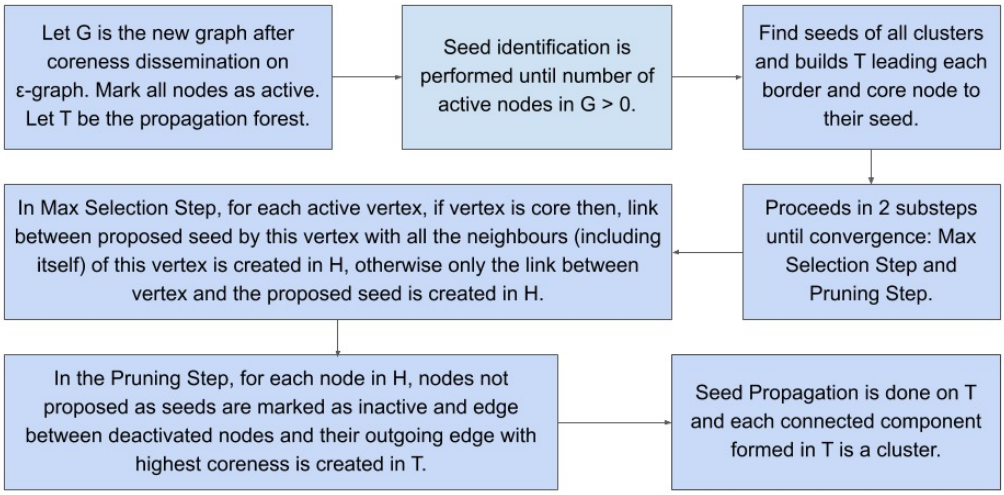

Fig. 2. Overview of Phase 2

Propagation forest is created having seed as root for each cluster and from that different clusters are created.

EVALUATION OF THE ALGORITHM

The evaluation of the algorithm was carried out by running it on both synthetically generated and externally used datasets and comparing the results with exact DBSCAN, by comparing its scalability against SPARK-DBSCAN and IRVINGC-DBSCAN. **Quality metrics such as compactness, separation, recall and speed-up** are used to distinguish.

Compactness measures how closely are items in each cluster. Separation measures how well items in different clusters are separated. Recall of a cluster is the fraction of the node pairs that are in the same cluster with that in a reference cluster. Speed-Up metric measures the algorithm runtime improvement when increasing the number of cores dedicated to the computation.

Since computing the above metrics is computationally hard, data points are picked randomly with uniform sampling and averaging independent runs for each data point.

In **2-dimensional space**, clustering quality is compared by checking time taken and recall on different datasets and scalability is measured by taking different dataset sizes and number of cores used.

In **d-dimensional space**, both clustering recall and time taken are compared with the number of dimensions used in different datasets.

In **textual data**, NG-DBSCAN (using both *word2vec* format and *jaro winkler's* edit distance separately) is compared with k-means (for k-means data was converted to word2vec format), using the compactness, separation and time taken metrics.

High compactness, less separation, less time taken was observed in most of the NG-DBSCAN cases in comparison with other DBSCAN algorithms irrespective of the data which proves that clusters are more separated, dense and efficiently computable in case of NG-DBSCAN as seen in the results of the paper.

EXPERIMENT: CODE AND DATASETS

We were not able to find the **Twitter** dataset and the **Spam** dataset used by the author. But we have *mailed* the author asking for the same.

We were only able to find the *implementation* of NG-DBSCAN in Java with Apache Spark framework.

REAL WORLD APPLICATIONS

NG-DBSCAN being a new algorithm (2016), we could not find any real-world applications that claim to use NG-DBSCAN. However, there are ample applications for it's parent algorithm "DBSCAN". And wherever DB-SCAN is used, NG-DBSCAN can be used there. Some applications of DBSCAN (and NG-DBSCAN) are:

- A widespread application of DBSCAN is the geographical clustering, where the goal is to cluster geo points having geo coordinates latitude, longitude. This will be very helpful in the following cases:
  - To *determine* geographical areas that are specific and personal to each user and look at how to build location-based services by extracting users' geographical regions from numerous geolocated events, such as check-ins in restaurants or cafes. Such a system could identify, for instance, areas that a given user typically frequents for dinner outings.
  - When we have too much of *geo-spatial* data, we might need to reduce the size of a data set down to a smaller set of spatially representative points. Here we can use NG-DBSCAN to cluster them down to a smaller dataset.
- Based on previous shows you have watched in the past, Netflix will *recommend* shows for you to watch next. This can be done through NG-DBSCAN clustering of people who have watched similar shows. In fact, this is a very popular use of clustering that we all might be familiar with.
- It can be used to cluster all the emails which are similar to each other in some form which can be used as a symmetric distance measure in the algorithm for clustering.
- It can be used to cluster all the cricketers who are similar in their batting techniques using a d-dimensional dataset where each field in d-dimension is a probability value of playing that kind of shot by the player and taking each player as a data point.

We have not found any ML/Data Analysis Package that includes NG-DBSCAN algorithm.

CONCLUSION

We presented a review of NG-DBSCAN, a novel distributed algorithm for density-based clustering that produces quality clusters with arbitrary distance measures. This allows domain experts to choose the similarity function appropriate for their data, and parallelism can be addressed by designers. We showed an overview of the steps involved in the algorithm and how it is evaluated. We found out that this algorithm has lots of real-life applications and mention some of them in the review report as well.

Our next steps will be to understand the algorithm in-depth and start its implementation. Then we will propose our iterations to make it an incremental clustering algorithm.