Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics, Istanbul, Turkey, May 27-29, 2006 (pp281-286)

Comparing Clustering Techniques for Telecom Churn Management

ADEM KARAHOCA, ALI KARA
Bahcesehir University, Engineering Faculty
Computer Engineering Department
34538 Bahcesehir – Istanbul, TURKEY

Abstract: - Mobile telecommunication sector has been accelerated with GSM 1800 licenses in the Turkey. Since then, churn management has won vital importance for the GSM operators. Customers should have segmented according to their profitability for the churn management. If we know the profitable customer segments, we have chance to keep in hand the most important customers via the suitable promotions and campaigns. In this study, we implemented clustering algorithms to 250 subscribers' 100MBs call detail records(CDRs), demographic data and billing information. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) compared with K-Means, Expectation Maximization and Farthest-First clustering techniques. As a result, DBSCAN has good distinct clusters for profiling customer segments.

Key-Words: - Clustering Algorithms, Data Mining, Clustering for GSM, DBSCAN, Benchmarking of Clustering Methods, Churn Management

1 Introduction

All of the GSM companies would like to know what would happen to their loyal subscribers. It is surely impossible accurately to know that, however it could be estimated if you have valuable recent data and a useful miner for them.

Cellular phone usage is very important for the GSM operators. Both voice and data services are basic revenue gaining items for them. Thus, GSM operators should have to predict their customer behaviors to keep them registered. Managers want to understand why some customers remain loyal while others leave from the telecom company. By this viewpoint, they can be constructed a model that derived from historical data of loyal and left customers. The model illustrates the customer loyalty and churn rate.

Previous experiences show us that most effective churn management can be obtained with following independent parameters [1]: occupation of the subscriber, monthly expense, monthly income, credit limit, average call length for a month, average call length for 3 months, average call length for 4 to 6 months, average SMS amount for a month, average SMS amount for 3 months, average SMS amount for 4 to 6 month.

In order to make successful churn management, our main object is to investigate density-based clustering algorithms as a data miner and evaluate its results. There are many clustering methods known in the data mining literature. We preferred Density-Based Spatial Clustering of Application with Noise (DBSCAN) method to develop and evaluate for call detail records (CDRs) of GSM subscribers. Furthermore we have observed what is happening when a new member has joined to the set of members clustered. Noise tolerance was one of the most wondered parts of clustering for GSM datasets, so we have developed a method for this purpose and accelerates it so that it is satisfying.
Fundamental objectives of this paper are:

- Demonstrating efficiency of DBSCAN in the GSM datasets.
- Observing results of the event when a new member joins to the set of members.
- To decide which clustering structure is better for noise percent known, as Noise Tolerance.
- Comparison of DBSCAN with K-Means, Expectation Maximization and Farthest-First as another clustering method.

We have demonstrated the subject detailed in the 2nd Section. The main problems that we encountered during development have been exhibited in the Section 3. The solutions have followed the problems in the 4th Section. In section 5, DBSCAN has been compared with K-Means, Expectation Maximization and Farthest-First clustering techniques. You can find conclusions of this work in the Section 6.

2 DBSCAN Implementation for CDRs

Density-Based Spatial Clustering of Application with Noise (DBSCAN) method evaluated as Density-Based method in this paper. In order to cluster GSM data, we preprocessed the CDRs, demographic data and billing information about the GSM subscribers and we produced scores between 0-250 for each subscriber. As discussed in [1], classification techniques work well for churn management, by general classification methods; it's quite easy to separate the groups, if we can define the general qualifications of the groups. In addition to this, we must be sure that we have only 3 separated groups. For this reason we focused on DBSCAN algorithm that can be used for very large databases.

2.1 The Algorithm DBSCAN

Traditional DBSCAN algorithm has two variables. Meanwhile those are X and Y coordinates of the DBSCAN surface. We have initially mentioned traditional algorithm of DBSCAN, but first of all we have to know 2 main parameters of the algorithm.

The key idea of density-based clustering is that for each object of a cluster the neighborhood of a given radius (*Eps*) has to contain at least a minimum number of objects (*MinPts*), i.e. the cardinality of the neighborhood has to exceed some threshold [2].

If two clusters C1 and C2 are very close to each other, it might happen that some point p belongs to both, C1 and C2. Then p must be a border point in both clusters because otherwise C1 would be equal to C2 since we use global parameters. In this case, point p will be assigned to the cluster discovered first [3].

After exhibiting traditional DBSCAN, we can start to mention about DBSCAN for GSM dataset. As we said before, traditional DBSCAN has 2 coordinates on the set of point surface. However in GSM dataset, we have 1 coordinate and 1 index. Remarking 1 index, we have pointed to GSM operator subscriber number (MSISDN); and 1 coordinate means the output score of independent variables that we evaluated. We have examined the problems and modifications in the section 3, while implementing GSM data onto traditional DBSCAN algorithm.

2.2 Estimation over DBSCAN for GSM

So far, we have created clusters in set of points according to Eps and MinPts. Then when a new member joins to the set, we could estimate which cluster would contain the member point. If we know level of noise members, and need to know how many alternations of clusters could happen, we can use the second estimation part of DBSCAN for GSM dataset. We have called this procedure as "Noise Tolerance".

2.2.1 New Member Addition into GSM Data Set

After a new subscriber joins to GSM operator, we have to execute DBSCAN algorithm again. But, we can not be sure that all clusters will remain same. Adding new subscriber to our clusters and cluster levels may be changed. The entrance and re-execute algorithm is like below:

```
SetOfPoints.AddMember(MSISDN, OutputScore)
SetOfPoints.SetClusterIDs(UNCLASSIFIED)
//re-execute of DBSCAN
DBSCAN (SetOfPoints, Eps, MinPts)
  // SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point, ClusterId,
       Eps, MinPts) THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

2.2.2 Noise Tolerance

The second part DBSCAN estimation is to calculate how many alternative clusters we might have, when we enter the noise level of the GSM subscribers.

For this purpose, we defined a few new variables, such as NoisePercent, MinOutput, MaxOutput etc. The algorithm of Noise Tolerance can simply be explained as;

1. We do not know Eps and MinPts, and would like to know how many alternations we might have?
2. We have a set of GSM subscribers and a noise percent. For example, noise percent is equal to 10% means that 10% of subscribers constitutes noise cluster (means 10% could not be joined to any cluster).
3. Under these conditions what is our Eps and MinPts? And how many alternatives do we have to form clusters?

```
FOR I FROM 1 TO (MaxOutput-MinOutput) DO
  Eps = 1
  J = 1
  noiseSize = 0
  WHILE((J<SetOfPoints.size) AND (noiseSize <=
```

```
                                        noisePercent))
    MinPts = J
    DBSCAN() //Traditional DBSCAN above
    noiseSize = 0
    FOR K FROM 1 TO (SetOfPoints.size) DO
      IF(SetOfPoints.Member[K].ClusterID=0) THEN
        noiseSize = noiseSize + 1
      END IF
    END FOR
    ClusterID = ClusterID – 1
    IF((noiseSize = noisePercent) AND (ClusterID >=
     NextID(MinimumWished)
     Save(noiseSize, Eps, MinPts, ClusterID)
    END IF
    J = J + 1
  END WHILE
END FOR
```

Via this algorithm, we could prefer minimum cluster number. For example, if subscribers clustered into less than 3 parts are not valid for our application, we can prefer MinimumWished as 3.

3 Usage of DBSCAN in Churn Management

We applied DBSCAN clustering algorithm to define different clusters of GSM subscribers. We have been able to decide the campaigns matching the clusters of subscribers.

When we chose Eps=17 and MinPts=3, we have reached the results below:

Table 1: Clusters with known Eps & MinPts

```
NOISE Values: There is no NOISE!
Cluster( 1):
  MSISDN=   2600000   Output=   246
  MSISDN=   2600010   Output=   202
  MSISDN=   2600025   Output=   224
  MSISDN=   2600030   Output=   213
  MSISDN=   2600035   Output=   245
  MSISDN=   2600045   Output=   211
  MSISDN=   2600085   Output=   199
  MSISDN=   2600105   Output=   240
Cluster( 2):
  MSISDN=   2600015   Output=    16
  MSISDN=   2600060   Output=     7
  MSISDN=   2600070   Output=    32
  MSISDN=   2600080   Output=    43
  MSISDN=   2600095   Output=    15
  MSISDN=   2600100   Output=    11
  MSISDN=   2600110   Output=    50
  MSISDN=   2600120   Output=    29
  MSISDN=   2600125   Output=    44
Cluster( 3):
  MSISDN=   2600020   Output=   178
  MSISDN=   2600040   Output=   161
  MSISDN=   2600050   Output=   153
  MSISDN=   2600130   Output=   157
  MSISDN=   2600135   Output=   162
Cluster( 4):
  MSISDN=   2600005   Output=    70
  MSISDN=   2600055   Output=   119
  MSISDN=   2600065   Output=    98
  MSISDN=   2600075   Output=    97
  MSISDN=   2600090   Output=   122
  MSISDN=   2600115   Output=    81
  MSISDN=   2600140   Output=   105
  MSISDN=   2600145   Output=    88
```

We initially tested 20 subscribers' related data, and want to start campaigns for them. But we don't know which subscribers will join to which campaign. We thought and planned about everything in our strategy, according to the fact that we have only 3 clusters for 3 kinds of subscribers (loyal, hopper, lost). After with DBSCAN, we have seen that we might have more than 3 clusters, and there is no noise value. In real world, we cannot cluster some of our subscribers. Thus the noise subscribers are belonging to noise cluster. If this is a problem for our GSM Company, the problem can be solved as below.

For another example, we started campaigns, after clustering the subscribers. During the campaigns, we have started to think about members. The part of DBSCAN algorithm to add new member, which was defined in the section 2, has helped us and the new subscribers have participated in their campaign clusters. This action has formed the first part of DBSCAN estimation task.

Working on the second part of DBSCAN estimation task, we have considered what if we do not know what are our Eps and MinPts. In the 2nd section, we have mentioned about this problem and given the solution of it in pseudo code. The example below is showing the results.

If we chose that we don't want any noise cluster (means noise percent is equal to 0%), and we would like to observe how many alternations we might have for number of clusters, Eps and MinPts. We could run the algorithm for various noise percents. We ran DBSCAN for the sample above and determined the MinimumWished is 3. MinimumWished, as we mentioned about it in the section 2, means the minimum number of clusters in a set of subscribers. The results are in the Table 2.

Table 2: Results of noise tolerance

NOISE	Eps	MinPts	Cluster No
0	17	3	4
0	18	1	4
0	18	2	4
0	18	3	4
0	19	1	4

0	19	2	4
0	19	3	4
0	20	1	3
0	20	2	3
0	20	3	3
0	22	4	3
0	23	4	3
0	24	4	3
0	25	4	3
0	25	5	3
0	26	5	3
0	27	6	3
0	28	6	3
0	29	6	3

Processing Time: 00:00:39

The results mean that when we accept Eps=24 and MinPts=4, we achieve the zero noise with 3 clusters. And we can check whether we can reach 5 clusters with zero noise.

When we ran DBSCAN for our sample, we have seen that zero noise with 5 clusters is impossible. Furthermore, we have paid attention on the processing time. Even for a tiny sample, it is not acceptable. We have measured it on the Pentium-4 1.8 GHz CPU and 1 Gb memory. To shorten the long processing time, we have offered an improvement in the section 4.

4 Solutions for the DBSCAN Run-Time Productivity Problem in GSM

We ran the algorithm DBSCAN on various samples of data sets; in any run-time we couldn't be satisfied with run-time length. Changing samples couldn't change the results, which was too long. Then we tried debugging the code to find bottleneck. Solutions on 2 points have helped us to improve Noise Tolerance procedure of DBSCAN algorithm for GSM data set.

4.1 Omitting the Idle Process

According to Noise Tolerance algorithm, we added all possible Eps and MinPts values into DBSCAN loop, and then we saved the valid values that are accessible.

If we remember how the algorithm works, Eps and MinPts values are increasing in sequence. For a MinPts value, if an Eps cannot satisfy the valid result, increment of MinPts is an idle process, which steals from run-time.

With a basic control, we have broken the loop, and omitted the idle processes. When we pay attention to the code of Noise Tolerance, we could see that we haven't used for loop, but while loop.

Since the main problem of run-time length was noise tolerance; we focused on the code of it. We remembered that we asked the "MinimumWished" number of clusters from the user, and checked the Eps and MinPts, which satisfy the value asked.

On this point, we considered whether it is accelerating the program, if we postpone the control. We have gained a giant improvement on run-time speed. We have surely used the control somewhere in the program, but not in the main (and the longest) loop. By testing system after changing code, we have seen we might keep memory busy more than before, but the CPU has calmed down. Using this variation, we have run the algorithm again and reach the 10 seconds as processing time on the same configuration, 40 seconds instead for the test subscribers' data set.

5 DBSCAN vs. K-Means, Expectation Maximization and Farthest-First Clustering Algorithms

In this section, we have evaluated the performance of DBSCAN, comparing it with the performance of some other clustering algorithms. We have selected K-Means (KM), Expectation Maximization (EM) and Farthest-First (FF) methods as rival for DBSCAN.

We compared the Algorithm DBSCAN with the others. We have already developed a tool by C++ for DBSCAN; we have used it for benchmark process. However for the other methods such as KM, EM and FF we have used WEKA, which is the popular open-source data mining tool of University of Waikato.

Three kinds of subscribers are considered for churn management, as we mentioned in the Section 3. That's why we have run all algorithms for 3 clusters, and observed the results regarding to these clusters, which are loyal, hopper and lost. We have implemented the algorithms for this purpose. In all experiments, we have accepted the noise percentage ("noise tolerance" as we called) is equal to 0%.

All experiments have been run on the workstation that is configured with Intel Pentium 1.7GHz CPU, 1GB memory.

For the benchmark we have not used the data above, which has 30 instances, but we have used the input that has 250 instances. About K-Means by WEKA, we have observed the results below:

Fig. 1. K-Means results for 3 clusters

```
Cluster centroids:

Cluster 0
        Mean/Mode:  2600569.2268      52.8969
        Std Devs:       310.3219      32.5843
Cluster 1
        Mean/Mode:  2600981.0759     176.519
        Std Devs:       168.2464      46.9536
Cluster 2
        Mean/Mode:  2600296.7123     180.0685
        Std Devs:       187.4708      42.4671

Clustered Instances

0       97 ( 39%)
1       79 ( 32%)
2       73 ( 29%)
```

We have initially paid attention that mean value of output attributes. For the cluster 0, mean value is 52.8969, which is acceptable. But for the second and third cluster they are 176.519 and 180.0685, and then they are too close to each other, which mean they could contain some values of each other. We should look out the standard deviations of these values. If the standard deviations are less than the difference between the mean values, prejudice might have misled us. However we have seen that standard deviations are too high, which means that the clusters aren't separated from each other. The first question in the minds is how we can start a campaign for these clusters, which are too close to each other, unless churn management can clearly distinguish any subscriber than the others. Thus, we can conclude that K-means clustering algorithm did not work well for our data sets.

Fig. 3. EM results for 3 clusters

```
Attribute: MSISDN
Normal Distribution. Mean = 2600568.7097 StdDev = 324.0399
Attribute: output
Normal Distribution. Mean = 39.3265 StdDev = 23.6204

Cluster: 1 Prior probability: 0.2445

Attribute: MSISDN
Normal Distribution. Mean = 2601048.9213 StdDev = 125.0499
Attribute: output
Normal Distribution. Mean = 160.2842 StdDev = 55.6823

Cluster: 2 Prior probability: 0.4871

Attribute: MSISDN
Normal Distribution. Mean = 2600432.9918 StdDev = 265.9064
Attribute: output
Normal Distribution. Mean = 163.5414 StdDev = 54.9707
Clustered Instances

0       75 ( 30%)
1       60 ( 24%)
2      114 ( 46%)
```

We ran the Expectation Maximization and Farthest First algorithms to reach valid results for churn management respectively.

Fig. 2. Farthest-First results for 3 clusters

```
Cluster centroids:

Cluster 0
          2600795.0 34.0
Cluster 1
          2600000.0 246.0
Cluster 2
          2601165.0 250.0

Clustered Instances

0       129 ( 52%)
1        61 ( 24%)
2        59 ( 24%)
```

But when we saw that characteristics of the results are similar to K-Means, we have considered that we wouldn't be able to reach the results those Churn Management needs, by the clustering methods such as partitioning etc. Churn management for telecom companies naturally needs to determine certain borders of clusters to start a campaign that depends on their characteristics.

To cluster the set of subscribers according to our purpose, we need to separate them as their dense, because density of subscribers' outputs could determine the characteristic of the campaign that would be applied. Just this reason could indicate that DBSCAN algorithm (as a density based clustering algorithm) is better than the others.

In order to indicate our suggestion, we have executed DBSCAN tool that we developed on telecom data that we had, and then have examined the output of DBSCAN for the telecom data.

Fig. 4. DBSCAN results for 3 clusters

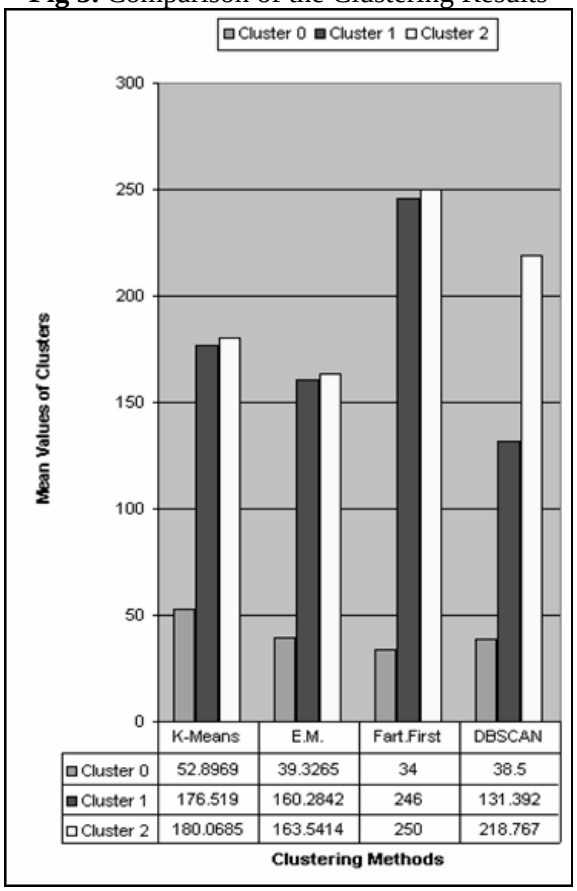
```
Cluster centroids:

Cluster 0 Prior probability: 0.2932
      Mean of Output: 218.767
      Std Devs: 18.3434
Cluster 1 Prior probability: 0.2972
      Mean of Output: 38.5
      Std Devs: 21.1923
Cluster 2 Prior probability: 0.4096
      Mean of Output: 131.392
      Std Devs: 28.2836

Clustered Instances

0       73 ( 29%)
1       74 ( 30%)
2      102 ( 41%)
```

After we ran the DBSCAN tool on the telecom data, we have experienced that our idea is right. There are 3 clusters and 3 of them are clearly distinguished from each other in the latest results. If

we pay attention to mean values of the clusters, we can see that they are far from the others, regarding to the standard deviations.

If we wanted to see all results in a graphic, Fig. 5. would help us. Mean values of all clusters by any methods have been illustrated in the graphic. Then we can see how clusters are close to each others in any clustering method.

Fig 5. Comparison of the Clustering Results



Via the Fig. 5., we can observe that mean values of Cluster-1 and Cluster-2 are quite near to each other, which means that the subscribers' characteristics are so near to each other. However there isn't any problem like that about in DBSCAN, because it's obvious to see that all clusters are far from the others.

We cannot surely say that the results of K-Means, Expectation Maximization and Farthest-First are useless, but we may encounter problems while clustering to telecom churn management by these methods.

Consequently we have experienced that there is a right way towards clustering to telecom churn management by DBSCAN, as a density based algorithm, due to the fact that the algorithm DBSCAN includes clusters according to their density. Only DBSCAN supplies the main requirement of telecom churn management with it is "clear separation".

6 Conclusions

DBSCAN, as a clustering algorithm, has been implemented to the GSM sector, have run to help Churn Management and the results have been demonstrated in this paper.

The algorithm DBSCAN has been defined detailed in order to remove question marks in the minds during implementation time of the algorithm to the GSM sector. While defining DBSCAN some problems have been come across. Solving the problems, we have explored how plain the code of DBSCAN is. It's obvious to see that DBSCAN works faster than its equivalents, by means of this plainness.

The Density-based algorithm DBSCAN has been compared with some other clustering methods and after performance test, DBSCAN has seemed more suitable than K-Means, Expectation Maximization and Farthest-First for GSM operators to churn management.

References:

[1] Karahoca, A., Data Mining Via Cellular Neural Networks In The GSM Sector, The 8th IASTED *International Conference Software Engineering And Applications*, Cambridge, MA, 2004, pp. 19-24.

[2] Ester M., Kriegel H.-P., Sander J., and Xu X. 1996. Incremental Clustering for Mining in a Data Warehousing Environment, 4*th International Conference on Knowledge Discovery and Data Mining (KDD-98)*, 1998.

[3] Beckmann N., Kriegel H.-P., Schneider R, and Seeger B. 1990. The R*-tree: An Efficient and obust Access Method for Points and Rectangles, Proc. ACM SIGMOD *Int. Conf. on Management of Data*, Atlantic City, NJ, 1990, pp. 322-331.

[4] Stonebraker M., Frew J., Gardels K., and Meredith J.1993. The SEQUOIA 2000 Storage Benchmark, Proc. ACM SIGMOD *Int. Conf. on Management of Data*, Washington, DC, 1993, pp. 2-11.

[5] Ester M., Kriegel H.-P., Sander J., and Xu X. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.