Peer Review

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

Contents

PEER REVIEW 1: GROUP 9 ROCK: A ROBUST CLUSTERING ALGORITHM FOR CATEGORICAL ATTRIBUTES

SHORT SUMMARY OF THE REPORT

Report starts with the introduction of the ROCK algorithm; then Section 2 describes the overview of static version of the algorithm; Section 3 covers the related work; Section 4 explains the incremental algorithm; Section 5 is about code structure; Section 6 and Section 7 presents data sets and evaluation results respectively.

'ROCK' is a clustering algorithm that uses the idea of linking data points instead of clustering them on the basis of distance between data points. The two key steps in the static algorithm are the computation of links and merging of clusters based on goodness measure. The incremental version of ROCK clustering algorithm adds new points to the original database in batches. Since the data set can be large, first a random sample is drawn from the data set. After that, the ROCK algorithm is run over the sampled points that employ links. Finally, the remaining data points on the disk are allocated to the relevant clusters.

Initially, each point is represented as a single cluster. For each of the cluster, i, we maintain a local heap which contains every other cluster j, such that $link[i, j]$ is non zero. The local heap is arranged in descending order on the basis of their goodness value. During each iteration, the pair of clusters with the highest goodness is merged and the goodness measure of the corresponding clusters is updated. The algorithm terminates either when the desired number of clusters has been reached or when all the clusters are well separated.

Incremental Algorithm: When a batch of b points is added to old k clusters we obtain total of $b + k$ clusters. For each of the b new cluster, we find cluster with the best goodness measure to merge, algorithm is repeated till there are again k clusters. After this, outliers are removed from the clusters.

Two real-life data sets Mushroom and Congressional Votes Data sets were used for testing ROCK and incremental ROCK, containing only categorical attributes. The metrics used for result comparison are running time, memory footprint, clustering quality and data distribution in clusters.
For Mushroom data set, 1500 points were processed statically and 6624 were added incrementally while for Congressional Votes Data set, 350 were processed statically and 85 were added incrementally.

Different parameters like θ, k, n affects the clustering quality in different ways. On increasing θ, the number of links in the data set decreases and quality of links increases. For best clustering, k should be close to the number of the class labels. On increasing the number of points added, clustering quality decreases.

KEY STRENGTHS OF THE PROJECT

- The project explained why no existing incremental algorithm exists for ROCK even though the ROCK is a popular clustering algorithm and was presented in the year 2000.
- The proposed model results are highly similar to the original ROCK results with random sampling, to the point where the size of clusters is very similar. For example, in the Mushroom

dataset result, 17 out of 20 clusters have the same number of data points. Moreover, the size of the remaining clusters is also very similar to the ROCK algorithm.
- The proposed model explored the idea of batch insertion.
- The incremental model performs significantly better than static ROCK with high precision.

KEY WEAKNESSES OF THE PROJECT

- The incremental algorithm does not include deletion.
- There was a slight trade-off in time complexity for ease of implementation in the py-clustering library. The study implemented static ROCK based on the py-clustering library.
- It's accuracy and clustering quality decreases significantly on increasing the number of incrementally added points and size of the dataset.

SUGGESTED IMPROVEMENTS IN THE PROJECT

- The use of adjacency list instead of adjacency matrix can be explored to increase space efficiency.
- An improved similarity measure like the one suggested in QROCK can be used.
- Could have implemented Static ROCK parallely (like DROCK) instead of sequentially to improve the time complexity.
- max_goodness could be calculated efficiently by using some other data structures like set or priority queue.
- Clusters are considered outliers if its size is less than or equal to five percent of the largest cluster. This value of five percent seems to have found through guess work. A better formulation or description of how one came up with this value would have improved the project.

SUGGESTED IMPROVEMENTS IN THE REPORT

- Report should contain a toy dataset over which we can verify that the algorithm runs correctly
- The flowchart can be made simpler by removing the use of C++ variables.
- Some of the urls can be made links which on clicking will direct to that url.
- Correctness of the algorithm should be included in the report.
- There should be some sources in the report (like email) through which one can contact the authors.
- Description of various parameters like f(θ), θ, m_a and m_b is missing from the report. Adding lucid description of the parameters aids in understanding the algorithm well.
- Clustering results are compared in tabular form by listing each clusters' description, this way seems inefficient and time consuming, a graphical way or some single value metric calculation would have been better.

OVERALL RATING

On analyzing the report of Group 9, we found the overall report and idea to be much satisfactory and worthy, hence for calculating overall rating we have divided the rating as follows.

- **Key strengths of the project:** We found strengths of the algorithm promising, the results were significantly better than static ROCK and time taken was comparatively smaller. Thus we are rating strengths 2.5 out of 2.5.
- **Key weaknesses of the project:** The main weakness of the project is that it does not contains methods for point deletion. As deletion is a crucial part of dynamic data set, thus

absence of it makes the algorithm incomplete. Therefore, we are rating algorithm 1.8 out of 2.5 in this aspect.

- **Suggested improvements in the project:** As a good project has very little scope of improvements but we found few very significant improvements which can be done in the project. Thus, we are rating it 2.2 out of 2.5.
- **Suggested improvements in the report:** Report has a lot of scope for improvements. Report could have been made more clear and lucid. Thus, we are rating it 2.0 out of 2.5.

Thus, overall rating of the project is $2.5 + 1.8 + 2.2 + 2.0 = 8.5/10$.

PEER REVIEW 2: GROUP 8
AN INCREMENTAL STRUCTURAL CLUSTERING ALGORITHM FOR NETWORKS

SHORT SUMMARY OF THE REPORT

Report starts with the introduction of SCAN; Section 2 gives a review of SCAN; Section 3 gives a review of related works; Section 4 demonstrates the proposed Incremental algorithm. Results and their analysis are given in Section 5. Lessons learnt and future scope of the project have been outlined in Sections 6 and 7 respectively.

The project proposes an incremental version of the SCAN algorithm. The SCAN algorithm can not only identify clusters but can also identify hubs and outliers. It works well for static graphs, but most modern real-world graphs are dynamic; hence it is very inefficient for dynamic graphs. The incremental version of SCAN works for dynamic networks, called ISCAN.

SCAN uses the neighborhood similarity of vertices as a clustering criterion instead of direct edges. Only edges with structural similarity higher than epsilon are considered. The algorithm then identifies core vertices which are vertices with more than μ neighbors, and starts creating clusters from these cores using a BFS algorithm. After the clustering is completed, the remaining vertices are classified as either hubs or outliers based on their neighbours. With n vertices and m edges, it has a running time of $O(m^{1.5})$.

Incremental SCAN gives an option of edge and vertex addition/deletion and it works by avoiding recomputation of all the SCAN-generated clusters. On an edge addition or deletion, the structural similarities of only a subset of edges are affected. This ensures that only a limited number of edges need to be checked which drastically reduces the computation complexity. A decrease in structural similarity of an edge can result in the splitting of clusters, while an increase in structural similarity of an edge may result in the merger of two clusters. Once these edges similarities are updated, the clusters can be modified by adding or removing these edges from the BFS trees. With n vertices and m edges, the time complexity of the incremental algorithm for each edge updation is $O(m + nlogn)$.

In most of the datasets used, ISCAN outperforms SCAN in terms of running time. While multithreading SCAN offers a major improvement in SCAN run time, multithreading incremental SCAN did not achieve much improvement. While performing some parameter tuning for SCAN, it was also observed that the run time decreases as epsilon increases.

KEY STRENGTHS OF THE PROJECT

- Multithreading operations were implemented which speeds up the computation through multiple threads for similarity calculations.
- Proper algorithm codes, flowcharts and definitions allows the reader to properly understand the base of the suggested static and incremental algorithm.
- The incremental version performs significantly better than its static counterpart.
- The project has also presented the drawbacks of the Incremental version of SCAN and explained the need of a new incremental algorithm for SCAN.

KEY WEAKNESSES OF THE PROJECT

- Mostly memory and time taken are used to compare the incremental and static SCAN.

- The memory storage overhead for maintaining the data structures is visible, despite the fact that the extra memory required for each increment is minimal.

SUGGESTED IMPROVEMENTS IN THE PROJECT

- Metrics to compare the accuracy of incremental and static SCAN could have given a clear picture of the cluster quality.
- Could have further optimized similarity value calculations using some of the related works like pruning techniques used in *pSCAN*.
- Comparison could be made between some incremental variants of SCAN with ISCAN between time and memory which could help to determine how important ISCAN is.

SUGGESTED IMPROVEMENTS IN THE REPORT

- Observations for the algorithm are stated as it is, by taking reference from ISCAN. A proof or an intuition of the proof would have added more clarity in the report.
- Report should contain a toy dataset over which we can verify that the algorithm runs correctly.
- Clusters when compared using some graphical methods would help in comparing the accuracy even better.
- Some intuition and discussion about the parameters like μ and ϵ is missing from the report which is necessary.

OVERALL RATING

On analyzing the report of Group 8, we found the overall report and idea to be much satisfactory, clear and worthy, hence for calculating overall rating we have divided the rating as follows.

- **Key strengths of the project:** We found strengths of the algorithm promising, the results were significantly better than static SCAN and time taken was comparatively smaller. Thus we are rating strengths 2.5 out of 2.5.
- **Key weaknesses of the project:** The main weakness of the project is that it contains less metrics for comparison of incremental SCAN with other versions. Therefore, we are rating algorithm 2.4 out of 2.5 in this aspect.
- **Suggested improvements in the project:** As a good project has very little scope of improvements but we found few very significant improvements which can be done in the project. Thus, we are rating it 2.3 out of 2.5.
- **Suggested improvements in the report:** Report has some few scope for improvements. Thus, we are rating it 2.3 out of 2.5.

Thus, overall rating of the project is $2.5 + 2.4 + 2.3 + 2.3 = 9.5/10$.