World Academy of Science, Engineering and Technology
International Journal of Economics and Management Engineering
Vol:2, No:10, 2008

Determining Cluster Boundaries using Particle Swarm Optimization

Anurag Sharma, and Christian W. Omlin

***Abstract*—**Self-organizing map (SOM) is a well known data reduction technique used in data mining. Data visualization can reveal structure in data sets that is otherwise hard to detect from raw data alone. However, interpretation through visual inspection is prone to errors and can be very tedious. There are several techniques for the automatic detection of clusters of code vectors found by SOMs, but they generally do not take into account the distribution of code vectors; this may lead to unsatisfactory clustering and poor definition of cluster boundaries, particularly where the density of data points is low. In this paper, we propose the use of a generic particle swarm optimization (PSO) algorithm for finding cluster boundaries directly from the code vectors obtained from SOMs. The application of our method to unlabeled call data for a mobile phone operator demonstrates its feasibility. PSO algorithm utilizes U-matrix of SOMs to determine cluster boundaries; the results of this novel automatic method correspond well to boundary detection through visual inspection of code vectors and k-means algorithm.

***Keywords*—**Particle swarm optimization, self-organizing maps, clustering, data mining.

I. Introduction

THE self-organizing map (SOM), developed by Kohonen [5] is a neural network model for the analysis and visualization of high dimensional data. The nonlinear statistical relationships between high-dimensional data are projected into simple topologies such as two-dimensional grids. The SOM thus reduces information while preserving the most important topological relationships of the data elements on the two-dimensional plane. SOMs are trained using unsupervised learning, i.e. no prior knowledge is available and no assumptions are made about the class membership of data.

Several techniques have been proposed for clustering the outputs of SOMs, i.e. their code vectors. Typically, clustering methods based on partitioning or hierarchical methods are applied to cluster the SOM code vectors; however, the solutions found normally do not reflect the clustering suggested by visual inspection of code vectors [4]. The difficulty in detection of cluster boundaries in SOMs has and the resulting problems of interpretability have limited their application to automatic knowledge [1].

Data clustering with SOMs is typically carried out in two stages: first, the data set is clustered using the SOMs which provide code vector, i.e. prototype data points; then, the code vectors are clustered [3]. The use of code vectors in place of the raw data leads to significant gains in the speed of clustering. Our proposed method identifies cluster boundaries using particle swarm optimization on the distribution of code vectors.

The distance between code vectors can be represented and visualized using a so-called *unified distance matrix* (U-matrix); It is possible to manually define cluster boundaries on U-matrices; however, this is a very tedious process and prone to errors. Even when clusters can be identified through visual inspection, the obtained results by different people are not necessarily the same.

Clustering through particle swarm optimization applied to U-matrices overcomes this problem.

A. Self-Organizing Maps

The SOM training algorithm involves essentially two processes, namely vector quantization and vector projection [6]. Vector quantization creates a representative set of vectors, so-called output vectors or prototype vectors from input vectors. This massively reduces the map size to be processed; hence, computation load decreases considerably thereby making clustering of very large datasets feasible. The second process, vector projection, projects output vectors onto a SOM of lower dimension (mainly 2 or 3); this can be useful for data visualization.

The basic SOM model is a set of prototype vectors with a defined neighborhood relation. This neighborhood relation defines a structured lattice, which may be linear, rectangular or hexagonal arrangement of map units. SOMs are trained an unsupervised, competitive learning process. This process is initiated when a winner unit is searched from map units, which minimizes the Euclidean distance measure between data samples x and the map units m_i. This unit is described as the best matching node, the code vector m_c:

$$\|x - m_c\| = \min\{\|x - m_i\|\} \qquad (1)$$
$$where,\ c = \arg\min\{\|x - m_i\|\}$$

Manuscript received August 1, 2006. This work was supported / sponsored by the University of the South Pacific, Suva, Fiji.

Anurag Sharma is with the School of Computing, Information and Mathematical Sciences, the University of the South Pacific, Suva, Fiji (e-mail: sharma_au@usp.ac.fj).

Christian W. Omlin is with the School of Computing, Information and Mathematical Sciences, the University of the South Pacific, Suva, Fiji (e-mail: omlin_c@usp.ac.fj).

Then, the map units are updated in the topological neighborhood of the winner unit, which is defined in terms of the lattice structure. The update step can be performed by applying

$$m_i(t+1) = m_i(t) + h_{ci}(t)\left[x(t) - m_i(t)\right] \quad (2)$$

where t is an integer, the discrete-time coordinate and $h_{ci}(t)$ is the so-called neighborhood kernel defined over the lattice points. The average width and form of h_{ci} defines the "stiffness" of the "elastic surface" to be fitted to the data points. The last term in the square brackets is proportional to the gradient of the squared Euclidean distance $d(x,m_i) = ||x-m_i||^2$. The learning rate $\alpha(t) \in [0,1]$ must be a decreasing function over time and the neighborhood function $h^c(t,i)$ is a non-increasing function around the winner unit defined in the topological lattice of map units. A typical choice is a Gaussian around the winner unit defined in terms of the coordinates **r** in the lattice of neurons [7], [9].

$$h^c(t,i) = \alpha(t).\exp\left(-\frac{\left\|r^i - r^c\right\|}{2\sigma(t)^2}\right) \quad (3)$$

Training is an iterative process which chooses a winner unit by the means of a similarity measure and updates the values of code vectors in the neighborhood of the winner unit. This iterative process is repeated until convergence is reached. All the output vectors are projected on to a 1 or 2 dimensional space, where each neuron corresponds to an output vector that is the representative of some input vectors [6].

SOMs will represent areas of high data densities with many map units whereas only few code vectors will represent sparsely populated areas. Thus, SOMs approximate the probability density function of the input data [9].

Self-organizing maps may be visualized by using a unified distance matrix (U-matrix) representation, which visualizes the clustering of the SOM by calculating distances between map units. An alternative choice for visualization is Sammon's mapping which projects high-dimensional map units onto a plane by minimizing the global distortion of inter point distances.

B. Unified Distance Matrices

A unified distance matrix (U-matrix) representation of SOMs visualizes the cluster structure by calculating the distances between map unit and mean/median distances from its neighbors. The distance ranges are represented by different colors (or grey shades). Dark shades represent large distance, i.e. big gaps exist between the code vector values in the input space; light shades represent small distance, i.e. map units are tightly clustered together. U-matrices are useful tools for visualizing clusters in input data without having any priori information about the clusters [9].

C. Particle Swarm Optimization

The Particle swarm Optimization (PSO) algorithm was originally designed to solve continuous and discrete problems of large domain [11]. It is a generic algorithm to solve optimizing engineering problems. PSO is inspired by the social behavior of bird flocking or fish schooling.

The analogy involves simulating social behavior among individuals (particles) "flying" through a multidimensional search space, each particle representing a single intersection of all search dimensions. The particles evaluate their positions relative to a goal (fitness) at every iteration, and particles in a local neighborhood share memories of their "best" positions; they use those memories to adjust their own velocities, and thus subsequent positions [2].

The original PSO formulae define each particle as potential solution to a problem in D-dimensional space. The position of particle i is represented as

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \quad (4)$$

Each particle also maintains a memory of its previous best position, represented as

$$P_i = (p_{i1}, p_{i2}, ..., p_{iD}) \quad (5)$$

A particle in a swarm is moving; hence, it has a velocity, which can be represented as

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{iD}) \quad (6)$$

At each iteration, the velocity of each particle is adjusted so that it can move towards the neighborhood's best position known as *lbest* (P_i) and global best position known as *gbest* (P_g) attained by any particle present in the swarm [2].

After finding the two best values, each particle updates its velocity and positions according to (7) and (8) weighted by a random number *c1* and *c2* whose upper limit is a constant parameter of the system, usually set to value of 2.0 [11].

$$V_i(t+1) = V_i(t) + c1 \cdot (P_i - X(t)) + c2 \cdot (P_g - X_i(t)) \quad (7)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (8)$$

All swarm particles tend to move towards better positions; hence, the best position (i.e. optimum solution) can eventually be obtained through the combined effort of the whole population.

II. Experiments

In this paper, clustering of data is done in three steps: first, we cluster data with SOMs which produces prototype vectors and use U-matrices for data visualization. Visualization alone does not extract clusters; we use PSO to automatically extract cluster from U-matrices.

For this study, we use call detail record for prepaid service subscribers from a real mobile telecommunication network. The data set contains 500 masked subscribers, each with calling data for a period of 6 months. We based our

investigation on Mobile Originating Calls (MOC) extracted from the data set. These are calls that were initiated by the subscribers within a 6-month period [7]. We have selected only 5 components/fields for data mining. The fields are:

1. Party Called
2. Cell ID
3. Area code
4. Peak hour
5. Call length

Table I shows few samples of the MOC data set.

TABLE I
SAMPLES OF ORIGINAL CALL DATA

Subscriber ID	27826779051	27826708859	27826764710
party called id	836521312	845150532	829531076
cell ID	11242	26300	12402
area code	142	434	151
called time stamp	16/04/2003 14:10	15/04/2003 14:26	5/4/2003 18:19
call length	27	28	50

Since all the selected fields except *call length* are symbolic data, we need to convert them to numeric data for processing. The frequency of each feature with respect to the subscriber is calculated. For example, the number of times one particular subscriber called another party is the frequency of the feature *Party Called* for a given subscriber.

TABLE II
DATA CONVERSION TECHNIQUE FROM SYMBOLIC TO NUMERIC

Subscriber ID	**Party Called ID**
27822535935	723746300
27822535935	723746300
27822535935	723746300
27822535935	827062435

(a)

Subscriber ID	**Party Called ID**	**Frequency**
27822535935	723746300	3
27822535935	827062435	1

(b)

Subscriber ID	**Party Called ID**	**Frequency**	**Rank**
27822535935	723746300	3	1
27822535935	827062435	1	2

(c)

Table II (a) shows that one subscriber has called other two parties a number of times, (b) shows number of times each party has been called and (c) shows the frequency arranged in descending order; ranks are assigned to party called from higher frequency to lower frequency. In this way, symbolic data has been converted to numeric data.

A. Visual Inspection

We have used SOM-toolbox version 2.0 Beta developed by [10]. As mentioned above, we have extracted non-duplicate records of call data where transaction type is MOC. In the preprocessing stage, we used linear scaling on all the feature values whose variance is normalized to one. This method uses principal component analysis (PCA) on the input data. The code vectors are initialized to lie in the same input space that is spanned by two eigenvectors corresponding to the largest eigenvalues of the input data. Finally, we chose 195 maps with topology size of 13x15 units to be trained with batch training algorithm; this results in good cluster visualizations.

An initial idea of the number of clusters in the SOM, as well as their spatial relationships, is usually acquired by visual inspection of the map. The most widely used methods for visualizing the cluster structure of the SOM are distance matrix techniques especially the unified distance matrix [8].

Our cluster boundaries are based on a U-matrix representation as shown in Fig. 1. It appears to have 6 clusters which are vaguely distinguished. Clusters from Fig. 1 can be selected manually but it would tedious process and it would also not guarantee consistency. Hence, we aim to develop automated methods [8].

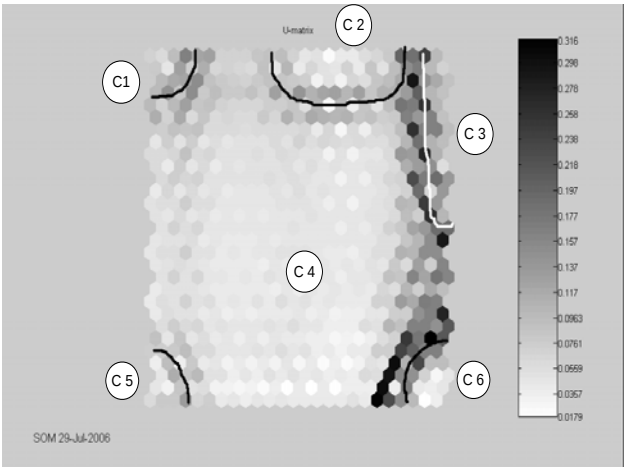

Fig. 1 Graphical view of U-matrix of call data indicates 6 clusters which are labeled as C1, C2, …, C6. The distances between code vectors are represented by grey shades. The shaded scale on right hand side shows the distance measurement. The darker the shades, the greater the distance between the code vectors. Hence darker region indicates cluster boundary and lighter region indicates cluster itself

Our automated method uses a generic particle swarm optimization algorithm to find cluster boundaries in U-matrices. Since a U-matrix contains about the cluster structure, we have utilize this information for PSO initialization. We simply approximate cluster centers and its boundary size. PSO uses these constraints as initial conditions to search the best cluster boundary through optimization technique.

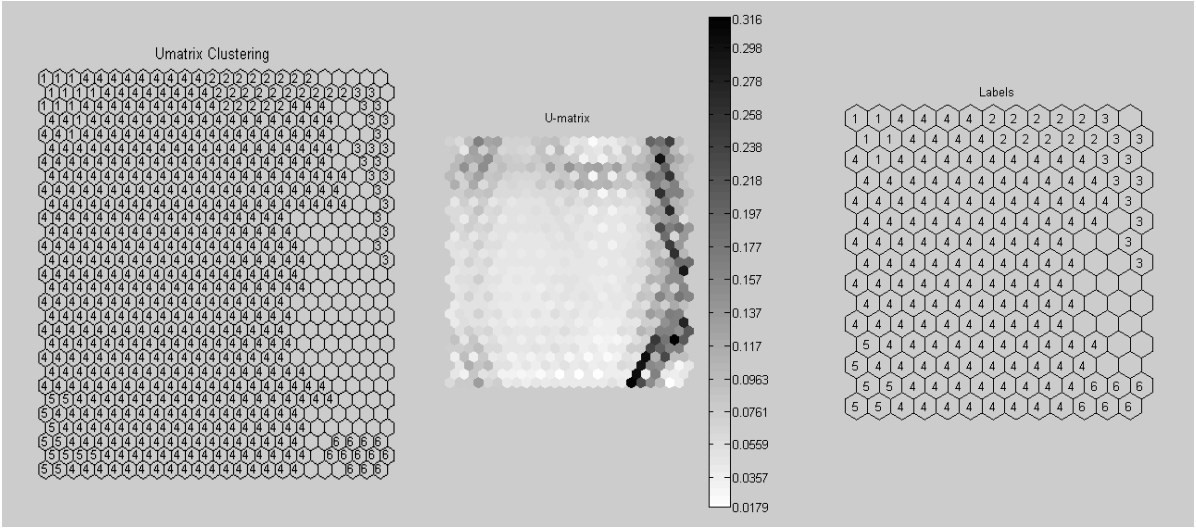

Fig. 2 (a) U-matrix of call data labeled through PSO algorithm. (b) Graphical U-matrix of call data. (c) Prototype vectors of corresponding U-matrix with labels

The basic idea behind this method is to shrink the boundary iteratively. The swarm particles need to move from one place to another with some velocity which requires two chosen indices to be swapped from their initial positions to get new positions. Our objective function uses these indices to shrink (move the boundary one position towards cluster center) the boundary at the point where they lie.

B. Clustering with Particle Swarm Optimization

The clustering algorithm proceeds as follows:

1) Initialize the principal cluster boundary along with a set of neighboring boundaries – inner and outer boundaries - and the corresponding cluster center. The initial cluster boundary is approximated as rectangular shape and cluster centers are approximated as middle point of denser regions through visual inspection of U-matrix. Now for each iteration of PSO:

2) Adjust ranks: PSO generates particles' positions (sequence/array of numbers) and changes their positions by picking 2 indices and swapping them. We call these indices ranks. These sequences have constant size; they are utilized as cluster boundary whose size changes iteratively. We simply arrange the ranks corresponding to the size of shrunken cluster boundary i.e. mathematically:

*new_rank = ceil(rank * SOM_boundary_size/PSO_sequence_size);*

3) Shrink cluster boundary: Shrink the cluster boundary gradually towards cluster center by picking both ranks (indices) and moving it one row and/or column inwards.

4) Calculate average weight: Since the initial principal boundary is extracted from a U-matrix, each weight unit is the distance between clusters or neighborhood distance. Additionally, some neighboring boundaries are calculated; this improves the search for cluster boundaries. If the weight unit of an outer boundary, i.e. a boundary that is further away from the corresponding cluster center than the principal boundary, is higher than the weight unit of the principal boundary, then we add to the since it is likely to be a cluster boundary. If weight units for either the principal or any of the inner boundaries are lower, then again we increase their corresponding weights. In some cases, we can also decrease weights; for example, if the weights for some outer boundary are lower than the corresponding values for the principle boundary, then the outer boundary is in fact not a boundary; instead, it belongs to the interior of another cluster. Similarly, inner boundaries (interior part of cluster) whose weights are higher than the corresponding values of the principal boundary in fact do not belong to the interior of the cluster; rather it is part of the cluster boundary.

5) Solution: Repeated application steps 2 to 4 results in a maximum average weight, which corresponds to the best cluster boundary.

The best boundary is the one which has highest average value of distances between points and its neighbors.

We have used PSO version (4.2) developed by Maurice Clerc. Cluster boundaries of all the 6 clusters were found through PSO.

In our experiment, we used 195 map units which are arranged in 15X13 lattice hence U-matrix has 29X25 distance values. Fig. 2 (b) shows graphical U-matrix of call data, (a) shows clustering result identified through PSO on U-matrix, where same numbers belong to the region of same cluster and (c) shows the corresponding code vectors that are also labeled. As discussed in Section 2.A, 6 clusters have been identified visually; we subsequently applied the PSO algorithm to find improved cluster boundaries. We only need two initial conditions, i.e. cluster center and maximum boundary limit, which can be approximated through visual inspection. For instance, to get the cluster boundary at the top-middle area of U-matrix indicated by C2 in Fig. 1, the cluster center can be

located near 1st row and 16th column of U-matrix and its boundary limits can be taken as 6X12 matrix units which can easily enclose that cluster. The PSO algorithm has produced an expected cluster boundaries for all the clusters whose labeled U-matrix map units are shown in Fig. 2 (a). U-matrix's corresponding map units i.e. code vectors are shown in Fig 2 (c) which are also labeled with corresponding label numbers. These numbers indicate that they belong to the same numbered clusters shown in Fig. 2.

C. Measure Cluster Quality

We have used the *square-error criterion* to measure the quality of clusters. The square-error for a cluster is the sum of the squared Euclidean distances between each sample of that cluster and its centroid. This error is also called *within-cluster variation* [12]:

$$e_k^2 = \sum_{i=1}^{n_k} (x_{ik} - M_k)^2 \quad (9)$$

Where N is number of samples in *n* dimensional space in *K* clusters $\{C_1, C_2, \ldots, C_k\}$. x_{ik} is the i^{th} sample belonging to cluster C_k and M_k is the centroid of cluster i.e.

$$M_k = (1/n_k) \sum_{i=1}^{n_k} x_{ik} \quad (10)$$

We have tested and compared the square-error of clusters obtained though PSO with visual inspection of SOM and k-means algorithm. Experimental results are shown in Table III. Each row indicates square-error of clusters with different methods. It is clear from the results that PSO has performed the best with minimum total square-error. Even for the individual clusters PSO shows minimum error in general.

TABLE III
COMPARATIVE ANALYSIS OF K-MEANS, VISUAL INSPECTION OF SOM AND PSO ALGORITHM BASED ON THE *SQUARE-ERROR* FOR ALL 6 CLUSTERS

Cluster	k-means	Visual Inspection of SOM	PSO
1	8.9584×10^8	2.9405×10^8	1.9154×10^8
2	1.5141×10^9	8.8548×10^8	5.2245×10^8
3	4.2245×10^8	4.8196×10^8	4.3196×10^8
4	3.3214×10^9	5.6417×10^9	6.2689×10^9
5	1.6318×10^9	4.3605×10^8	3.8992×10^8
6	6.9328×10^8	7.3955×10^8	6.7410×10^8
Total	**1.6318×10^9**	**4.3605×10^8**	**3.8992×10^8**

III. CONCLUSION

We have proposed a solution for the automatic identification of cluster boundaries for SOM code vectors through particle swarm optimization (PS0). The swarm particles iteratively shrink several cluster boundaries to get the most suitable one. The idea behind this algorithm is to find clusters in the unified distance matrix (U-matrix) where map units form a denser region, i.e. SOM output vectors are close to each other. Conversely, lighter regions (with dark shades in U-matrices) indicate cluster boundaries. The total cluster size depends on the number of clusters visualized through the U-matrix. The algorithm works well with small sized problems but processing speed and efficiency degrade as the size increases. Some other generic clustering algorithms can also be tested and compared with the solution found by PSO.

REFERENCES

[1] A. Rauber and D. Merkl, "Automatic Labelling of Self-Organizing Maps: Making a Treasure Maps Reveal Its Secrets," in *Proc. 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD99. Beijing, China, (1999).

[2] A. Carlistle, and G. Dozier. (1998). Adapting Particle Swarm Optimization to Dynamic Environments [Online]. Available: http://www.CartistleA.edu

[3] J. Vesanto, and E. Alhoniemi, "Clustering of the Self-Organizing Map," *IEEE transaction on neural network*, Vol. 11, No. 3, May 2000.

[4] J. Vesanto, and M. Sulkava, "Distance matrix based clustering of the self-organizing map," in *Proc. International Conference on Artificial Neural Networks – ICANN 2002*, Lecture Notes in Computer Science, No. 2415, pages 951-956. Springer-Verlag, 2002.

[5] T. Kohonen, Self-Organizing Maps. Springer-Verlag, Berlin, Germany, 2001.

[6] B. Jiang, and L. Harrie, "Selection of streets form a network using self-Organizing maps," *Transactions in GIS*, Vol 8(3), pages 335-350, 2004.

[7] O. Abidogun, "Data Mining, Fraud Detection and Mobile Telecommunication: Call pattern Analysis with Unsupervised Neural Networks," M.Sc. thesis, University of the Western Cape, Bellville, Cape Town, South Africa, 2004.

[8] J. Vesanto, and E. Alhoniemi, "Clustering of the Self-Organizing Map," *IEEE Transaction on Neural Neetworks*, Volume 11, Number 3, pages 586-600, 2000.

[9] J. Hollmén, "Process Modelling using the Self-Organizing Map," M.Sc. thesis, Dept. Computer Science, Helsinki Univ. of Technology, Finland, 1996.

[10] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "SOM Toolbox for Matlab 5," ISBN 951-22-4951-0, ISSN 1456-2243, Espoo, Finland, 2000.

[11] J. Kennedy, and R. C. Eberhart, "The particle swarm: social adaptation in information processing systems," In Corne, D., Dorigo, M., and Glover, F., Eds., *New Ideas in Optimization*. London: McGraw-Hill, pp. 379-387, 1999.

[12] M. Kantardzic, "Cluster Analysis," Data Mining – concepts, models, methods, and algrorithms. Wiley InterScience, pp 129-132, 2003.