A Fast and Stable Incremental Clustering Algorithm

Steven Young, Itamar Arel, Thomas P. Karnowski, Derek Rose
Machine Intelligence Lab
Electrical Engineering and Computer Science Department
University of Tennessee
Knoxville, TN 37996
email: syoung22@utk.edu, itamar@eecs.utk.edu, tkarnows@utk.edu, derek@utk.edu

***Abstract*— Clustering is a pivotal building block in many data mining applications and in machine learning in general. Most clustering algorithms in the literature pertain to off-line (or batch) processing, in which the clustering process repeatedly sweeps through a set of data samples in an attempt to capture its underlying structure in a compact and efficient way. However, many recent applications require that the clustering algorithm be online, or incremental, in the that there is no *a priori* set of samples to process but rather samples are provided one iteration at a time. Accordingly, the clustering algorithm is expected to gradually improve its prototype (or centroid) constructs. Several problems emerge in this context, particularly relating to the stability of the process and its speed of convergence. In this paper, we present a fast and stable incremental clustering algorithm, which is computationally modest and imposes minimal memory requirements. Simulation results clearly demonstrate the advantages of the proposed framework in a variety of practical scenarios.**

I. Introduction

We focus here on clustering as an unsupervised learning process, which attempts to represent the partitions of data of unknown class origin without feedback or information beyond what is inherently gained from the data. Its purpose is to find underlying groups, or clusters, which ideally share similar features. Most commonly, the purpose of unsupervised clustering is to autonomously learn how to best discretize a space with the intent of classifying unseen data samples into one of several clusters with the assumption that commonly classed samples share a common features. As class labels are unnecessary for training or adjusting parameters, learning here implies a presentation of a set of samples that need not be segmented into training, test, and validation subsets. In this paper, we introduce a new approach to incremental or online clustering that puts strict restraints on centroid estimation and modification in an attempt to handle the stability/plasticity dilemma that plagues all data-driven systems. As will be detailed in the following section, our approach augments traditional competitive learning clustering algorithms.

Reaching stability implies that observations are consistently assigned or associated with centroids. In practice, incremental clustering reaches stability when centroid means are changing with very small tolerance. Thus the average centroid change rate approaches zero as more observations are made. To reach stability, it is most common to utilize a learning rate which decays with time or metric to estimate or measure stability. There is a trade-off in flexibility or plasticity to accommodate new data which is made as the learning rate decays. Assuming stability is reached, this effectively means that data samples encountered after a pseudo steady state have smaller effect on subsequent centroid locations. Parameter selection and adaptation is particularly important when attempting to assimilate new data or observations into an existing set of centroids.

It should be noted that there are alternative fundamental clustering approaches that share similar goals with this work. Hierarchical clustering methods build representations by grouping data in a tree of clusters and can be either agglomerative or divisive. Both perform splitting or merging of data as the tree forms though special considerations must be made in the case of handling incremental data samples, particularly with mind to tree size and balance. BIRCH is a hierarchical method which handles large data sets incrementally, compactly, and efficiently [1]. BIRCH suffers from being able to only spherically represent clusters, work with low dimensional data, and further may not be suspended or stopped. However BIRCH doesn't depend as greatly on data presentation order as other methods such as CLARANS [2]. Partition based measures, which break incoming data into sets represented by centroids, are generally separated into two types based on their objective criterion. Methods like k-means and k-modes seek to minimize mean square error [3] while others form maximum likelihood estimates of Bayesian probabilities using expectation maximization [4]. Fuzzy membership clustering methods, such as fuzzy C-means allow a data sample to belong to more than one cluster, effectively giving each cluster a rank of importance in generating a sample [5]. Model based clustering methods, such as SOM nets [6], often require structures (neural networks) that limit their scalability. A more detailed look at the specifics of each type of clustering can be found in the surveys [7], [8], [9].

To briefly highlight the difficulties of online clustering tackled in this paper, consider a system which has dynamic cluster generation centers (or a mutating underlying structure) that produces observations with different probabilities. We must be mindful that, as time progresses, the clusters may move, disappear and (re)appear, or become more or less prolific. Further, we are interested in the ease of implementation, portability (what are the restrictions on use in a system), and the memory and computational requirements.

In Section I, we introduce our proposed framework for efficient incremental clustering using a winner-take-all (WTA)

inspired partition approach. Following in Section II we highlight our simulation results and analysis into multiple case scenarios. We conclude with notes on improving the algorithm as well as future research directions.

II. Proposed Clustering Framework

The proposed clustering algorithm employs the WTA competitive learning paradigm, but with an online aspect in that it continuously updates centroids based on the input data stream, or observations. We assume the latter are not stored in advance such that it is not possible to iterate over the data set in a batch learning manner. We further assume that the algorithm utilizes a finite, fixed number of centroids.

In competitive learning clustering algorithms the learning rate is often adjusted to allow trade offs between faster learning in early phases and stability in later phases. Typically, the learning rate is monotonically decreasing, for example by making it a decaying exponential with the decay a function of the iterations completed. A more data-driven approach taken here assumes an adaptive learning rate which is adjusted as a function of the structure of the data, measured by a few performance and behavioral characteristics. Thus to describe the algorithm we must first introduce several key concepts unique to our formulation.

The first mechanism used is driven by the desire to guarantee unbiased treatment of all centroids, regardless of the input sample distribution. To achieve this goal, we employ a pseudo-entropy (PE) metric [10]. A very similar metric has been used in the past for batch clustering applications [11]. The formulation for the PE of a vector v of length D is given by

$$PE(v) = 1 - \frac{\sum v_i \log(v_i)}{\log(D^{-1})}, \tag{1}$$

where $\sum v_i = 1$.

This definition is clearly related to the entropy of v (assuming it is normalized) scaled by the maximum entropy of any vector of size D. The scaling and subtraction from unity means that PE is zero when the elements of v are all equal (i.e. form a uniform distribution) and is maximum at unity when all elements but one are zero. As will be detailed below, the PE serves as an estimate of how well the overall available centroids are covering the input data and thus helps diminish the updating of the centroids, but allows the algorithm to remain responsive to significant changes in the input sample stream.

Next, we introduce the concept of *starvation trace* comprising of a scalar for each centroid which is used to guarantee that clusters that are initially distant from observations are eventually updated. The starvation trace allows idle or starved centroids to accumulate credit over time when they are not the selected centroid. Once a centroid is selected for update, it loses this credit. The starvation trace is initialized to a constant vector of length D where D denotes the dimensionality of the observation. For each observation, whereby centroid c is chosen, the starvation trace for all non-selected centroids is reduced either by a fixed value or scaled by the PE. The starvation trace brings starved centroids artificially closer to an input sample thereby offering a chance to make movement toward regions of the data samples.

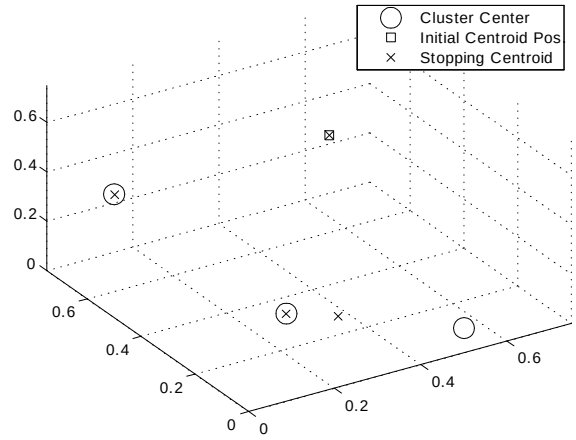

Fig. 1. Example showing lack of desired convergence when not using starvation traces

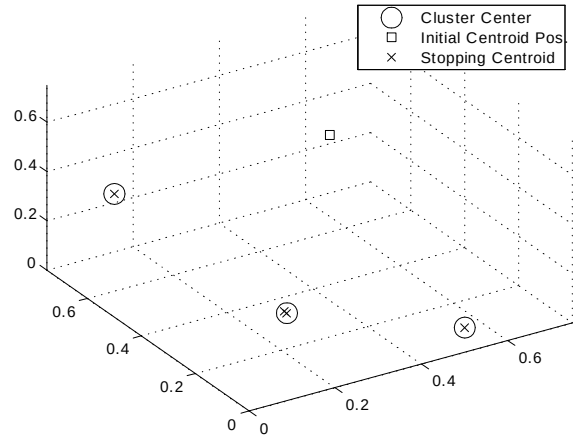

Fig. 2. Example showing desired convergence when utilizing using starvation traces

An example of how the starvation trace impacts the clustering process is illustrated in figures 1 and 2, which are in a 3-dimensional space. In this example, there are three actual clusters (denoted by a single point in the space, displayed as circles), all with uniform probability of occurrence, and the process attempts to fit four centroids to the data. For the purposes of illustration, all estimated cluster centers are set to the initial value of (0.5,0.5,0.5). Without modulating the distance of a centroid from input samples by the starvation trace it can be seen that one or more centroids can indefinitely be excluded from ever being updated from it's initial state,

as depicted in figure 1. Note that the final third centroid is spanning two cluster centers, and since its distance to both centers is shorter than the distance to the initial point, the centroid is simply continually drawn from both sides. Figure 2 shows that once the starvation trace is included, this undesired scenario can be avoided, as the fourth centroid accumulates starvation credits and is eventually chosen over the third centroid as closer to the second center.

A. Stabilizing the Process

A key aspect of a robust incremental clustering algorithm is stability in the sense that once good coverage of the data set has been reached, updating of the centroids should recede. We maintain a metric for each centroid which reflects the normalized standard deviation of that centroid's updates over time. We then modulate the centroid change by this metric to yield a stabilized process.

To achieve the above goal, we maintain two parameters for each centroid: the estimates update rate and its standard deviation. These are computed online and are given by the following expressions

$$\mu_x^{t+1} = \alpha\mu_x^t + (1-\alpha)d_x \tag{2}$$
$$\sigma_x^{t+1} = \beta\sigma_x^t + (1-\beta)\|\mu_x^t - d_x\|$$

where d_x is the distance between an observation and selected centroid x, and $\alpha, \beta < 1$. These estimates are smoothed metrics for the changes made to the centroids over time. As centroids better approximate actual dense areas of data clusters, the values of μ_x should approach a fixed value and the value of σ_x should converge to a relatively low level. To effectively utilize these constructs, we established a functional relationship between them expressed as

$$\hat{\rho}_x = \frac{2}{1 + e^{-\gamma\frac{\sigma_x}{1+\mu_x}}} - 1, \tag{3}$$

where γ is a constant. $\hat{\rho}$ allows us to scale centroid changes in the following manner. When centroids are not changing much μ and σ should be small and consequently $\hat{\rho}$ should be close to 0. Periods of large change in centroids will force $\hat{\rho}$ to be nearly unity. The centroid update change is then modulated by the moving average of $\hat{\rho}$ such that

$$\rho_x^{t+1} = c\rho_x^t + (1-c)\hat{\rho}_x, \tag{4}$$

thus stabilizing the process with respect to accuracy in sample data coverage

With these preliminaries set, we next describe the clustering algorithm. The estimated centroids are initialized to random values. A new observation is assigned to a single estimated centroid based on the minimum distance computed using a distance measure such as the Euclidean distance. The latter is weighted by the starvation trace and the estimate of ρ as defined earlier. That is, we use the augmented distance

$$d_x^t = dist(x^t, o^t) = \|x^t - o^t\|\left(1 - \rho_x^t\left(1 + \phi_x\right)\right), \tag{5}$$

where x^t denotes the centroid, o^t is the observation and ϕ_x is the starvation trace. The centroid selection simply performs $\arg\min_x\ d_x^t$ to select the winning centroid, followed by computation of the PE on the vector

$$PE\left(\frac{1-\rho_x^t}{\sum_x 1-\rho_x^t}\right) \tag{6}$$

which allows us to measure how close the entire vector ρ is to a uniform distribution. Over time, ρ should approach a uniform distribution indicating that effective data coverage and stability have been achieved.

If we define the prior centroid update for centroid x to be $\triangle_x$, then the new step size δ_x^t is computed as

$$\lambda^t = 0.5\left(1 - \frac{\langle x^t, o^t\rangle \triangle_x}{\|x^t - o^t\|\|\triangle_x\|}\right)$$
$$\delta_x^t = e^{-\alpha(1-\lambda^t)} \tag{7}$$

where $\langle\cdot,\cdot\rangle$ denotes a dot product, α is a constant and λ^t is a scalar measure of the cosine of the angle between the prior update center $\triangle_x$ and the new direction of the update $(x-o)$. This adjustment is intended to weaken the step size when there is negative correlation between consecutive directions of update, and strengthen it when there is a consistent centroid update direction. The idea is that when the direction is similar the estimated centroid is moving consistently in the same general direction between updates, and once it gets close to the true cluster centroid the direction is noisier and thus the step should be scaled back. As a simple illustration of this concept, consider a clustering algorithm where the step size is the cosine direction scaled by a fixed constant ϵ. The *direction* is still set based on the difference between the observation and centroid as before. Hence, the update rule can be expressed as

$$x^{t+1} = x^t - \frac{\epsilon\langle x^t, o^t\rangle^2 \triangle_x}{\|x^t - o^t\|^2\|\triangle_x\|}. \tag{8}$$

The step size is smoothed over time such that

$$\delta_x^{t+1} = k\delta_x^t + (1-k)e^{-\alpha(1-\lambda^t)}, \tag{9}$$

resulting in a final update for the winning centroid which is

$$x^{t+1} = x^t - (x^t - o^t)\delta_x^{t+1}\rho_x^{t+1}PE\left(\frac{1-\rho_x^t}{\sum_x 1-\rho_x^t}\right). \tag{10}$$

For reinforcement, we update the cluster centroid by the difference between the centroid and observation $(x^t - o^t)$ weighted by the element of ρ, the step size δ_x^t, and the pseudo-entropy of the entire ρ vector.

III. Simulation Results

In order to establish the robustness and speed of the proposed clustering algorithm, the latter has been evaluated over a range of pathological cases. The following is a description of each of the case studies along with their respective performance analysis.

Case 1: Observations (i.e. data samples) are uniformly distributed among the clusters. This is the most basic of cases that can be presented to the algorithm. The estimated number of necessary centroids represents the data well and each centroid

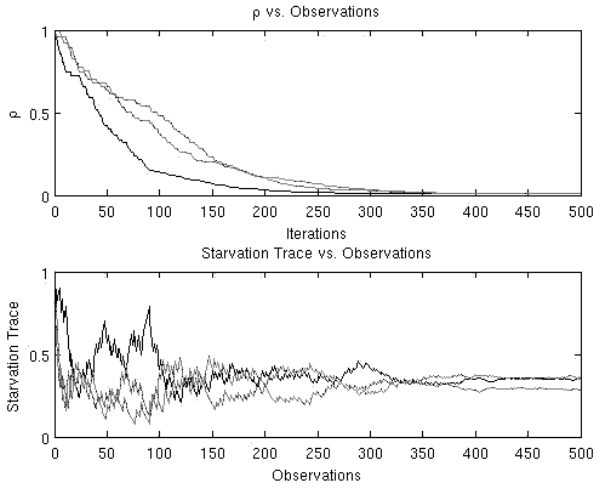

Fig. 3. ρ and starvation trace values for case (1)

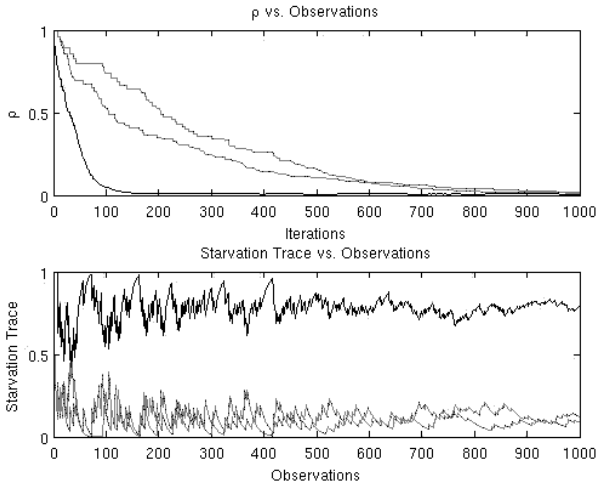

Fig. 5. ρ and starvation trace values for case (2)

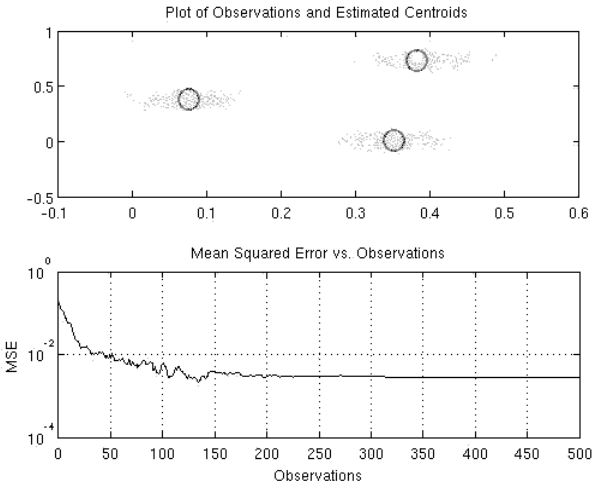

Fig. 4. Centroids established for the case of uniformly distributed data samples

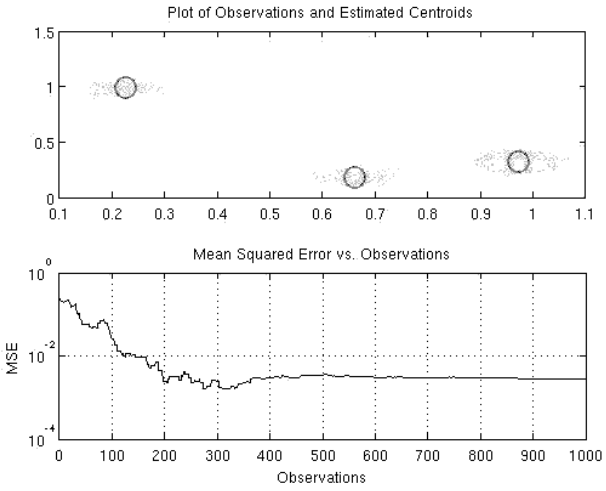

Fig. 6. Centroids established for case (2)

will receive updates at approximately the same rate. As can be observed in Figure 4, the final centroids, denoted by the large circles, accurately represent the observations. At the bottom of Figure 4, the change in the mean squared error between the estimated centroids and the actual cluster centers can be found. This shows that the stabilizing elements of the algorithm (ρ, δ_x, and PE) essentially freeze the centroid updates after about 200 observations. Figure 3 illustrates how ρ and the starvation trace change as additional observations are processed. As a result of the even distribution of observations among the clusters, the starvation traces for each centroid converge to roughly the same value. Conversely, as the centroids are updated to better represent their respective clusters, ρ decreases to a small value.

Case 2: A dominant cluster produces disproportional input samples (i.e. nonuniform distribution of data samples). In this scenario, 80% of the data samples pertained to the dominant cluster. Once the parameter ρ has determined that the centroids are representing the clusters well, it attenuates the starvation trace, keeping it from updating the centroids that have fewer observations. This is true despite the fact that not all of the starvation trace elements are converging to the same value, as can be seen in Figure 5. Without the incorporation of ρ, the starvation trace would assume the non-dominant centroids were starved and they would be inappropriately updated.

Case 3: One cluster has few observations relative to the others.

This scenario resembles the one in case (2). The parameter ρ keeps the online clustering algorithm from inferring that the cluster with few observations is starved, and the centroids are stable and estimated well. This case is illustrated in Figures 7 and 8. These figures were created with a weak cluster that only received 10% of the observations.

Case 4: Dominance of the observations alternative between clusters over time.

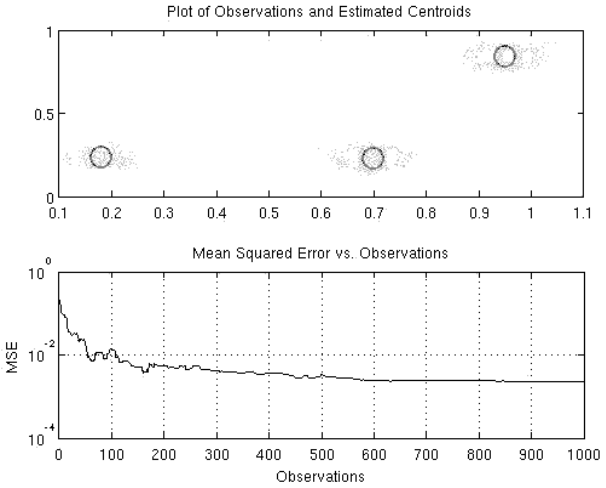

Fig. 7. Centroids for a case (3)

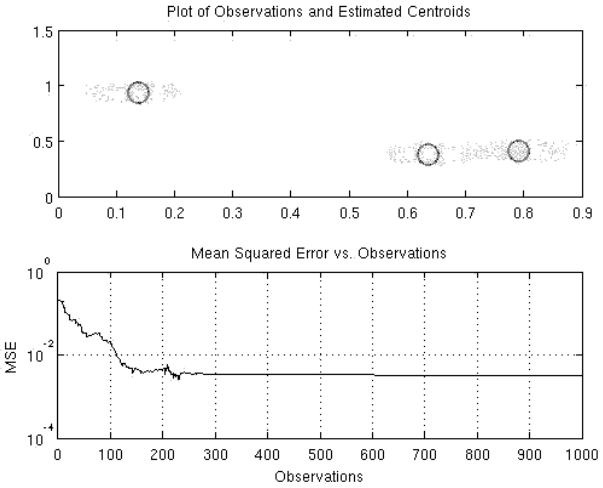

Fig. 9. Centroid formations for case (4)

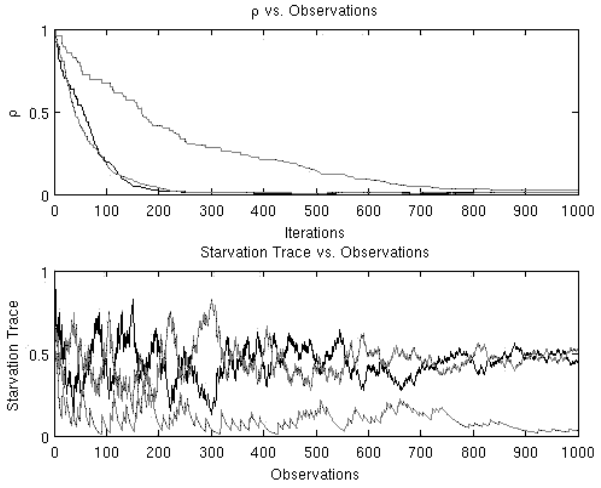

Fig. 8. ρ and starvation trace values for case (3)

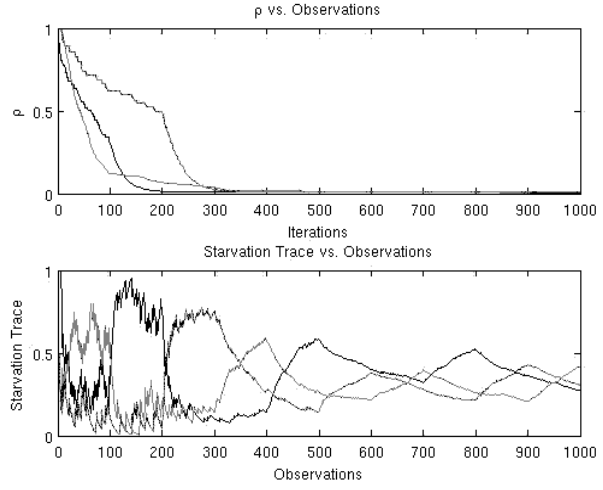

Fig. 10. ρ and starvation trace values for case (4)

This situation is handled well by the algorithm since it already handles the dominant cluster scenarios. The only other difficulty reflected by this case is the possibility of a shift in dominance, causing a centroid to infer that it is starved. As can be observed in Figures 9 and 10, this is being avoided by virtue of ρ. These figures were created by allocating 80% of the observations to a cluster and changing which cluster received this dominance every 10 iterations.

Case 5: The number of estimated centroids is greater than the actual number of clusters.

The starvation trace helps in addressing this scenario well. It gravitates the excess centroids to the clusters, and while there are still more centroids than actual clusters, all centroids are within the bounds of a cluster. This can be observed in Figures 11 and 12.

Case 6: The number of centroids is less than the actual number of clusters.

In this case, the centroids will do their best to cover the observation space as a whole. In other words, the centroids will represent multiple clusters as a single cluster. The key is that one would like centroids not to bounce around between clusters. Simulation results pertaining to this scenario are shown in Figures 13 and 14.

IV. Conclusions

This paper presented a fast and scalable incremental clustering algorithm that is modest in both computational and memory resources. In particular, starvation of prototypes and the general stability of the process are guaranteed using specific statistical metrics given the input data has a reasonably stable structure. The approach is particularly useful in applications where fast, online clustering of large databases is required. For future work, flexibility toward adding new and removing unnecessary centroids is of great concern. Generally, we feel

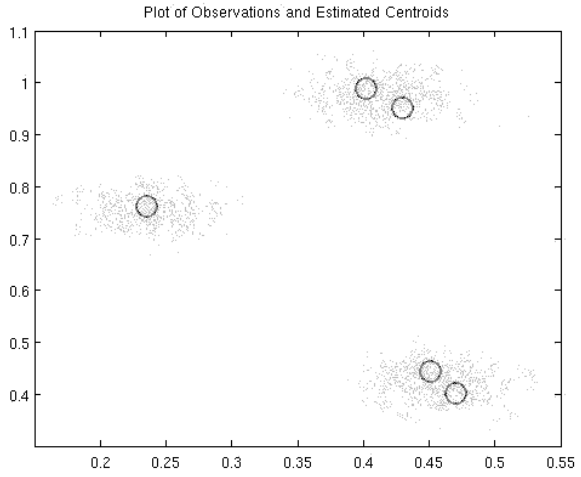

Fig. 11. Centroid formations for case (5)

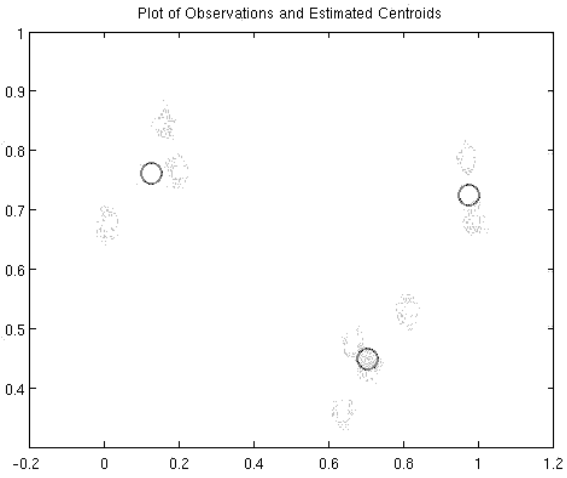

Fig. 13. Centroids formation for case (6)

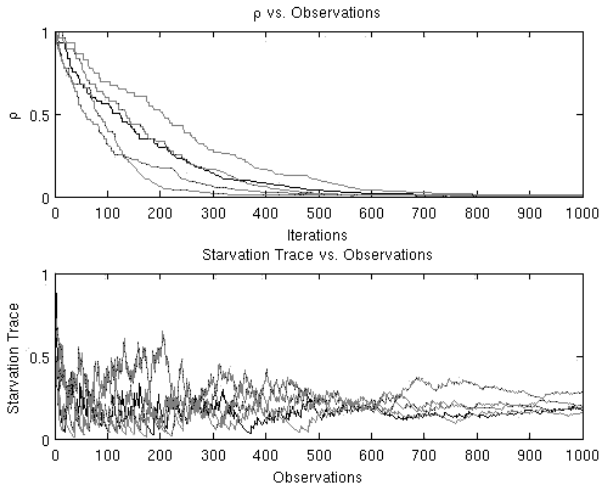

Fig. 12. ρ and starvation trace values for case (5)

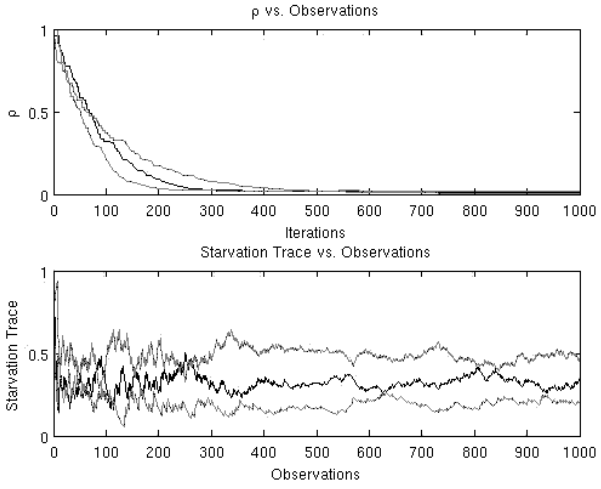

Fig. 14. ρ and starvation trace values for case (6)

the metrics presented here can be used as indicators of the discriminatory power of the centroid set. Determining time for convergence in relation to the data sample frequency is also important for cases where an estimation till stability is particularly useful. We believe this information may be encoded in the behavior of our ρ metric under real world cases and look towards ways of extracting such predictions.

REFERENCES

[1] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," 1996, pp. 103–114.

[2] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," 1994, pp. 144–155.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, October 2000.

[4] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*. John Wiley & Sons, 1997.

[5] M. Sato, Y. Sato, and L. C. Jain, *Fuzzy Clustering Models and Applications*, J. Kacprzyk, Ed. Physica-Verlag, 1997.

[6] T. Kohonen, *Self-organizing maps*, 3, Ed. Springer, 2001.

[7] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," 1999.

[8] P. Berkhin, "Survey of clustering data mining techniques," 2002.

[9] S. B. Kotsiantis and P. E. Pintelas, "Recent advances in clustering: A brief survey," *WSEAS Transactions on Information Science and Applications*, vol. 1, pp. 73–81, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.1130

[10] Cover and Thomas, *Elements of Information Theory*. Wiley-Interscience, 2006.

[11] C. Chinrungrueng and C. Sequin, "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate," *Neural Networks, IEEE Transactions on*, vol. 6, no. 1, pp. 157–169, Jan 1995.