DENISTY AND PARTITION BASED CLUSTERING ON MASSIVE THRESHOLD BOUNDED DATA SETS.

by

ARUNA SAI KANNAMAREDDY

Bachelor of Technology in Information Technology, Jawaharlal Nehru Technological University, India, 2013

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science College of Engineering

KANSAS STATE UNIVERSITY, Manhattan, Kansas

2016

Approved by:

Major Professor Dr. William H.Hsu

## COPYRIGHT

ARUNA SAI KANNAMAREDDY

2016

## ABSTRACT

The project explores the possibility of increasing efficiency in the Clusters formed out of massive data sets which are formed using threshold blocking algorithm. Clusters thus formed are denser and qualitative. Clusters that are formed out of individual clustering algorithms alone, donot necessarily eliminate outliers and the clusters generated can be complex, or improperly distributed over the data set. Threshold blocking algorithm, a current research paper from Michael Higgins of Statistics Department on other hand, in comparison with existing algorithms performs better in forming the dense and distinctive units with predefined threshold. Developing a Hybridized algorithm by implementing the existing clustering algorithms to re-cluster these units thus formed is part of this project.

Clustering on the seeds thus formed from threshold blocking Algorithm, eases the task of clustering to the existing algorithm by eliminating the overhead of worrying about the outliers. Also, the clusters thus generated are more representative of the whole. Also, since the threshold blocking algorithm is proven to be fast and efficient, we now can predict a lot more decisions from large data sets in less time. Predicting the similar songs from Million Song Data Set using such a hybridized algorithm is considered as the data set for the evaluation of this goal.

# 

# Table of Contents

[Table of Contents](#_Toc477907812)

[List of Figures](#_Toc477907813)

[List of Tables](#_Toc477907814)

**Acknowledgements**

1. **Introduction**
2. **Background and Related Work** 
   1. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   2. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   3. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
3. **Proposed Models 7**
   1. K-Means with Threshold Blocking Algorithm. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      1. . . . . . . . . . . . . . . . . . . . . . . . .
      2. . . . . . . . . . . . . . . . . . . . . . . . . . .
      3. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   2. DBSCAN with Threshold Blocking Algorithm. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      1. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      2. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      3. . . . . . . . . . . . . . . . . . . . . . . . . . . . .

|  |  |
| --- | --- |
| **4 Experiments** |  |
| 4.1 Dataset Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . |  |
| 4.3 Experimental Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . |  |
| 4.3.1 K-Means with Threshold Blocking Algorithm. . . . . . . . . . . . . . . . . . . . . . . . . |  |
| 4.3.2 DBSCAN with Threshold Blocking Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . |  |
| 4.3.3 Performance Evaluation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . |  |
| **5 Results** |  |
| **Bibliography** |  |
| **A Title for This Appendix** |  |
| **B Technologies** |  |
|  |  |

# List of Figures

# List of Tables

**ACKNOWLEDGMENTS**

**CHAPTER 1**

## INTRODUCTION

Improving massive experiments with threshold Blocking paper describes a novel approach for sampling algorithm to any covariate dataset.

The goal of this project aims at analyzing the performance variations of the mentioned threshold blocking algorithm in association with prominent clustering algorithms. In order to achieve this goal, the following sequence is carried out.

Initially , the threshold Blocking algorithm is used along with Clustering Algorithm to form clusters among the dataset. Since blocks are already similar units, clustering algorithm on those blocks form more efficient clusters. For this purpose, DBSCAN (Density-based spatial clustering of applications with noise) and k-means clustering algorithms are used in this project. The clusters formed from DBSCAN in association with Threshold blocking algorithm are compared against the clusters formed from running DBSCAN alone on the data. In the similar way, the clusters formed from the k-means algorithm are compared against the clusters formed from running only k-means on the data. Thus the performance and compatibility of the algorithm with prominent clustering algorithms is analyzed.

**CHAPTER 2**

## LITERATURE SURVEY

DBSCAN Algorithm

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a density-based clustering algorithm. Given a set of points in some space, the algorithm groups together points that are closely packed together i.e., points with many nearby neighbors while marking as outliers points that lie alone in low-density regions whose nearest neighbors are too far away.

The brief description of the algorithm is as follows:-

Given a set of points in some space which are to be clustered, they are classified as core points, density reachable points and outliers.

* Core Points:- A point *p* is a core point if at least minPts points are within distance *ε*(*ε* is the maximum radius of the neighborhood from *p*) of it (including *p*). Those points are said to be *directly reachable* from *p*. By definition, no points are *directly reachable* from a non-core point.
* Reachable Points:- A point *q* is reachable from *p* if there is a path *p*1, ..., *pn* with *p*1 = *p* and *pn* = *q*, where each *pi*+1 is directly reachable from *pi* (all the points on the path must be core points, with the possible exception of *q*).
* Outliers:- All points not reachable from any other point are outliers.

Now if *p* is a core point, then it forms a *cluster* together with all points (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points.

A cluster then satisfies two properties:

1. All points within the cluster are mutually density-connected where two points *p* and *q* are density-connected if there is a point *o* such that both *p* and *q* are density-reachable from *o*.  Density-connectedness *is* symmetric.
2. If a point is density-reachable from any point of the cluster, it is part of the cluster as well.

Algorithm code

Applications

Pros and Cons

Comparison with other algorithms in terms of performance

1. DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means.
2. DBSCAN can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster. Due to the MinPts parameter, the so-called single-link effect (different clusters being connected by a thin line of points) is reduced.
3. DBSCAN has a notion of noise, and is robust to outliers.
4. DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database. (However, points sitting on the edge of two different clusters might swap cluster membership if the ordering of the points is changed, and the cluster assignment is unique only up to isomorphism.)
5. DBSCAN is designed for use with databases that can accelerate region queries, e.g. using an R\* tree.
6. The parameters minPts and ε can be set by a domain expert, if the data is well understood.

Complexity

DBSCAN visits each point of the database, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the time complexity is mostly governed by the number of regionQuery invocations. DBSCAN executes exactly one such query for each point, and if an indexing structure is used that executes a neighborhood query in O(log n), an overall average runtime complexity of O(n log n) is obtained (if parameter ε is chosen in a meaningful way, i.e. such that on average only O(log n) points are returned). Without the use of an accelerating index structure, or on degenerated data (e.g. all points within a distance less than ε), the worst case run time complexity remains O(n²). The distance matrix of size (n²-n)/2 can be materialized to avoid distance recomputations, but this needs O(n²) memory, whereas a non-matrix based implementation of DBSCAN only needs O(n) memory.

<http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf>

<https://en.wikipedia.org/wiki/DBSCAN>

K-Means Algorithm

**CHAPTER 3**

**DATA SET**

The Million Song Dataset is a freely-available collection of audio features and metadata for a million contemporary popular music tracks. The dataset contains only the feature analysis and metadata for one million songs but not the audio provided by [The Echo Nest](http://the.echonest.com/). The size of the entire dataset is around 280GB containing almost one million song records. The features of each record in the dataset consists of the following features.

|  |
| --- |
| artist\_mbid: db92a151-1ac2-438b-bc43-b82e149ddd50  the musicbrainz.org ID for this artists is db9...  artist\_mbtags: shape = (4,)  this artist received 4 tags on musicbrainz.org  artist\_mbtags\_count: shape = (4,)  raw tag count of the 4 tags this artist received on musicbrainz.org  artist\_name: Rick Astley  artist name  artist\_playmeid: 1338  the ID of that artist on the service playme.com  artist\_terms: shape = (12,)  this artist has 12 terms (tags) from The Echo Nest  artist\_terms\_freq: shape = (12,)  frequency of the 12 terms from The Echo Nest (number between 0 and 1)  artist\_terms\_weight: shape = (12,)  weight of the 12 terms from The Echo Nest (number between 0 and 1)  audio\_md5: bf53f8113508a466cd2d3fda18b06368  hash code of the audio used for the analysis by The Echo Nest  bars\_confidence: shape = (99,)  confidence value (between 0 and 1) associated with each bar by The Echo Nest  bars\_start: shape = (99,)  start time of each bar according to The Echo Nest, this song has 99 bars  beats\_confidence: shape = (397,)  confidence value (between 0 and 1) associated with each beat by The Echo Nest  beats\_start: shape = (397,)  start time of each beat according to The Echo Nest, this song has 397 beats  danceability: 0.0  danceability measure of this song according to The Echo Nest (between 0 and 1, 0 => not analyzed)  duration: 211.69587  duration of the track in seconds  end\_of\_fade\_in: 0.139  time of the end of the fade in, at the beginning of the song, according to The Echo Nest  energy: 0.0  energy measure (not in the signal processing sense) according to The Echo Nest (between 0 and 1, 0 => not analyzed)  key: 1  estimation of the key the song is in by The Echo Nest  key\_confidence: 0.324  confidence of the key estimation  loudness: -7.75  general loudness of the track  mode: 1  estimation of the mode the song is in by The Echo Nest  mode\_confidence: 0.434  confidence of the mode estimation  release: Big Tunes - Back 2 The 80s  album name from which the track was taken, some songs / tracks can come from many albums, we give only one  release\_7digitalid: 786795  the ID of the release (album) on the service 7digital.com  sections\_confidence: shape = (10,)  confidence value (between 0 and 1) associated with each section by The Echo Nest  sections\_start: shape = (10,)  start time of each section according to The Echo Nest, this song has 10 sections  segments\_confidence: shape = (935,)  confidence value (between 0 and 1) associated with each segment by The Echo Nest  segments\_loudness\_max: shape = (935,)  max loudness during each segment  segments\_loudness\_max\_time: shape = (935,)  time of the max loudness during each segment  segments\_loudness\_start: shape = (935,)  loudness at the beginning of each segment  segments\_pitches: shape = (935, 12)  chroma features for each segment (normalized so max is 1.)  segments\_start: shape = (935,)  start time of each segment (~ musical event, or onset) according to The Echo Nest, this song has 935 segments  segments\_timbre: shape = (935, 12)  MFCC-like features for each segment  similar\_artists: shape = (100,)  a list of 100 artists (their Echo Nest ID) similar to Rick Astley according to The Echo Nest  song\_hotttnesss: 0.864248830588  according to The Echo Nest, when downloaded (in December 2010), this song had a 'hotttnesss' of 0.8 (on a scale of 0 and 1)  song\_id: SOCWJDB12A58A776AF  The Echo Nest song ID, note that a song can be associated with many tracks (with very slight audio differences)  start\_of\_fade\_out: 198.536  start time of the fade out, in seconds, at the end of the song, according to The Echo Nest  tatums\_confidence: shape = (794,)  confidence value (between 0 and 1) associated with each tatum by The Echo Nest  tatums\_start: shape = (794,)  start time of each tatum according to The Echo Nest, this song has 794 tatums  tempo: 113.359  tempo in BPM according to The Echo Nest  time\_signature: 4  time signature of the song according to The Echo Nest, i.e. usual number of beats per bar  time\_signature\_confidence: 0.634  confidence of the time signature estimation  title: Never Gonna Give You Up  song title  track\_7digitalid: 8707738  the ID of this song on the service 7digital.com  track\_id: TRAXLZU12903D05F94  The Echo Nest ID of this particular track on which the analysis was done  year: 1987  year when this song was released, according to musicbrainz.org |

Since the project aims to identify similar songs to group them into genres, only few fields among all the above fields are sufficient for the task. Loudness, Tempo, Time\_Signature, Duration and Key are the fields that will be used in this project. So, the million records consisting only these fields is used in the Experiment.

Source:- https://labrosa.ee.columbia.edu/millionsong/

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere.

The Million Song Dataset. In Proceedings of the 12th International Society

for Music Information Retrieval Conference (ISMIR 2011), 2011.

Data Filtering:-

The data is distributed using hdf5 files which are converted to .csv extension files using python wrapper. The created .csv files are further filtered to retrieve only the required parameters for the analysis of given problem. Thus, from the .csv files the fields loudness, tempo, time\_signature, duration and key are filtered to form the dataset used in the prediction task.

Cross- Validation:-

Cross-validation is used to validate the model to check how the statistical analysis results will generalize to an independent data set.It is used here to estimate how accurately our predictive model will perform in practice. Usually in a supervised learning for a prediction problem, the known set of data (i.e., data with cluster labels) is partitioned into training data and testing data. The model is trained on training data and is validated against testing data. This is done in order to avoid problems like overfitting and will give an insight on how the model will generalize to an independent dataset. In case of unsupervised learning for prediction problem, the dataset donot contain labels to follow the same approach. So, the crossvalidation is done against the error rate on clustering results of training data and testing data.

In the project, intially 75% of data is taken as training data and 25% of data is taken as testing data. The hybridized algorithm model is trained on the training data and thus, formed model is used to predict the cluster number for the testing data. This prediction donot change the cluster center formed out of the model. The inter-cluster and intra-cluster average distances for the clusters are used as measures to validate the system. These measures are calculated for the clusters that are formed from training data is validated against the clusters that are formed after merging the test data with training data model clusters. The measures are to be similar in order to avoid overfitting of the model.

Feature Selection:-

The original Million Song dataset donot contain any labels or genre information. The goal of clustering task in the project is to predict the genre label of each song. To identify the features corresponding to the task on given data set, a feature selection algorithm PCA or Random forest is chosen. An alternative data set exists which contains partial data from the Million Song Dataset along with genre labels. This dataset is chosen to identify the predictors.

**CHAPTER 4**

**PROPOSED MODEL**

Project:- The project aims to study the performance of threshold algorithm in association with clustering algorithms. The focus is on how the efficiency and accuracy of the clustering algorithm is bolstered by the use of threshold blocking algorithm as a preprocessing step. Threshold Blocking Algorithm published in the paper, when combined with either K-Means or DBSCAN for clustering purpose is referred to as hybridized clustering algorithm. In order to obtain the performance metrics such as efficiency and accuracy, the hybridized clustering algorithm is compared against the clustering algorithm. Data is run individually on both on hybridized clustering algorithm and as well as on the clustering algorithm. The clusters thus formed out of both the approaches are evaluated against parameters like inter-cluster distance, intra-cluster distance, Silhoutte Coefficient and similarity between the cluster outputs. This project uses K-Means and DBSCAN as clustering algorithms to study the performance threshold blocking algorithm by carrying out the below two experimental approaches:-

1. Threshold Blocking Algorithm with K-Means Vs K-Means
2. Threshold Blocking Algorithm with DBSCAN Vs DBSCAN

The project uses Million Song Dataset to evaluate the performance in both the experiments. The Million Song Dataset which is a covariate data is chosen for this experiment and the clusters thus formed out of this data depict the similarity between the songs in the dataset. The clusters formed by running the clustering algorithms are expected to produce different genre sets, where each cluster is representative of a genre. So, all the songs in a cluster belong to a genre which is different from the genre of a point belonging to different cluster. As the dataset contains only 13 genres, we run the clustering algorithms to divide the data into 13 clusters.

Initially, Million Song dataset is given to threshold blocking algorithm to form clusters such that each cluster contains minimum number of elements equal to threshold value. These samples are closely connected points in multi-dimensional space. The threshold value ensures that data is divided into samples, where each sample consists of points with high similarity measure between any two points in the sample. The centroid calculated for the sample represents the characteristics of sample as a whole. The centroids calculated from each of these clusters is given to both K-means Algorithm and DBSCAN Algorithm. So, the project consists of two parts. Firstly, we analyze the performance and validate the results of hybridized algorithm consisting of Threshold Blocking Algorithm and K-means Algorithm. Secondly, the same steps are repeated against ThresholdBlocking Algorithm and DBScanAlgorithm.

K-Means with Threshold Blocking Algorithm: -

The Million Song Data set is initially clustered using a random k value by Threshold Blocking Algorithm. The centroids of the above clusters formed out of this algorithm is given as input to K-Means Algorithm. The K-Means algorithm is made to divide these centroids into 13 clusters where each cluster representing the genre. The centroid of the sample and the points corresponding to the sample are clustered into the same cluster consisting of the centroid of the sample. Since, the centroid of the sample represents it as a whole, the points of the sample as well can be clustered into the same cluster as the centroid. Thus, all the records of the data set are divided into 13 clusters.

On the other hand, the entire data set is given to K-Means for cluster analysis. The clusters thus formed using K-Means are compared against the clusters formed by above hybridized algorithm to check how many points overlap and how many points donot overlap. Also, with various values of k , the change in intra-cluster and inter-cluster distances, time, memory and other such cluster evaluation factors are used to depict the performance of hybrid algorithm.

DBSCAN with Threshold Blocking Algorithm:-

In the DBSCAN, the first step of sampling based on the k value remains same as above. The Million Song Dataset in divided into samples or clusters consisting of minimum k points in each sample. The centroids of these samples are passed to DBSCAN for analysis. The minimum number of points and epsilon values that are to be given as input to DBSCAN are determined using KNNDistplot. These values and the centroids of the samples are passed to DBSCAN for cluster analysis.

DBSCAN donot require the number of clusters to be given to the algorithm. DBSCAN generates clusters of arbitrary number representing the genres. The cluster evaluation metrics like intra-cluster and inter-cluster distances, time, memory are calculated for the generated clusters. These metrics are calculated for every instance of k value that is passed to algorithm. A range of k values are chosen to be given as input to the threshold blocking algorithm like in kmeans to check the performance variance over various values of k.

DBSCAN is also run on the dataset without any processing step of threshold blocking algorithm. The clusters thus generated are used to compare the similarity with the clusters generated by DBSCAN and threshold blocking algorithm. The metrics of generated clusters are also computed which are compared with the hybridized algorithm for every instance of k.

Data Flow diagram:-

REFERENCES:-

Introduction to Data Mining by Pang-Ning Tan, Michael Steinbach, Vipin Kumar.

<http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf>