Clustering using Renyi's Entropy

Robert Jenssen[†‡], Kenneth E. Hild II[†], Deniz Erdogmus[†], Jose C. Principe[†] and Torbjørn Eltoft[‡]

† Computational NeuroEngineering Laboratory
Department of Electrical and Computer Engineering
University of Florida
Gainesville FL. 32611, USA

‡ Electrical Engineering Group
Department of Physics
University of Tromsø
N-9037 Tromsø, Norway

***Abstract*— We propose a new clustering algorithm using Renyi's entropy as our similarity metric. The main idea is to assign a data pattern to the cluster, which among all possible clusters, increases its within-cluster entropy the least, upon inclusion of the pattern. We refer to this procedure as differential entropy clustering.**

Not knowing the true number of clusters in advance, initially a number of small clusters are "seeded" randomly in the data set, labeling a small subset of the data. Thereafter all remaining patterns are labeled by differential entropy clustering.

Subsequently, we identify the "worst cluster" by a quantity we name the between-cluster entropy. Its members are re-clustered, again by differential entropy clustering, reducing the overall number of clusters by one. This procedure is repeated until only two clusters remain.

At each step we store the current labels, thus producing a hierarchy of clusters. The between-cluster entropy also enables us to select our final set of clusters in the cluster hierarchy.

We demonstrate the clustering algorithm when applied both to artificially created data sets and a real data set.

I. Introduction

Clustering is one of the fundamental problems of pattern recognition. It aims at organizing data patterns into natural groups, or clusters, in an unsupervised manner. Important applications of clustering can be found in areas such as data mining [1], image segmentation [2], signal compression [3] and machine learning [4].

Traditionally, clustering has been divided into two commonly used approaches, the partitional and the hierarchical algorithms [5].

A partitional clustering algorithm obtains a single partition of the data, often by minimizing a cost function, e.g. the sum of squared errors between patterns and cluster centroids. The well known K-means algorithm [6] is one such method that is very popular. However, this method only works well for hyper-spherical data, or at best hyper-elliptical data, because the squared error criterion can only capture second order statistics in the data. This method also requires a-priori knowledge about the number of partitions the data should be grouped into.

This work was partially supported by NSF grants ECS-9900394 and EIA 0135946.

A hierarchical clustering algorithm on the other hand, produces a hierarchy of clusters, where different sets of clusters can be obtained at different levels in the hierarchy. Similarity between clusters can be defined in several ways [7], often resulting in different clustering results for different similarity measures.

In recent years, sophisticated clustering algorithms such as artificial neural networks [8], and support vector machines [9], have been proposed. These methods are capable of finding clusters of any shape, without a-priori knowledge about the number of clusters. These methods are often complicated, requiring fine tuning of some parameters. Various attempts have also been made to utilize information theory [10] in clustering [11]. However, information theoretic approaches often impose unrealistic parametric assumptions about the data distributions in order to evaluate the information theoretic metric [12].

In this paper we propose a simple and intuitive clustering algorithm firmly rooted in information theory, utilizing Renyi's entropy as our similarity metric. Renyi's entropy lends itself nicely to non-parametric estimation, overcoming the difficulty in evaluating traditional entropy metrics. Using entropy as our metric, we are able to utilize all the information contained in the distribution of the data, and not only mere second order statistics as many traditional clustering algorithms are limited to.

In our approach, we assign a new pattern to the cluster, which among all possible clusters, *increases its within-cluster entropy the least*, upon inclusion of the pattern. To evaluate groupings we define a quantity which we name the *between-cluster entropy*, also based on Renyi's entropy. This quantity was first introduced by Gokcay and Principe [12], there referred to as the cluster evaluation function (CEF). Our method is capable of finding clusters of any shape, without knowing the true number of clusters in advance.

In the next section the differential entropy clustering is explained, and a method for non-parametrically estimating the within-cluster entropy directly from data is introduced. In section III the between-cluster entropy is reviewed. Section IV presents some performance studies, both on artificial data

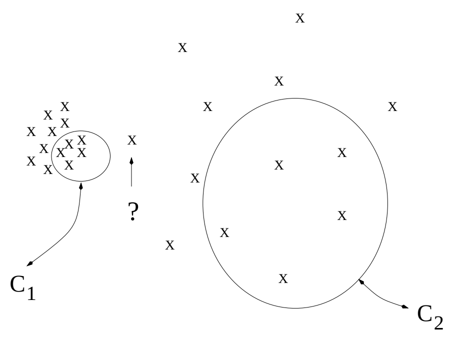


Fig. 1. Assigning a data pattern to a cluster.

sets and a real data set. In section V we make our concluding remarks.

II. Differential Entropy clustering

Consider the situation depicted in Fig. 1. A set of patterns, or feature vectors, is distributed in feature space. Initially a subset of the feature vectors have been assigned to cluster C_1 or C_2. These are shown as the encircled points. The problem of clustering is now to decide whether a new pattern $\mathbf{x}$ (pointed to by the question mark) should be assigned to C_1 or C_2.

We propose to cluster $\mathbf{x}$ based on a simple observation. If $\mathbf{x}$ is wrongly assigned to C_1, the uncertainty, or entropy, of C_1 will increase more than the entropy of C_2 will, if $\mathbf{x}$ is assigned correctly to C_2.

Hence, in the general case of having initial clusters C_k, $k = 1, \ldots, K$, assign $\mathbf{x}$ to cluster C_i if

$$H(C_i + \mathbf{x}) - H(C_i) < H(C_k + \mathbf{x}) - H(C_k), \qquad (1)$$

for $k = 1, \ldots, K$, $k \neq i$, where $H(C_k)$ denotes the entropy of cluster C_k. We refer to this method as differential entropy clustering.

At this point, three questions arise. 1) How to estimate entropy directly from data? 2) How to initially cluster a subset of the data? 3) How to decide which pattern $\mathbf{x}$ to be clustered next?

A. Within-cluster entropy

Previously, entropy has been a metric difficult to evaluate without imposing unrealistic assumptions about the data distributions [12]. Recently it was discovered that an entropy measure proposed by Renyi [13] lends itself nicely to non-parametric estimation directly from data [14].

Renyi's entropy for a stochastic variable $\mathbf{X}$ with probability density function (pdf) $f_{\mathbf{X}}$ is given by [13]

$$H_R(\mathbf{X}) = \frac{1}{1-\alpha} \log \int f_{\mathbf{X}}^{\alpha} \, d\mathbf{x}, \; \alpha > 0, \; \alpha \neq 1. \qquad (2)$$

Specifically, for $\alpha = 2$ we obtain [14]

$$H_R(\mathbf{X}) = -\log \int f_{\mathbf{X}}^2 \, d\mathbf{x}, \qquad (3)$$

which is called Renyi's quadratic entropy [14].

This expression can easily be estimated directly from data by the use of Parzen window density estimation, with a multidimensional Gaussian window function. Assume that cluster C_k consists of the set of discrete data points $\mathbf{x}_i$, $i = 1, \ldots, N_k$. Now, the pdf estimate based on the data points of C_k is given by [15]

$$\hat{f}_{\mathbf{X}} = \frac{1}{N_k} \sum_{i=1}^{N_k} G(\mathbf{x} - \mathbf{x}_i, \sigma^2 \mathbf{I}), \qquad (4)$$

where N_k is the number of data points in C_k, and we have used a symmetric Gaussian kernel with covariance matrix $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$. By substituting (4) into (3), and utilizing the properties of the Gaussian kernel, we obtain an estimate of the entropy of C_k as

$$H(C_k) = -\log V(C_k), \qquad (5)$$

where

$$V(C_k) = \frac{1}{N_k^2} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} G(\mathbf{x}_i - \mathbf{x}_j, 2\sigma^2 \mathbf{I}). \qquad (6)$$

Since the entropy is calculated based on points assigned to the same cluster, we refer to (5) as the within-cluster entropy.

In this paper, all $G(\mathbf{x}_i - \mathbf{x}_j, 2\sigma^2\mathbf{I})$, $i, j = 1, \ldots, N$, where N is the total number of data patterns in the data set, are calculated and stored in an N^2 symmetric proximity matrix. This means that after the proximity matrix has been obtained, all calculations are simple matrix manipulations. Obviously this limits somewhat the size of the data set we can cluster.

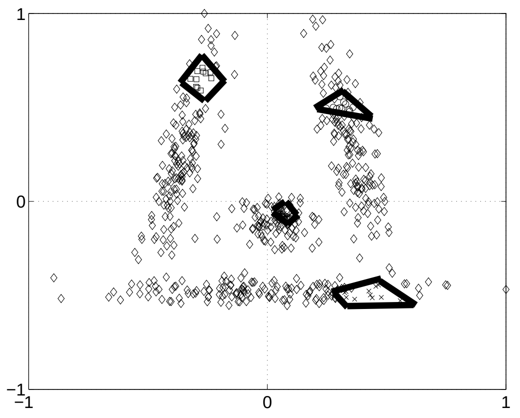
Another issue regards the kernel size σ. Our experiments have shown that promising clustering results are obtained provided that σ is chosen such that the Parzen pdf estimate is relatively accurate. Choosing σ thus belongs to the more general problem of non-parametric pdf estimation. In this paper, we choose σ manually. It should be noted that our algorithm has resemblance to other kernel based clustering methods, such as spectral clustering [16] and Mercer kernel based clustering [17]. In all such methods the kernel size is a parameter of great importance, which has to be chosen based on some heuristic. Note that for all data sets used in the experiments we normalize each feature individually to have a range $[-1, 1]$ and zero mean.

B. Cluster initialization

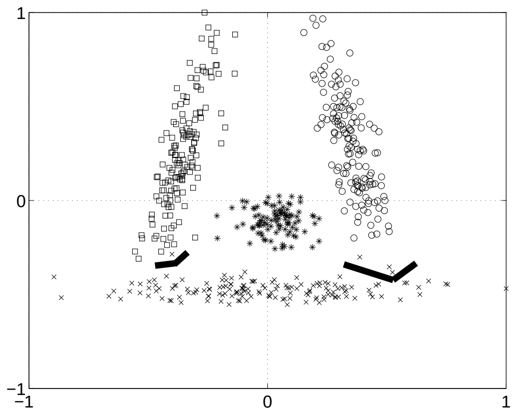
Initially we "seed" K_{init} clusters in the data set. This is done by first randomly selecting K_{init} "seed" patterns from the data set, each initially representing a cluster. Thereafter the point closest to any member of a cluster is included in the cluster, until a preselected value of N_{init} patterns have been assigned to each cluster. The grouping is done this way instead of just finding the N_{init} nearest neighbors to the each "seed" pattern, since this makes the grouping more sensitive to the data structure [12].

C. Selecting the next pattern for clustering

The next pattern to be clustered can be selected in several ways.



(a)



(b)

Fig. 2. Example of differential entropy clustering.

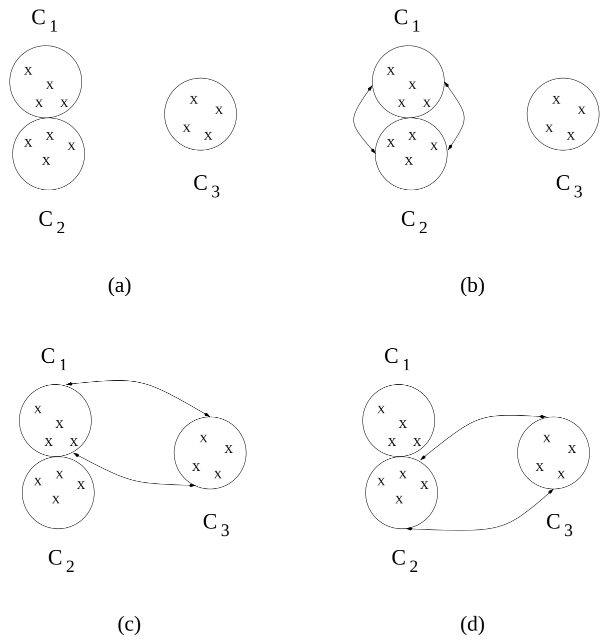


Fig. 3. Identifying "worst cluster" using between-cluster entropy. In this example C_1 or C_2 is the "worst cluster."

- Randomly - this has the disadvantage that early clustering of points far from the initial clusters can make the clustering process un-stable.
- As the pattern closest to a cluster prototype - this approach makes the clustering more stable. An example of a cluster prototype can e.g. be the cluster mean.

Figure 2 shows an example of clustering with the method described above. In this example we select the next pattern to be clustered as the pattern closest to one of the cluster means. The regions where the initial clusters are "seeded" are shown in (a) enclosed by the thick lines. The initial cluster sizes were $N_{\text{init}} = 10$. In (b) the resulting set of clusters is shown for $\sigma = 0.08$. Each cluster is marked by a different symbol. In this case four errors are made. The thick lines in (b) indicate where the erroneously labeled patterns actually belong. It should be noted that K-means fails completely on this kind of data, as we will show in section IV. The data set used here is taken from [8].

Obviously, this example is not very realistic. We do not know the true number of clusters in advance, and even if we do, we can not be sure to "seed" a pattern in each of the true clusters.

The solution we propose to this problem is to "seed" a large number of clusters, label all the points, and then to re-assign labels to the members of the "worst cluster". Re-labeling the members of the "worst cluster" is again done by differential entropy clustering. The procedure is repeated, reducing the number of clusters by one at each step. Hence, to find the "worst cluster" we need a means for cluster evaluation, and this is the topic for the next section.

III. Entropy Cluster Evaluation

Gokcay and Principe [12] proposed an information theoretic clustering algorithm based on the *cluster evaluation function.* Their algorithm seeks valleys in the data by minimizing the CEF.

Here we propose to use the CEF in a different manner. We do not intend to minimize it, only to compute it, in order to find the "worst cluster" at each step. Toward this end we use the CEF to define a quantity we name the between-cluster entropy.

A. Between-cluster entropy

Instead of using (6) to compute the entropy of each cluster individually, we modify it such that the double sum runs over all data points, and includes a membership function, $M(\mathbf{x}_{ij})$ which equals one if $\mathbf{x}_i$ and $\mathbf{x}_j$ belongs to different clusters, and zero if not. We name the resulting expression the between-cluster entropy, which is given by

$$H(C_1, \ldots, C_K) = -\log V(C_1, \ldots, C_K), \qquad (7)$$

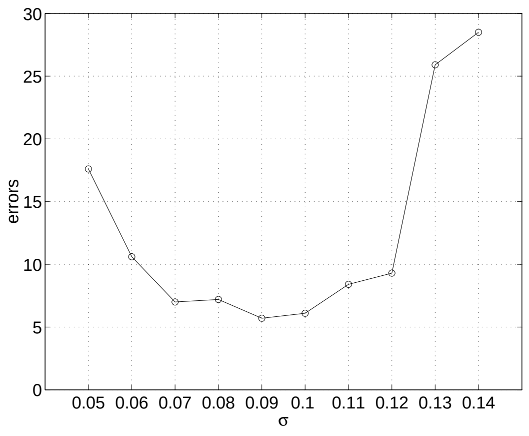


Fig. 4. Average amount of errors for different σ.

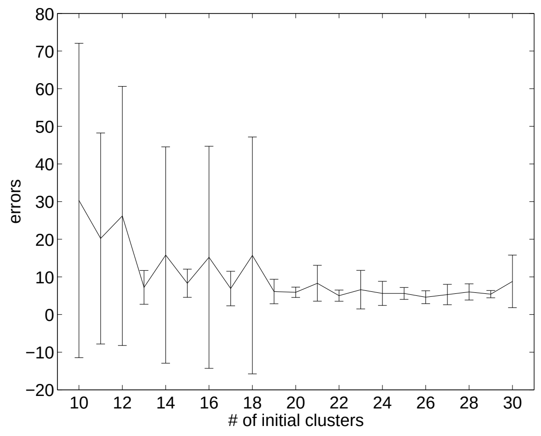


Fig. 5. Mean error for different K_{init}.

where

$$V(C_1,\ldots,C_K) = \ldots$$
$$= \frac{1}{2\prod_{k=1}^{K} N_k}\sum_{i=1}^{N}\sum_{j=1}^{N} M(\mathbf{x}_{ij})G(\mathbf{x}_i-\mathbf{x}_j,2\sigma^2\mathbf{I}), \quad (8)$$

for clusters C_k, $k=1,\ldots,K$. Equation (8) equals the CEF introduced in [12].

If the clusters are well separated, $V(C_1,\ldots,C_K)$ will have a small value, consequently $H(C_1,\ldots,C_K)$ will have a large value. This provides us with a tool for cluster evaluation.

B. Identifying the "worst cluster"

In Fig. 3 we have illustrated how the between-cluster entropy can be used to identify the "worst cluster". In (a) a data set has been partitioned into three clusters. We proceed by eliminating one cluster at a time, and calculate the between-cluster entropy based on the remaining clusters in each case. To be more specific, by eliminating, we mean that the membership function M is set to zero whenever acting on any of the members of the eliminated cluster. For example, in (b) C_3 is eliminated, and $H(C_1,C_2)$ is calculated, as indicated by the arrows. Likewise in (c) and (d).

The "worst cluster" is now selected as the cluster that when eliminated, results in the largest between-cluster entropy based on the remaining clusters, because this means that the remaining clusters are the most separated clusters. In the situation depicted in Fig. 3, this method results in either C_1 or C_2 being identified as the "worst cluster".

C. Selecting the final set of clusters

Note that before each time the members of a cluster are re-assigned labels, and the number of clusters is reduced by one, all patterns have been labeled. Thus at each step a set of clusters exist. We store the cluster labels at each step, thus producing a hierarchy of cluster assignments. We continue this procedure until only two clusters remain. The issue now is to decide where in the hierarchy to select our final clustering.

At each step we calculate the overall between-cluster entropy based on all clusters defined at that step, as given by (7). Our experiments have shown that when we continue re-clustering, the between-cluster entropy increases drastically when the number of clusters is reduced to a value less than the true number of clusters.

Thus, by monitoring the difference in between-cluster entropy calculated before and after re-clustering at each step, we are able to select our final set of clusters in the hierarchy.

D. Instability due to random initialization

Because the initial clusters are "seeded" randomly, the clustering result can differ when clustering the same data set several times. This is most prevalent for data sets consisting of highly irregular clusters. The between-cluster entropy calculated for the final clustering in each case can provide us with a good indication of which random initialization provided the best result.

IV. Performance studies

In this section we test the performance of our algorithm on three data sets, two artificially created and one real.

First we re-visit the data set shown in Fig. 2. In this experiment, we show the influence of σ on the clustering results. We "seed" $K_{\text{init}} = 20$ initial clusters each with $N_{\text{init}} = 10$ members. The next pattern to be clustered is the one closest to some already labeled pattern. For each σ the algorithm is run 10 times. Figure 4 shows the average number of resulting errors for a range of σ's. We see that for $0.06 \leq \sigma \leq 0.12$ the algorithm performs very well, producing on average between 6 and 10 errors. It should be noted that we obtain almost as good results when choosing the next pattern to be clustered as the one closest to one of the cluster means.

Figure 5 shows the the clustering performance as a function of K_{init} over 10 runs using $\sigma = 0.095$. The error bars indicate the standard deviation. Large error bars for $K_{\text{init}} \leq 18$ are typically produced because of one complete failure out of the 10 runs. For $K_{\text{init}} \geq 19$, the algorithm proves to be always stable.

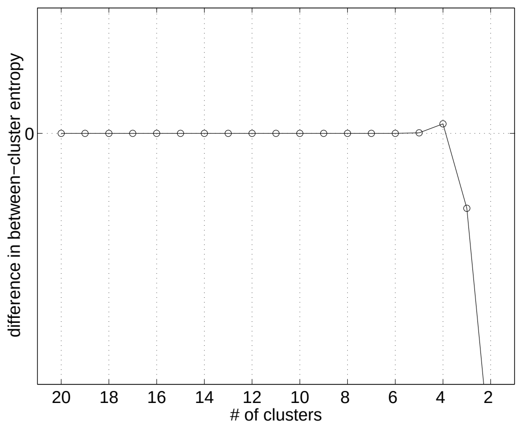


Fig. 6. Determining true number of clusters.

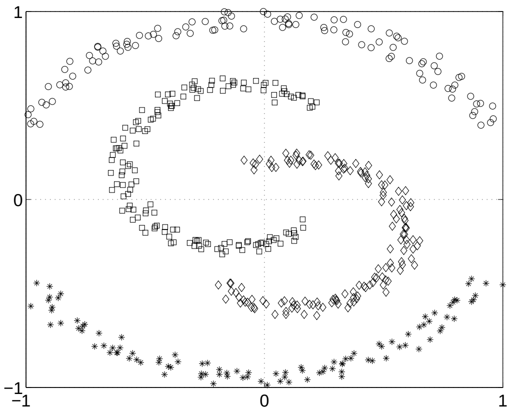


Fig. 8. Result for highly irregular clusters, $\sigma = 0.09$.

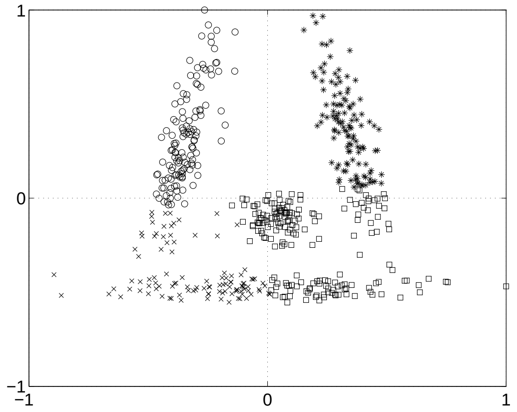


Fig. 7. K-means result for data set shown in Fig. 2.

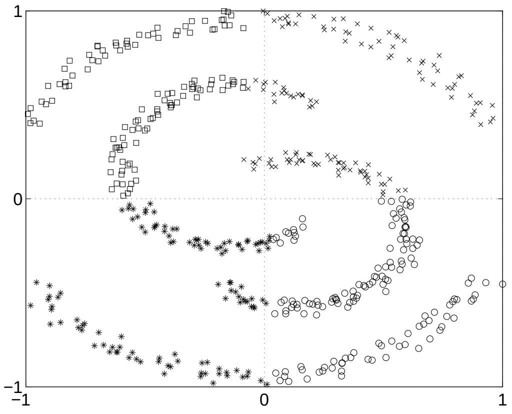


Fig. 9. K-means result for highly irregular clusters.

Figure 6 shows a plot of the between-cluster entropy at the current step subtracted the between-cluster entropy at the previous step. The true number of clusters is four. It can be seen that when we go from four clusters to three, the between-cluster entropy becomes large compared to at the previous step, creating a distinct decrease in the plot. We select our final clustering result as the set of clusters that exist before the first significant decrease in the plot.

For comparison, we show in Fig. 7 the result of clustering the same data set with the popular K-means algorithm. We show the best result out of 10 runs. It is clear that K-means is un able to cope with this kind of data set.

Figure 8 shows the clustering result for some highly irregular clusters. For instance, for a kernel size $\sigma = 0.09$ a perfect clustering is obtained in nine out of 10 trials. However, the range of σ's for which we obtain satisfying clustering results is more narrow than in the previous experiment.

As shown in Fig. 9 K-means fails completely also in this case. This is to be expected since the clusters are highly irregular.

Finally, we test our method on the WINE data set, extracted from the UCI repository database [18]. This data set consists of 178 instances in a 13-dimensional feature space, where the features are found by chemical analysis of three different types of wines. We include this data set in our analysis because it shows that our algorithm is capable of performing well in a high dimensional feature space.

For $0.25 \leq \sigma \leq 0.3$ we obtain clustering results with only a few failures. E.g. for $\sigma = 0.26$, we obtain a mean error of 7.6 over 10 runs, when we select the next pattern to be clustered as the one closest to some labeled pattern. When selecting the next pattern as the one closest to one of the cluster means, the mean error was 9.2.

V. Conclusion

We have presented a new clustering algorithm using a non-parametric estimate of Renyi's entropy as our similarity metric. A data pattern is assigned to the cluster increasing it's within-cluster entropy the least among all clusters, upon inclusion of the pattern. Cluster groupings are evaluated by means of the between-cluster entropy.

We have shown that our algorithm performs well on data sets of irregular and non-spherical shape. We attribute this property to the entropy metric, which captures information contained in the data distribution beyond second order statistics.

We have also shown that the algorithm is capable of clustering a high-dimensional data set.

One drawback of our algorithm lies in the computational complexity involved in calculating the between-cluster entropy. It requires an $O(N^2)$ operation, where N is the total number of patterns in the data set. For large data sets, where we do not have the ability to store a proximity matrix in the memory, this becomes a problem. The within-cluster entropy can be calculated recursively when including a new pattern in a cluster, thus it requires an $O(N_k)$ operation for each cluster, $C_k, \ k = 1, \ldots, K$.

In addition, at present, we have not implemented any automatic procedure to determine the kernel size σ, such that we ensure that the inherent Parzen pdf estimate is relatively accurate. Another possibility is to anneal the kernel size during the clustering process, from a large value to a small value. This could perhaps reduce the number of initial clusters needed, by providing more robustness in the differential entropy stage in the first phases of the clustering algorithm, and provide a fine-tuning in the final phases of the algorithm. One could also associate a unique kernel for every data pattern. Each kernel could be adapted based on the neighboring patterns. A combination of the aforementioned strategies could also be implemented. These suggestions are topics for further research.

References

[1] D. Judd, P. McKinley, and A. K. Jain, “Large-Scale Parallel Data Clustering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 871–876, 1998.

[2] H. Frigui and R. Krishuapuram, “A Robust Competitive Clustering Algorithm with Applications in Computer Vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 450–465, 1999.

[3] H. M. Abbas and M. M. Fahmy, “Neural Networks for Maximum Likelihood Clustering,” *Signal Processing*, vol. 36, no. 1, pp. 111–126, 1994.

[4] C. Carpineto and G. Romano, “A Lattice Conceptual Clustering System and its Application to Browsing Retrieval,” *Machine Learning*, vol. 24, no. 2, pp. 96–122, 1996.

[5] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[6] J. McQueen, “Some methods for classification and analysis of multivariate observations,” in *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[7] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data Clustering: A Review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[8] T. Eltoft and R. J. P. deFigueiredo, “A New Neural Network for Cluster-Detection-and-Labeling,” *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1021–1035, 1998.

[9] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, “Support Vector Clustering,” *Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2001.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & sons, 1991.

[11] S. J. Roberts, R. Everson, and I. Rezek, “Minimum Entropy Data Partitioning,” in *Ninth International Conference on Artificial Neural Networks*, 1999, vol. 2, pp. 844–849.

[12] E. Gokcay and J. Principe, “Information Theoretic Clustering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 158–170, 2002.

[13] A. Renyi, “On Measures of Entropy and Information,” in *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1960, pp. 547–561.

[14] J. Principe, D. Xu, and J. Fisher, *Unsupervised Adaptive Filtering*, vol. 1, chapter 7 ”Information Theoretic Learning”, John Wiley & Sons, 2000.

[15] E. Parzen, “On the Estimation of a probability density function and the mode,” *Ann. Math. Stat.*, vol. 32, pp. 1065–1076, 1962.

[16] A. Y. Ng, M. Jordan, and Y. Weiss, “On Spectral Clustering: Analysis and an Algorithm,” in *Advances in Neural Information Processing Systems*, 2002, number 14, pp. 849–856.

[17] M. Girolami, “Mercer Kernel-Based Clustering in Feature Space,” *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 780–784, 2002.

[18] R. Murphy and D. Ada, “UCI Repository of Machine Learning databases,” Tech. Rep., Dept. Comput. Sci. Univ. California, Irvine, 1994.