Application of combinations of algorithm to use in Link Prediction in Sparse Matrix

Pratyush Kumar Singh pratyush.kumarsingh.cse20@heritageit.edu.in
Rakesh Kumar rakesh.kumar.cse20@heritageit.edu.in
Ashish Jha ashish.jha.cse20@heritageit.edu.in
Jagrit Drolia jagrit.drolia.cse20@heritageit.edu.in

Under the guidance of
Professor Lopamudra Dey
lopamudra.dey@heritageit.edu

Department of Computer Science and Engineering
Heritage Institute of Technology, Kolkata, West Bengal

Abstract- **We propose to solve the link prediction problem in sparse matrix using clustering with association rule mining. The model learns latent features from the sparse matrix, and makes better predictions.**
To present the effect of clustering the data onto the association rules. Hence, we have compared the results of two different approaches: Finding association rules without consumer segmentation, and with consumer segmentation. The data analysis framework is applied to the data of Online Retail. By extracting the most important information from Online Retail data, we claim that this framework provides offers, the right product/advertisement to the right consumer.
Predicting from a sparse matrix using algorithms like ARM are quite expensive and it takes a lot of time and space but using Clustering with ARM we can predict in Big O(n)+2k time where k is frequent item sets.

Keywords- Link prediction, matrix factorization, side information.

I. INTRODUCTION

The main goal of data mining is to define a process for discovering significant patterns or anomalies in a large volume of data. It has been applied to decision support problems in diverse areas such as medical diagnosis, targeted marketing, bioinformatics, sociology, networking, and information security, making data mining one of the most widely studied topics in intelligent systems. Data mining incorporates theory and practical developments from many older research areas such as databases, machine learning, artificial intelligence, distributed computing, information retrieval, and statistics, and lends an integrative perspective to these research areas. Due to the breadth of both applications and foundational theory in data mining research, it is often divided along methodological lines, into tasks such as classification, clustering, association, decision support, and visualization.[2] Association rule mining, Apriori, K-Means are subtopics which have been explored by many [1] research groups. It addresses the problem of discovering relationships between instances that originate from dependence or interaction

There are various problem which are there in computer science regime one of them is factorizing a Spare matrix.

Sparse Matrix is a matrix in which most of the elements are zero.

11	22	0	0	0	0	0
0	33	44	0	0	0	0
0	0	55	66	77	0	0
0	0	0	0	0	88	0
0	0	0	0	0	0	99

Fig 1. The above matrix contains only 9 nonzero elements, with 26 zero elements. Its sparsity is 74%, and its density is 26%. (Example of Sparse Matrix)
The number of zero-valued elements divided by the total number of elements (e.g., m × n for an m × n matrix) is called the sparsity of the matrix.

Proposing an efficient way and then extracting information from it to predict the link for developing a recommender system is going to be the topic of interest here.

Various algorithms have been proposed for factorizing a sparse matrix some of the known ones are:

Incidence Matrix Factorization [5]

In mathematics, an **incidence matrix** is a matrix that shows the relationship between two classes of objects. If

the first class is *X* and the second is *Y*, the matrix has one row for each element of *X* and one column for each element of *Y*. The entry in row *x* and column *y* is 1 if *x* and *y* are related (called *incident* in this context) and 0 if they are no

Adjacency Matrix Factorization

An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.

SVD Methods

The Singular Value Decomposition of an m × n real or complex matrix M is a factorization of the form U Σ V ∗ where U is an m × m real or complex unitary matrix, Σ is an m × n rectangular diagonal matrix with non-negative real numbers on the diagonal, and V is an n × n real or complex unitary matrix. If M is real, U and V T = V ∗ are real orthonormal matrices.

Different algorithm that can be applied in getting the result are:

a) Apriori Algorithm
b) K-Means
c) Apriori+K-Means
d) DB Scan
e) DBScan+ Apriori

II. DIFFERENT APPROACHES

We started from Apriori first and then tried combination of different algorithm which are:

1. *Apriori Algorithm*

The [3] Apriori algorithm is probably the most well-known algorithm in the area of frequent items discovery (Agrawal, Imielinski, & Swami, 1993). The algorithm takes advantage of the property that any subset of a frequent item set must be a frequent item set. If we have (N+1)-item set then we use the (N)-item set (N is number of items in the set) to discover it. Thus, the discovered frequent item sets of the first pass are used to generate the candidate sets of the second pass. Once the candidate 1-item sets are found their supports are counted to discover the frequent 2-itemsets by scanning the database. In the third pass, the frequent 2-itemsets are used to generate candidate 3-item sets. Termination condition, where there are no more new frequent item set, is found in Figure 2-2 . The algorithm contains two steps:

1. Join step: The first step is to join all frequent items of size k-1 ((k-1)-item set) with themselves to generate candidate K-item set. As a result the new list of k-item sets has been produced.
2. Prune step: This step come from the Apriori property which states if an item set is not frequent, then all its supersets are absolutely not a frequent set. Therefore we can prune all Candidate k- item sets by checking whether all its (k-l)-item sets subsets are frequent or not. If we find any member of (k-1)-item sets is not we can prune its superset from a new list.

Method: apriori_gen() [(Agrawal & Srikant, 1994)]
Input: set of all large (k-1)-item sets L_{k-1}
Output: A superset of the set of all large k-item sets
// Join step
Ii = Items i
insert into C_k
Select $p.I_1$, $p.I_2$, ……. , $p.I_{k-1}$, q .I_{k-1}
From L_{k-1} is p, L_{k-1} is q
Where $p.I_1 = q.I_1$ and …… and $p.I_{k-2} = q.I_{k-2}$
and $p.I_{k-1} < q.I_{k-1}$.
// Pruning step
For all item sets $c \in C_k$ do
For all (k-1)-subsets s of c do
If ($s \notin L_{k-1}$) then
delete c from C_k

Apriori Algorithm Pseudo code

Apriori Property

All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that

All subsets of a frequent itemset must be frequent (Apriori propertry).
If an itemset is infrequent, all its supersets will be infrequent.

2. *K-Means*

K-means[6] clustering is an algorithm to classify or to group a number of objects based on attributes/features into K number of group. K is a positive integer and the grouping is done by minimizing the sum of squares of

distances between data and the corresponding [4] cluster centroid (the center).
The above algorithm in pseudocode:

```
Initialize k means with random values

For a given number of iterations:
    Iterate through items:
        Find the mean closest to the
        item
        Assign item to mean
        Update mean
```

Reasons of time difference

Time complexity of Apriori is exponential 2^d
Where d is the total no of items

Time complexity of k-means is O(n * K * I * d)
n = number of points,
K = number of clusters,
I = number of iterations,
d = number of attributes

For large value of n K, I and d become constant and time complexity of K-Means shifted to Big O(n)

Applying K-Means and Apriori the time complexity shifted to Big $O(n)+2^k$
Where k is frequent itemset.

3. *DBSCAN* [7]

(Density-Based **S**patial Clustering of Applications with Noise) is a popular unsupervised learning method utilized in model building and machine learning algorithms. It is a clustering method that is used in machine learning to separate clusters of high density from clusters of low density. Given that DBSCAN is a density-based clustering algorithm, it does a great job of seeking areas in the data that have a high density of observations, versus areas of the data that are not very dense with observations. DBSCAN can sort data into clusters of varying shapes as well, another strong advantage.

The *DBSCAN algorithm* is based on this intuitive notion of "clusters" and "noise". The key idea is that for each point of a cluster, the neighbourhood of a given radius has to contain at least a minimum number of points.

Real life data may contain irregularities, like –

i) Clusters can be of arbitrary shape such as those shown in the figure below.

ii) Data may contain noise

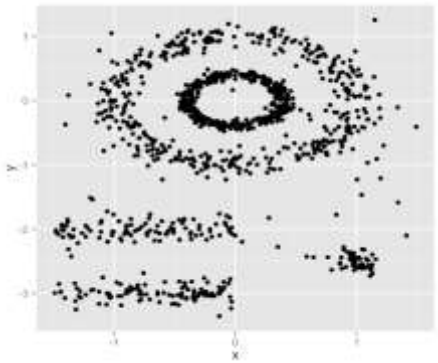


Fig 2. shows a data set containing nonconvex clusters and outliers/noises.

DBSCAN algorithm requires two parameters –

1. *eps* : It defines the neighbourhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered as neighbours. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and majority of the data points will be in the same clusters. One way to find the eps value is based on the k-distance graph.

2. *MinPts*: Minimum number of neighbours (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

In this algorithm, we have three types of data points.

a) *Core point*: A point is a core point if it has more than MinPts points within eps.

b) *Border Point*: A point which has fewer than MinPts within eps but it is in the neighbourhood of a core point.

c) *Noise*: A point which is neither a core point nor a border point.

DBSCAN works as such:

- Divides the dataset into *n* dimensions
- For each point in the dataset, DBSCAN forms an *n* dimensional shape around that data point, and then counts how many data points fall within that shape.

- DBSCAN counts this shape as a *cluster.* DBSCAN iteratively expands the cluster, by going through each individual point within the cluster, and counting the number of other data points nearby.

- Going through the aforementioned process step-by-step, DBSCAN will start by dividing the data into *n* dimensions. After DBSCAN has done so, it will start at a random point (in this case let's assume it was one of the red points), and it will count how many other points are nearby. DBSCAN will continue this process until no other data points are nearby, and then it will look to form a second cluster.

III. Experiments and Results

In the year 2016 a paper was published by Sho Yokoi and Hiroshi Kajino and Hisashi Kashima on Link Prediction by Incidence Matrix Factorization in which they proposed that IMF is more suitable for Sparse matrix factorization then any other methods available more specifically AMF. We have used this knowledge in our paper.

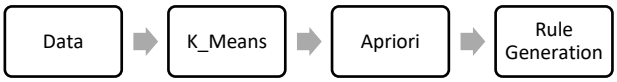
We applied apriori algorithm on our retail shop data set and found out that which item are related to one another and this result we want to use to predict link among data item, now the related items which we get are basically represented as 0 and 1 what it really mean is if item A and B are related to one another then we will have a 2-D matrix whose rows and column will represent two items and the value in the corresponding cell as 1 represent that they are related and if value in corresponding cell is 0 it means they are not related to one another. Since we can have thousands and thousands of elements so we can say that the matrix which will be formed will have elements as 0 and 1 so it will be a sparse matric as most of element will not be related to one another. Now performing any operation on this sparse matrix is a computationally very costly and finding a best way of matrix multiplication is a NP-Hard problem and similarly factorization of matrix is also a NP-Hard problem they are ways to factor a matrix which we have mentioned above and our task is to compare the combination of different algorithm to compute the time it takes to predict a link among different item set.

In this research we have applied K-Means with Apriori and then computed the time and compared it with prior one in which Apriori is applied solely and have found a difference in timing which are as follow

Apriori

Total Time Taken 7.48435

K-Means+Apriori



Total Time Taken 4.515625

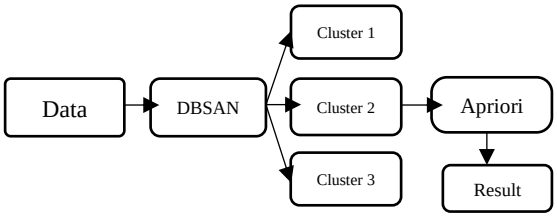
We applied DBSCAN algorithm on with Apriori algorithm and tried to analyse the the how time it is going to change and we found the following results:

DBSCAN on the retail data.

After applying DBSCAN we classified data into three clusters and stored it into excel file

We then applied apriori algorithm for predicting link among the items and we observed the time it took to predict the link, following is our finding

DBSCAN+Apriori



Total Time Taken 0.15625

Algorithm Name	Apriori	KMeans+ Apriori	DBSCAN+Apriori
Time Taken	*7.48435*	*4.515625*	*0.15625*

Comparison between different approaches

Algorithm Applied	Expected Accuracy
Apriori	70-75%
K-Means	50-55%
DB Scan	35-40%

Expected Accuracy Based on time of execution

This is a part of our test data which we used to predict the link among them

	Invoice	Stock Code	Unit Price	Lower	lower	Description	Invoice	CustomerID	Country
0	536365	0.169726	7.65	2	set 7 babushka nesting boxes	SET 7 BABUSHKA NESTING BOXES	2010-12-01 08:26:00	1.0000	United Kingdom
1	536365	0.156121	4.25	6	glass star frosted hight holder	GLASS STAR FROSTED HIGHT HOLDER	2010-12-01 08:26:00	1.0000	United Kingdom
2	536366	0.168142	1.85	6	hand warmer union jack	HAND WARMER UNION JACK	2010-12-01 08:26:00	1.0000	United Kingdom
3	536366	0.168129	1.85	6	hand warmer red polka dot	HAND WARMER RED POLKA DOT	2010-12-01 08:26:00	1.0000	United Kingdom
4	536367	0.169633	2.10	6	poppy’s playhouse bedroom	POPPY’S PLAYHOUSE BEDROOM	2010-12-01 08:26:00	1.0000	United Kingdom

From the above data after applying our approach following results were obtained

['BOX OF 6 ASSORTED COLOUR TEASPOONS'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['BOX OF 6 ASSORTED COLOUR TEASPOONS']
['BOX OF VINTAGE ALPHABET BLOCKS'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['BOX OF VINTAGE ALPHABET BLOCKS']
['BOX OF VINTAGE JIGSAW BLOCKS'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['BOX OF VINTAGE JIGSAW BLOCKS']
['DOORMAT NEW ENGLAND'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['DOORMAT NEW ENGLAND']
['FELTCRAFT PRINCESS CHARLOTTE DOLL'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['FELTCRAFT PRINCESS CHARLOTTE DOLL']
['HOME BUILDING BLOCK WORD'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['HOME BUILDING BLOCK WORD']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['IVORY KNITTED MUG COSY']
['IVORY KNITTED MUG COSY'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['LOVE BUILDING BLOCK WORD'] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ['LOVE BUILDING BLOCK WORD']
["POPPY'S PLAYHOUSE BEDROOM"] -> ['ASSORTED COLOUR BIRD ORNAMENT']
['ASSORTED COLOUR BIRD ORNAMENT'] -> ["POPPY'S PLAYHOUSE BEDROOM"]

IV. Conclusion

In this paper we tried to test combination of different algorithm and analysed how much time is going to be taken for different combination and we tried to justify the result we got, we also took the help of previous papers Sho Yokoi and Hiroshi Kajino and Hisashi Kashima on Link Prediction by Incidence Matrix Factorization.

The experiment result showed that our approaches produced a better overall performance then the previous approaches. Applying clustering techniques and the predicting link is more efficient and reduces time significantly, clustering algorithm which can cluster data more efficiently will have reduce the time accordingly. In our project we used inbuilt functions and have tested for data set in the range of 1000-10000, we will further try to tune our model and try more combination of algorithms to improve our model.

Use of Simulation software

Most of the algorithm used are open source and freely available, we used python more specifically Anaconda python as a programming language and platform as most of the algorithms are freely available and easy to implement in Anaconda python, we used Jupiter Notebook as an interface to test our proposed work, Model have been tested on system having operating system as Ubantu 19.10(Linux), 8 GB RAM, processor used Intel Core i5 7th gen, GPU Nvidia Geforce 940MX.

V. References

1. (Hand, 2006)Hand, D. J. (2006). Data Mining. *Encyclopedia of Environmetrics*, *2*.
2. (Liu, 1998) Liu, Bing, Wyn ne Hsu, and Yiming Ma. "Integrating classification and association rule mining." *KDD*. Vol. 98. 1998.
3. (Al-Maolegi, 2014)Yılmaz, Nergis, and Gülfem Işıklar Alptekin. "The Effect of Clustering in the Apriori Data Mining Algorithm: A Case Study." *Proceedings of the World Congress on Engineering*. Vol. 3. 2013.
4. (Yokoi, 2017)Yokoi, Sho, Hiroshi Kajino, and Hisashi Kashima. "Link Prediction in Sparse Networks by Incidence Matrix Factorization." *Journal of Information Processing* 25 (2017): 477-485.
5. (Pakhira, 2015)M. K. Pakhira, "A Linear Time-Complexity k-Means Algorithm Using Cluster Shifting," 2014 International Conference on Computational Intelligence and Communication Networks, Bhopal, 2014, pp. 1047-1051, doi: 10.1109/CICN.2014.220.
6. DBSCAN Clustering in ML | Density based clustering.