ECM — A Novel On-line, Evolving Clustering Method and Its Applications

Qun Song [1] and Nikola Kasabov [2]

[1, 2] Department of Information Science, University of Otago
P.O Box 56, Dunedin, New Zealand

(E-mail: [1] qsong@infoscience.otago.ac.nz, [2] nkasabov@otago.ac.nz)

Abstract

In this paper, an on-line, dynamic clustering method — ECM, Evolving Clustering Method, is proposed. The ECM is usually used for on-line systems, in which it performs an one-pass, maximum distance-based clustering process without any optimisation. The ECM can also be applied to off-line problems where a constrained minimisation is introduced.

1. Introduction

To design a fuzzy inference system, we need to partition the input sample space to create fuzzy rules. The partitioning should be carried out in an on-line mode if the fuzzy systems are required to be created and modified dynamically. The ECM was specially designed for the DENFIS, *Dynamic Evolving Neural Fuzzy Inference System* [1, 2], and in this case, the ECM dynamically performs scatter partitioning of the input space for the purpose of creating fuzzy inference rules and evolving a fuzzy system. For the similar purpose, the ECM can also be fuzzified and applied to the EFuNN, *Evolving Fuzzy Neural Network* [3].

Although the ECM suits on-line, dynamic systems it can also be applied to off-line tasks, e.g. *the K-means clustering* [4] can get initial value made by an ECM more effective than made by using of a random method. In the off-line cases, a constrained optimisation is applied to the ECM, which makes a pre-defined objection function, based on a distance measure, to reach a minimum value subject to the given constraints.

In this paper, we describe the basic principle of ECM and its algorithm and then, we show some examples of ECM application and comparison with some other well-known clustering methods.

2. The Principle and Algorithm of ECM

2.1 ECM's Principle

The ECM is a fast, one-pass algorithm for dynamic clustering of an input stream of data. It is a distance-based clustering method where the cluster centres are represented by evolved nodes in an on-line mode. For any such cluster the maximum distance, *MaxDist*, between an sample point, which belongs to one cluster and is the farthest from this cluster centre, and its cluster centre, is less than or equal to a threshold value, *Dthr*, that has been set as a clustering parameter. This parameter would affect the number of clusters to be created.

In the clustering process, the data samples come from a data stream and this process starts with an empty set of clusters. When a new cluster is created, its cluster centre, ***Cc***, is located and its cluster radius, *Ru*, is initially set with a value 0. With following samples presented one after another, some already created clusters will be updated through changing their centres' positions and increasing their cluster radiuses. Which cluster should be updated and how it should be changed, depend on the position of the current data sample. A cluster will not be updated any more when its cluster radius, *Ru*, has reached the special value that is, usually, equal to the threshold value *Dthr*.

2.2 ECM's Algorithm

The ECM algorithm is given below as a procedure and the Figure 1 shows a brief ECM clustering process in a 2-D space.

- Step 1: Create the first cluster C_1 by simply using the first sample from the input data stream and taking its position as the first cluster centre $\boldsymbol{Cc}_1$, and initially setting the cluster radius Ru_1 with a value 0 (Figure 1a).
- Step 2: If all samples from the data stream have been presented, the clustering process finishes. Else, the current input sample, $\boldsymbol{x}_i$, is taken and the *normalised Euclidean distances* $d(i, j)$, between this sample point and all n already created cluster centres $\boldsymbol{Cc}_j$,

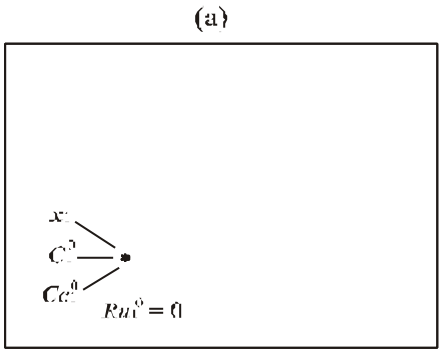
$$d(i, j) = \| \boldsymbol{x}_i - \boldsymbol{Cc}_j \|, \; j = 1, 2, \ldots, n, \qquad (1)$$

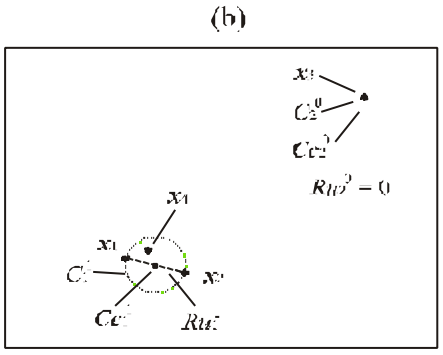
are calculated. In this paper, the *distance* between two q-element vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ means a *normalised Euclidean distance* defined as follows:

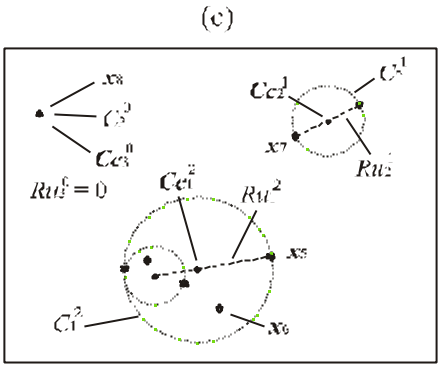
$$\| \boldsymbol{x} - \boldsymbol{y} \| = \left(\sum_{i=1}^{q} | x_i - y_i |^2 \right)^{1/2} / q^{1/2} \qquad (2)$$

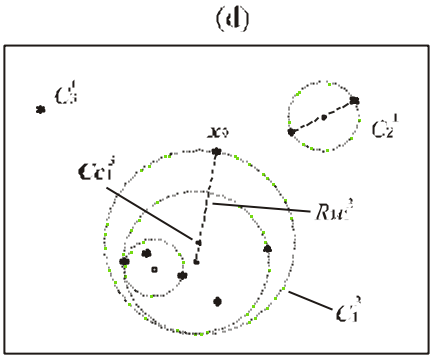
where $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{R}^q$.

- Step 3: If there is a cluster C_m with its centre $\boldsymbol{Cc}_m$; cluster radius Ru_m; and a distance value $d(i, m)$, which is between the cluster centre $\boldsymbol{Cc}_m$ and the sample $\boldsymbol{x}_i$ and defined as follows:









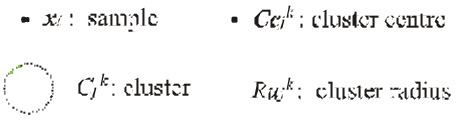


Figure 1. A brief clustering process using ECM with examples $\boldsymbol{x}_1$ to $\boldsymbol{x}_9$ in a 2-D space:

(a) The example $\boldsymbol{x}_1$ causes the ECM to create a new cluster C_1^0

(b) $\boldsymbol{x}_2$: update cluster $C_1^0 \rightarrow C_1^1$
$\boldsymbol{x}_3$: create a new cluster C_2^0
$\boldsymbol{x}_4$: do nothing

(c) $\boldsymbol{x}_5$: update cluster $C_1^1 \rightarrow C_1^2$
$\boldsymbol{x}_6$: do nothing
$\boldsymbol{x}_7$: update cluster $C_2^0 \rightarrow C_2^1$
$\boldsymbol{x}_8$: create a new cluster C_3^0

(d) $\boldsymbol{x}_9$: update cluster $C_1^2 \rightarrow C_1^3$

$$d(i, m) = \min_j d(i, j) = \min_j (\| \boldsymbol{x}_i - \boldsymbol{Cc}_j \|),$$
$$j = 1, 2, \ldots, n. \qquad (3)$$

and, $d(i, m) = Ru_m$,

It is regarded that the current sample $\boldsymbol{x}_i$ belongs to the cluster C_m. In this case neither a new cluster is created, nor any existing cluster is updated (e.g. data vectors $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$ in Figure 1). The algorithm then returns to Step 2.

Else,

- Step 4: Find a cluster C_a, with its centre $\boldsymbol{Cc}_a$; cluster radius Ru_a and the distance value $d(i, a)$ from all n existing clusters through calculating the extended distance values:

$$s(i, j) = d(i, j) + Ru_j, \quad j = 1, 2, \ldots, n, \tag{4}$$

and then selecting the cluster C_a with the minimum value $s(i, a)$:

$$s(i, a) = d(i, a) + Ru_a = \min_j s(i, j)$$

$$j = 1, 2, \ldots, n. \tag{5}$$

- Step 5: If $s(i, a) > 2Dthr$, the sample $\boldsymbol{x}_i$ can not belong to any existing clusters. A new cluster should be created in the same way as described in Step 1 (e.g. input data vectors $\boldsymbol{x}_3$ and $\boldsymbol{x}_8$ in Figure 1). The algorithm then returns to Step 2.

Else,

- Step 6: If $s(i, a) = 2Dthr$, the cluster C_a is updated by moving its centre, $\boldsymbol{Cc}_a$, and increasing its radius value, Ru_a. The updated radius Ru_a^{new} is set to be equal to $s(i, a)/2$ and the new centre $\boldsymbol{Cc}_a^{new}$ is located on the line connecting input vector $\boldsymbol{x}_i$ and the old cluster centre $\boldsymbol{Cc}_a$, so that the distance from the new centre $\boldsymbol{Cc}_a^{new}$ to the sample point $\boldsymbol{x}_i$ is equal to Ru_a^{new} (e.g. input data points $\boldsymbol{x}_2$, $\boldsymbol{x}_5$, $\boldsymbol{x}_7$ and $\boldsymbol{x}_9$ in Figure 1). The algorithm then returns to Step 2.

In this way, the maximum distance from any cluster centre to the farthest sample that belongs to this cluster, is kept within the threshold value *Dthr*, although the algorithm does not keep any information of passed samples.

2.3 ECM with Constrained Optimisation

After an one-pass clustering described in last section, the ECM has created n clusters, C_j, and found the corresponding cluster centres, $\boldsymbol{Cc}_j$, , $j = 1, 2, \ldots, n$. And then, the ECM further minimises an objective function based on a distance measure. For keeping the maximum distance less than or equal to the threshold value, the minimising process must implement subject to the constraints. Taking the *normalised Euclidean distance* (Equation 2.2), between the sample vector $\boldsymbol{x}_k$, belonging to a cluster C_j, and the corresponding cluster centre $\boldsymbol{Cc}_j$, as the measure, the objection function is defined as the follows:

$$J = \sum_{j=1}^{n} J_j = \sum_{j=1}^{n} \left(\sum_{k, \boldsymbol{x}_k \in C_j} \| \boldsymbol{x}_k - \boldsymbol{Cc}_j \| \right), \tag{6}$$

where $J_j = \sum_{k, \boldsymbol{x}_k \in C_j} \| \boldsymbol{x}_k - \boldsymbol{Cc}_j \|$ is the sub-objection function

within cluster C_j, $j = 1, 2, \ldots, n$,

and the constraints are defined as follows:

$$\| \boldsymbol{x}_k - \boldsymbol{Cc}_j \| = Dthr, \tag{7}$$

where $k, \boldsymbol{x}_k \in C_j$ and $j = 1, 2, \ldots, n$.

The clusters are typically defined as an $p \times n$ binary membership matrix $\boldsymbol{U}$, where the element u_{ij} is set to 1 if the i-th data point $\boldsymbol{x}_i$ belongs to the j-th cluster C_j; and 0 if otherwise. Once the cluster centres $\boldsymbol{Cc}_j$ are defined, the values u_{ij} are derived as follows:

if $\| \boldsymbol{x}_i - \boldsymbol{Cc}_j \| = \| \boldsymbol{x}_i - \boldsymbol{Cc}_k \|$, for each $j \neq k$;

$$u_{ij} = 1 \text{ else } u_{ij} = 0. \tag{8}$$

The ECM minimising algorithm works in an off-line, iterative mode on a batch of data repeating the following steps:

- Step1: Initialise the cluster centre $\boldsymbol{Cc}_j$, $j = 1, 2, \ldots, n$, which are produced by the ECM one-pass, on-line clustering.
- Step2: Determine the membership matrix $\boldsymbol{U}$ by using Equation (8).
- Step3: Employ the *constrained minimisation method* [5] with Equation (6) and (7) to obtain new cluster centres.
- Step4: Calculate the objective function J according to Equation (7). Stop if the result is below a certain tolerance value, or its improvement over previous iteration is below a certain threshold, or the iteration number of optimising operation is over a certain value. Else, the algorithm returns to Step2.

3. Examples: Applications of ECM for On-line and Off-line Problems

3.1 Example 1: On-line Clustering on Gas-furnace Data Set

In this example, we use the *gas-furnace time series data set*, which consists of 292 consecutive data pairs. In this case, the clustering simulations are implemented in the input space (two dimensions). For the purpose of comparative analysis the following six clustering methods are applied on the same data set:

(a) ECM, evolving clustering method (on-line, one-pass)
(b) EFuNN, evolving fuzzy-neural network clustering (standard mode, on-line, one pass)
(c) SC, subtractive clustering [6] (off-line, one pass)
(d) ECMc, evolving clustering with constrained optimisation (off-line)
(e) FCMC, fuzzy C-means clustering [7] (off-line)
(f) KMC, K-means clustering (off-line)

Each of them partitions the data set into *NoC* (= 15) clusters. The maximum distances, *MaxDist*, defined in section 2.1, and the values of the objection function *J*, defined by Equation (6), are measured for comparison and shown in Table 1. We can see from Table 1 that both ECM and ECMc obtain minimum values of *MaxDist* for on-line and off-line clustering, which indicates that these methods partition the data set more uniformly than other methods. Looking at the results from a different point of view, we can state that if all these clustering methods obtained the same value of *MaxDist*, ECM would result in a less number of clusters. Using the ECMc we can obtained a satisfying value of the objection function, and considering that the ECM clustering is an 'one-pass' on-line process, the objection value *J* for ECM simulation is acceptable as it is comparable with the *J* value produced by other methods.

Methods	*MaxDist*	Objective value: *J*
ECM	0.1	12.9
EFuNN	0.11	13.3
SC	0.15	11.5
ECMc	0.1	11.5
FCM	0.14	12.4
KM	0.12	11.8

Table 1 The results of clustering Gas-Furnace data set

3.2 Example 2: EFuNN with A Fuzzified ECM for Prediction on Mackey-Glass Data Set

In this example we use both standard EFuNN and EFuNN with a fuzzified ECM for a prediction task on Mackey-Glass time series data set [8]. To improve the EFuNN model using ECM technique, the ECM has to be fuzzified to become a part of EFuNN. We term it as ECMf, which has the algorithm similar to the ECM but with two differences:

(1) An ECMf works in the fuzzy input space (in the case of clustering), or both fuzzy input space and fuzzy output space (in the case of EFuNN) on a fuzzy data set – a fuzzified crisp training data set, instead of working in the input space on a crisp data set.
(2) A *normalised fuzzy Euclidean distance* is used to measure similarity instead of the *normalised Euclidean distance* used in the ECM.

The task is to predict value $x(t+6)$ from the input vector $[x(t–18)\ \ x(t–12)\ \ x(t–6)\ \ x(t)]$. Both training data set and testing data set include 500 data pairs. The prediction results including number of nodes, *RMSE* – Root Mean Square Errors and *NDEI* – Non-Dimension Error Index are listed in Table 2. From the results, it can be learnt that compared with a standard EFuNN model, the EFuNN with ECMf can obtain more accurate prediction results, both in on-line learning and off-line testing, with less number of rule nodes.

Methods	EFuNN with ECMf	Standard EFuNN
Rule Nodes	72	76
Learning RMSE	0.066	0.07
Learning NDEI	0.289	0.309
Testing RMSE	0.036	0.038
Testing NDEI	0.159	0.168

Table 2 The results of prediction on Mackey-Class Time-series data set using EFuNN with ECMf and standard EFuNN

3.3 Example 3: ECMs – An Extension of ECM for Supervised Classification on Two Spirals Problem

Generally, ECM implements in an unsupervised mode, however, it can be extended to a supervised classifier in some cases. We assume that a supervised classification task is to establish a classifier by learning a data set $\{\boldsymbol{x}_i\} = \{[x_{i1}, x_{i2}, \ldots, x_{ip}]\}$, $i = 1, 2, \ldots, l$, which respectively belong to q subsets (classes).

The idea of ECMs is to apply ECM to every class subset for finding the cluster centres:

$n_c = \sum_{k=1}^{q} m_k$, where m_k is the node's number in the k-th

subset.

For every input data vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_p]$, if the classifier finds one node $\boldsymbol{Cc}_{kj}$ from n_c nodes, which is situated in the k-th subset and has the minimum distance to the $\boldsymbol{x}$ in the input space, this input data $\boldsymbol{x}$ belongs to the k-th subset (class).

The two-spirals problem is a well-known benchmark task, which generates data points in a given density. The training data set, generated in density 1, consists of 194 data with 97 data for each spiral. The testing data set, generated in density 4, is composed of 770 data with 385 data for each spiral. The formulas used to generate the spirals are given below, and the training data are shown in Figure 2 (a).

$$\begin{cases} ? = (? + p/2) / p \\ ? = k\ \mathrm{p} / 16, \ k = 0, 1, 2, \ldots, 96. \end{cases}$$

(density 1, for training data)

$$\begin{cases} ? = (? + p/2) / p \\ ? = k\ \mathrm{p} / 64, \ k = 0, 1, 2, \ldots, 384. \end{cases}$$

(density 4, for testing data)

spiral 1:
$$\begin{cases} x = ? \cos(?) \\ y = ? \sin(?) \end{cases}$$

spiral 2:
$$\begin{cases} x = -? \mathrm{con}(?) \\ y = -? \sin(?) \end{cases}$$

The results of two cases of ECMs classification are shown in Figure 2 (b) and (c).

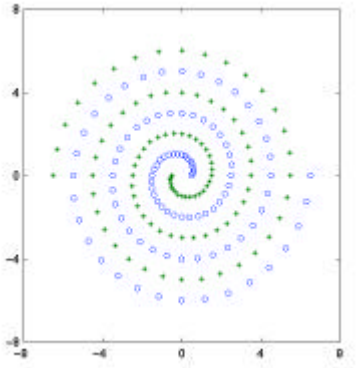


Figure 2 (a) The training data of the two-spirals problem

Figure 2 (b) Decision regions for ECMs with 64 nodes and 98.4% correct rate on the testing data

Figure 2 (c) Decision regions for ECMs with 124 nodes and 100% correct rate on the testing data

4. Conclusions

This paper introduces a new, on-line clustering technique: *Evolving Clustering Method* (ECM), which was specially designed for the on-line, evolving fuzzy inference system. The ECM has a fast 'one-pass' algorithm without any optimisation, its off-line extension, however, applies a constrained optimisation.

The ECM can be used as an independent method to solve some clustering and classification problems and we can see from the results of examples that the ECM, used in both on-line and off-line, are comparable with some other well-known clustering methods.

Further directions for this research include: (1) improve the ECM for on-line adaptive clustering and classification; (2) apply the ECM to the new research: mobile robot navigation and object tracking.

Acknowledgements

This research is part of a research programme funded by the New Zealand Foundation for Research Science and Technology, contract UOOX0016.

References

[1] Song, Q., "Dynamic Evolving Neural-Fuzzy Inference System (DENFIS): On-line Learning and Application for Time-series Prediction", Proc. 6th International Conference on Soft Computing, 696 – 701, Iizuka, Fukuoka, Japan, October, 2000.

[2] Kasabov, N., and Song, Q., "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-series Prediction", IEEE Trans. on Fuzzy Systems, accepted.

[3] Kasabov, N., "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation", in: Yamakawa, T. and G.Matsumoto (eds) Methodologies for the conception, design and application of soft computing, World Scientific, 271-274, 1998.

[4] Platt, J., "A Resource Allocating Network for Function Interpolation", Neural Comp., 3, 213-225, 1991.

[5] Optimization Toolbox User's Guide, The MATH WORKS Inc., 3: 19-24, 1996.

[6] Bezdek, J.C., "Pattern Recognition with Fuzzy Objective Function Algorithms", Plenum Press, New York, 1981.

[7] Chiu, S., "Fuzzy Model Identification Based on Cluster Estimation", Journal of Intelligent & Fuzzy System, Vol. 2, No. 3, Step. 1994.

[8] Mackey, M. C. and Glass, L., "Oscillation and Chaos in Physiological Control systems", Science, 197:287-289, 1977.