Integrating Fuzzy *c*-Means Clustering with PostgreSQL*

R. M. Miniakhmetov
tavein@gmail.com
South Ural State University

Abstract. Many data sets to be clustered are stored in relational databases. Having a clusterization algorithm implemented in SQL provides easier clusterization inside a relational DBMS than outside with some alternative tools. In this paper we propose Fuzzy *c*-Means clustering algorithm adapted for PostgreSQL open-source relational DBMS.

Keywords: fuzzy c-means; fcm; fuzzy clustering; postgresql; integrating clustering; relational dbms.

1. Introduction

Integrating clustering algorithms is a topic issue for database programmers [1]. Such an approach, on the one hand, encapsulates DBMS internal details from application programmer. On the other hand, it allows to avoid overhead connected with export data outside a relational DBMS. The *Fuzzy c-Means (FCM)* [2], [3], [4] clustering algorithm provides a fuzzy clustering of data. Currently this algorithm have many implementations on a high-level programming languages [5], [6]. For implementation the FCM algorithm in SQL we choose an open-source PostgreSQL DBMS [7].

The paper is organized as follows. Section 2 introduces basic definitions and an overview of the FCM algorithm. Section 3 proposes implementation of the FCM in

* This paper is supported by the Russian Foundation for Basic Research (grant No. 09-07-00241-a).

SQL called pgFCM. Section 0 briefly discusses related work. Section 0 contains conclusion remarks and directions for future work.

2. The Fuzzy c-Means Algorithm

The K-Means [8] is one of the most popular clustering algorithms, it is a simple and fairly fast [9]. The FCM algorithm generalizes K-Means to provide fuzzy clustering, where data vectors can belong to several partitions (*clusters*) at the same time with a given weight (*membership degree*). To describe FCM we use the following notation:

- $d \in \mathbb{N}$ – dimensionality of a data vectors (or data items) space;
- $l \in \mathbb{N} : 1 \le l \le d$ – subscript of the vector's coordinate;
- $n \in \mathbb{N}$ – cardinal number of training set;
- $X \subset \mathbb{R}^d$ – training set for data vectors;
- $i \in \mathbb{N} : 1 \le i \le n$ – vector subscript in a training set;
- $x_i \in X$ – the i-th vector in the sample;
- $k \in \mathbb{N}$ – number of clusters;
- $j \in \mathbb{N} : 1 \le j \le k$ – cluster number;
- $C \subset \mathbb{R}^{k \times d}$ – matrix with clusters' centers (*centroids*);
- $c_j \in \mathbb{R}^d$ – center of cluster j, d-dimensional vector;
- $x_{il}, c_{jl} \in \mathbb{R}$ – l-s coordinates of vectors x_i and c_j respectively;
- $U \subset \mathbb{R}^{n \times k}$ – matrix with membership degrees, where $u_{ij} \in \mathbb{R}$: $0 \le u_{ij} \le 1$ is a membership degree between vector x_i and cluster j;
- $\rho(x_i, c_j)$ – distance function, defines a membership degree between vector x_i and cluster j;
- $m \in \mathbb{R} : m > 1$ – the fuzzyfication degree of objective function;

- J_{FCM} – objective function.

The FCM is based on minimization of the *objective function* J_{FCM}:

$$J_{FCM}(X,k,m) = \sum_{i=1}^{N}\sum_{j=1}^{k} u_{ij}^{m}\rho^{2}(x_i,c_j) \qquad (1)$$

Fuzzy clusterization is carried out through an iterative optimization of the objective function (1). Membership matrix U and centroids c_{ij} are updated using the following formulas:

$$u_{ij} = \sum_{t=1}^{k}\left(\frac{\rho(x_i,c_j)}{\rho(x_i,c_t)}\right)^{\frac{2}{1-m}} \qquad (2)$$

$$\forall j,l \quad c_{jl} = \frac{\sum_{i=1}^{n} u_{ij}^{m}\cdot x_{il}}{\sum_{i=1}^{n} u_{ij}^{m}} \qquad (3)$$

Let s is a number of iteration, $u_{ij}^{(s)}$ and $u_{ij}^{(s+1)}$ are elements of matrix U on steps s and $s+1$ respectively, and $\varepsilon \in (0,1) \subset \mathbb{R}$ is a termination criterion. Then the termination condition can be written as follows:

$$\max_{ij}\{|u_{ij}^{(s+1)} - u_{ij}^{(s)}|\} < \varepsilon \qquad (4)$$

Objective function (1) converges to a local minimum (or a saddle point) [10].

Input: X, k, m, ε

Ouput: U

1. $s := 0, U^{(0)} := (u_{ij})$ {initialization}
2. **repeat**
   {computation of new centroids' coordinates}
3. Compute $C^{(s)} := (c_j)$ using formula (3) where $u_{ij} \in U^{(s)}$
   {update matrixes values}
4. Compute $U^{(s)}$ and $U^{(s+1)}$ using formula (2)
5. $s := s+1$
6. **until** $\max_{ij}\{|u_{ij}^{(s)} - u_{ij}^{(s-1)}|\} \le \varepsilon$

Fig. 1. The Fuzzy *c*-Means Algorithm.

Algorithm showed on Fig. 1 presents the basic FCM. The input of algorithm receives a set of data vectors $X = (x_1, x_2, \ldots, x_n)$, number of clusters *k*, fuzzyfication degree *m*, and termination criterion ε. The output is a matrix of membership degrees *U*.

3. Implementation of Fuzzy c-Means Algorithm using PostgreSQL

In this section we suggest pgFCM algorithm as a way to integrate FCM algorithm with PostgreSQL DBMS.

3.1 General Definitions

To integrate FCM algorithm with a relational DBMS it is necessary to perform matrixes U and X as relational tables. Subscripts for identification elements of

relational tables are presented in Table 1 (numbers n, k, d are defined above in a section 2).

Table 1. Data Elements Subscripts.

Subscript	Range	Semantics
i	$\overline{1,n}$	vector subscript
j	$\overline{1,k}$	cluster subscript
l	$\overline{1,d}$	vector's coordinate subscript

As a function of distance $\rho(x_i, c_j)$, without loss of generality, we use the Euclidean metric:

$$\rho(x_i, c_j) = \sqrt{\sum_{l=1}^{d} (x_{il} - c_{jl})^2} \qquad (5)$$

To compute the termination criterion (4) we introduce the function δ as follows:

$$\delta = \max_{ij} \{ | u_{ij}^{(s+1)} - u_{ij}^{(s)} | \} \qquad (6)$$

3.2 Database Scheme

Table 2 summarizes database scheme of pgFCM algorithm (underlined columns are primary keys).

Table 2. Relational Tables of pgFCM Algorithm

No.	Table	Semantics	Columns	Number of rows
1	SH	training set for data vectors (horizontal form)	$\underline{i}, x1, x2, \ldots, xd$	n
2	SV	training set for data vectors (vertical form)	$\underline{i,l}, val$	$n \cdot d$
3	C	centroids' coordinates	$\underline{j,l}, val$	$k \cdot d$
4	SD	distances between x_i and c_j	$\underline{i,j}, dist$	$n \cdot k$
5	U	degree of membership vector x_i to a cluster c_j on step s	$\underline{i,j}, val$	$n \cdot k$
6	UT	degree of membership vector x_i to a cluster c_j on step $s+1$	$\underline{i,j}, val$	$n \cdot k$
7	P	result of computation function δ (6) on step s	$\underline{d,k,n,s}, delta$	s

In order to store sample of a data vectors from set X it is necessary to define table $SH(\underline{i}, x1, x2, \ldots, xd)$. Each row of sample stores vector of data with dimension d and subscript i. Table SH has n rows and column i as a primary key. FCM steps demand aggregation of vector coordinates (sum, maximum, etc.) from set X. However, because of its definition, table SH does not allow using SQL aggregation functions. To avoid this obstacle we define a table $SV(\underline{i,l}, val)$, which contains $n \cdot d$ rows and have a composite primary key (i,l). Table SV represents a data sample from table SH ans supports SQL aggregation functions max and sum. Due to store coordinates of cluster centroids temporary table $C(\underline{j,l}, val)$ is defined. Table C has $k \cdot d$ rows and the composite primary key (j,l). Like the table SV, structure of table C allows to use aggregation functions. In order to store distances $\rho(x_i, c_j)$ table $SD(\underline{i,j}, dist)$ is used. This table has $n \cdot k$ rows and the composite primary key (i, j). Table $U(\underline{i,j}, val)$ stores membership degrees, calculated on *s*-th step. To store membership degrees on $s+1$ step similar table $UT(\underline{i,j}, val)$ is used. Both tables have $n \cdot k$ rows and the composite primary key (i, j). Finally, table $P(\underline{d,k,n,s}, delta)$ stores iteration number s and the result of computation function (6) *delta* for this iteration number. Number of rows in table P depends on the number of iterations.

3.3 The pgFCM Algorithm

The pgFCM algorithm is implemented by means of a stored function in *PL/pgSQL* language. Fig. 2 shows the main steps of the pgFCM algorithm.

Input: SH, k, m, eps

Ouput: U

{initialization}

1. Create and initialize temporary tables (U, P, SV, etc.)
2. **repeat**

{computations}

3. Compute centroids coordinates. Update table *C*.
4. Compute distances $\rho(x_i, c_j)$. Update table *SD*.
5. Compute membership degrees $UT = (ut_{ij})$.

{update}

6. Update tables *P* and *U*.

{check for termination}

7. **until** $P.delta \geq eps$

Fig. 2. The pgFCM Algorithm.

The input set of data vectors X stored in table SH. Fuzzyfication degree m, termination criterion ε, and number of clusters k are function parameters. The table U contains a result of pgFCM work.

3.4 Initialization

Initialization of tables SV, U, and P provided by SQL-code `I1`, `I2`, and `I3` respectively. Table SV is formed by sampling records from the table SH.

```
I1: INSERT INTO SV
      SELECT SH.i, 1, x1
      FROM SH;
    ...
    INSERT INTO SV
      SELECT SH.i, d, xd
      FROM SH;
```


For table U a membership degree between data vector x_i and cluster j takes 1 for all $i = j$.

```
I2: INSERT INTO U (i, j, val)
      VALUES (1, 1, 0);
    ...
    INSERT INTO U (i, j, val)
      VALUES (j, j, 1);
    ...
    INSERT INTO U (i, j, val)
      VALUES (n, k, 0);
```

In other words, as a start coordinates of centroids, first d data vectors from sample X are used.

$$\forall\, i = j \quad u_{ij} = 1 \Rightarrow c_j = x_i$$

When initializing the table P, the number of points k is taken as a parameter of the function $pgFCM$. A data vectors space dimensionality d and a cardinal number of the training set n also provided by function $pgFCM$ parameters. The iteration number *s* and *delta* initializes as zeros.

```
I3: INSERT INTO P(d, k, n, s, delta)
      VALUES (d, k, n, 0, 0.0);
```

3.5 Computations

According to Fig. 2, the computation stage is splitted to the following three sub-steps: computation coordinates of centroids, computation of distances, and computation membership degrees, marked as `C1`, `C2`, and `C3` respectively.

```
C1: INSERT INTO C
      SELECT R1.j, R1.l, R1.s1 / R2.s2 AS val
      FROM (SELECT j, l, sum(U.val^m * SV.val) AS s1
            FROM U, SV
            WHERE U.i = SV.i
            GROUP BY j, l) AS R1,
           (SELECT j, sum(U.val^m) AS s2
            FROM U
            GROUP BY j) AS R2
      WHERE R1.j = R2.j;
C2: INSERT INTO SD
      SELECT i, j, sqrt(sum((SV.val - C.val)^2))) AS
dist
      FROM SV, C
      WHERE SV.l = C.l
      GROUP BY i, j;
```

Through the FCM, computations of the distances provide by formula (2). In formula (3) the fraction's numerator does not depend on t, then we can rewrite this formula as follows:

$$u_{ij} = \rho^{\frac{2}{1-m}}(x_i, c_j) \cdot \left(\sum_{t=1}^{k} \rho^{\frac{2}{m-1}}(x_i, c_t) \right)^{-1} \qquad (7)$$

Thus, the computation of membership degrees can be written as follows:

```
C3: INSERT INTO UT
```



```
      SELECT i, j, SD.dist^(2.0^(1.0-m)) * SD1.den AS
val
      FROM (SELECT i, 1.0 / sum(dist^(2.0^(m-1.0))) AS
den
            FROM SD
            GROUP BY i) AS SD1, SD
      WHERE SD.i = SD1.i;
```

3.6 Update

Update stage of the pgFCM modifies P and U tables as shown below in `U1` and `U2` respectively.

```
U1: INSERT INTO P
      SELECT L.d, L.k, L.n, L.s + 1 AS s, E.delta
      FROM (SELECT i, j, max(abs(UT.`val - U.val)) AS
delta
            FROM U, UT
            GROUP BY i, j) AS E,
           (SELECT d, k, n, max(s) AS s
            FROM P
            GROUP BY d, k, n) AS L) AS R
```

Table UT stores temporary membership degrees to be inserted into table U. To provide the rapid removal all the table U rows, obtained at the previous iteration, we use the truncate operator.

```
U2: TRUNCATE U;
    INSERT INTO U SELECT * FROM UT;
```

3.7 Check

This stage is the final for the algorithm pgFCM. On each iteration the termination condition (4) must be checked.

To implement the check, the result *delta* of the function (6) from table P is stored in the temporary variable *tmp*.

```
CH1: SELECT delta INTO tmp
     FROM P,(SELECT d, k, n, max(s) AS max_s
             FROM P
             GROUP BY d, k, n) AS L
     WHERE P.s = L.max_s AND P.d = L.d AND P.k = L.k AND
P.n =L.n;
```

After selecting the *delta*, we need to check the condition $\delta < \varepsilon$. Then if this condition is true we should stop, otherwise, work will be continued.

```
CH2: IF (tmp < eps) THEN
       RETURN;
     END IF;
```

The final result of the algorithm pgFCM will be stored in table U.

4. Related Work

Research on integrating data mining algorithms with relational DBMS includes the following. Association rules mining is explored in [11]. General data mining primitives are suggested in [12]. Primitives for decision trees mining are proposed in [13].

Our research was inspired by papers [1], [14], where integrating K-Means clustering algorithm with relational DBMS, was carried out. The way we exploit is similar to mentioned above. The main contribution of the paper is an extension of results presented in [1], [14] for the case where data vectors may belong to several clusters. Such a case is very important in problems connected with medicine data analysis [15], [16]. To the best of our knowledge there are no papers devoted to implementing fuzzy clustering with relational DBMS.

5. Conclusion

In this paper we have proposed the pgFCM algorithm. pgFCM implements Fuzzy *c*-Means clustering algorithm and processes data stored in relational tables using PostgreSQL open-source DBMS. There are following issues to continue our research. Firstly, we plan to investigate pgFCM scalability using both synthetical and real data sets. The second direction of our research is developing a parallel version of pgFCM for distribution memory multiprocessors.

References

[1] C. Ordonez. Programming the K-means clustering algorithm in SQL. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 823–828.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. ACM Computing Surveys, 1999, Vol. 31, Iss. 3, pp. 264–323.

[3] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics, 1973, Vol. 3, Iss. 3, pp. 32–57.

[4] J. C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell, USA, 1981, p. 256.

[5] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and Weingessel A. Machine Learning Open-Source Package 'r-cran-e1071', 2010. http://cran.r-project.org/web/packages/e1071/index.html. Reference date: 13.06.2011.

[6] I. Drost, T. Dunning, J. Eastman, O. Gospodnetic, G. Ingersoll, J. Mannix, S. Owen, and K. Wettin. Apache Mahout, 2010. https://cwiki.apache.org/confluence/display/MAHOUT/Fuzzy+K-Means. Reference date: 13.06.2011.

[7] M. Stonebraker, L. A. Rowe, and M. Hirohama. The Implementation of POSTGRES. IEEE Transactions on Knowledge and Data Engineering, March 1990, Vol. 2, Iss. 1, pp. 125–142.

[8] J. B. MacQueen. Some Methods for Classification and Analysis of MultiVariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, Vol. 1, pp. 281–297.

[9] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling Clustering Algorithms to Large Databases. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, 1998, pp. 9–15.

[10] J. Bezdek, R. Hathaway, M. Sobin, and W. Tucker. Convergence Theory for Fuzzy *c*-Means: Counterexamples and Repairs. IEEE Transactions on Systems, Man and Cybernetics, 1987, Vol. 17, Iss. 5, pp. 873–877.

[11] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: alternatives and implications. Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, 1998, pp. 343–354.

[12] J. Clear, D. Dunn, B. Harvey, M. Heytens, P. Lohman, A. Mehta, M. Melton, L. Rohrberg, A. Savasere, R. Wehrmeister, and M. Xu. NonStop SQL/MX primitives for knowledge discovery. Proceedings of the 5th ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 425–429.

[13] G. Graefe, U. M. Fayyad, and S. Chaudhuri. On the Efficient Gathering of Sufficient Statistics for Classification from Large SQL Databases. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, 1998, pp. 204–208.

[14] C. Ordonez. Integrating K-Means Clustering with a Relational DBMS Using SQL IEEE Transactions on Knowledge and Data Engineering, 2006, Vol. 18, Iss. 2, pp. 188–201.
[15] A. I. Shihab. Fuzzy Clustering Algorithms and their Applications to Medical Image Analysis. PhD thesis, University of London, 2000.
[16] D. Zhang and S. Chen. A Novel Kernelized Fuzzy *c*-Means Algorithm with Application in Medical Image Segmentation. Artificial Intelligence in Medicine, 2004, Vol. 32, pp. 37–50.