Competitive Fuzzy Clustering

Hichem Frigui and Raghu Krishnapuram
Department of Computer Engineering and Computer Science
University of Missouri
Columbia, MO 65211
frigui or raghu@ece.missouri.edu

Abstract

In this paper, we introduce a new approach called Competitive Agglomeration (CA), which combines the advantages of hierarchical and partitional clustering techniques. The CA algorithm starts by partitioning the data set into a large number of small clusters. As the algorithm progresses, adjacent clusters compete for data points, and clusters that lose in the competition gradually become depleted and vanish. Thus, as the iterations proceed, we obtain a sequence of partitions with a progressively diminishing number of clusters. The final partition is taken to have the "optimal" number of clusters from the point of view of the objective function. Since the algorithm starts with an overspecified number of clusters, the final results are quite insensitive to the initialization and to local minima. In addition, we can incorporate different distance measures in the objective function of the CA algorithm to find an unknown number of clusters of various shapes. We illustrate the performance of the CA algorithm for the special cases of spherical, ellipsoidal, linear, and shell clusters.

1 Competitive Clustering

Let $\mathcal{X} = \{\mathbf{x}_j \mid j = 1, \ldots, N\}$ be a set of N vectors and let $\mathbf{B} = (\beta_1, \ldots, \beta_c)$ represent a C-tuple of prototypes each of which characterizes one of the C clusters. The CA algorithm minimizes

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d^2(\mathbf{x}_j, \beta_i) \; - \; \alpha \sum_{i=1}^{C} \left[\sum_{j=1}^{N} u_{ij} \right]^2 \quad (1)$$

Subject to

$$\sum_{i=1}^{C} u_{ij} = 1, \qquad for\ j \in \{1, \cdots, N\}. \quad (2)$$

In (1), $d^2(\mathbf{x}_j, \beta_i)$ represents the distance from feature vector $\mathbf{x}_j$ to the prototype β_i, u_{ij} represents the degree of membership of feature point $\mathbf{x}_j$ in cluster β_i, and $U = [u_{ij}]$ is a $C \times N$ constrained fuzzy C-partition matrix [1]. It should be noted that the number of clusters C in (1) is dynamically updated in the CA algorithm. The first component in (1) allows us to control the shapes and sizes of the clusters and to obtain compact clusters. The global minimum of this component is achieved when the number of clusters C is equal to the number of samples N. The second component in (1) is the sum of squares of the cardinalities of the clusters which allows us to control the number of clusters. The global minimum of this term (including the negative sign) is achieved when all points are lumped into one cluster, and all other clusters are empty. When both components are combined and α is chosen properly, the final partition will minimize the sum of intra-cluster distances, while partitioning the data set into the smallest possible number of clusters. The clusters which are depleted as the algorithm proceeds will be discarded, as explained later.

To minimize (1) with respect to $\mathbf{U}$, we apply Lagrange multipliers and obtain

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d^2(\mathbf{x}_j, \beta_i) \; - \; \alpha \sum_{i=1}^{C} \left[\sum_{j=1}^{N} u_{ij} \right]^2 - \sum_{j=1}^{N} \lambda_j \Big(\sum_{i=1}^{C} u_{ij} - 1 \Big). \quad (3)$$

We then fix $\mathbf{B}$ and solve

$$\frac{\partial J}{\partial u_{st}} = 2 u_{st} d^2(\mathbf{x}_t, \beta_s) \; - \; 2\alpha \sum_{j=1}^{N} u_{sj} - \lambda_t = 0, \quad (4)$$

to obtain an updating equation for the memberships u_{st}. Equation (4) is a set of $N \times C$ linear equations with $N \times C + N$ variables (u_{st}, and λ_t). Using them in conjunction with the N equations resulting from the constraint in (2), one can solve for the $N \times C + N$ variables. However, the solution can be simplified considerably by assuming that the membership values do not change significantly from one iteration to the next, and by computing the term $\sum_{j=1}^{N} u_{sj}$ in (4) using the membership values from the previous iteration. With this assumption, (4) reduces to

$$u_{st} = \frac{2\alpha \times N_s + \lambda_t}{2 d^2(\mathbf{x}_t, \beta_s)}, \quad (5)$$

where

$$N_s = \sum_{j=1}^{N} u_{sj}. \tag{6}$$

is the cardinality of cluster s. Using (5) and the constraint in (2), we obtain

$$u_{st} = u_{st}^{\mathrm{FCM}} + u_{st}^{\mathrm{Bias}}, \tag{7}$$

where

$$u_{st}^{\mathrm{FCM}} = \frac{\frac{1}{d^2(\mathbf{x}_t, \beta_s)}}{\sum_{k=1}^{C} \frac{1}{d^2(\mathbf{x}_t, \beta_k)}}, \tag{8}$$

and

$$u_{st}^{\mathrm{Bias}} = \frac{\alpha}{d^2(\mathbf{x}_t, \beta_s)}\left(N_s - \overline{N}_t\right). \tag{9}$$

In (9), $\overline{N}_t$ is defined as

$$\overline{N}_t = \frac{\sum_{k=1}^{C} \frac{1}{d^2(\mathbf{x}_t, \beta_k)} N_k}{\sum_{k=1}^{C} \frac{1}{d^2(\mathbf{x}_t, \beta_k)}},$$

which is simply a weighted average of the cluster cardinalities, where the weight of each cluster reflects its proximity to the feature point $\mathbf{x}_t$ in question.

As seen from (7), u_{st}^{FCM} in (8), is the membership term in the FCM algorithm [1] which takes into account only the relative distances of the feature point to all clusters. The second component in (7), u_{st}^{Bias}, is a signed bias term which depends on the difference between the cardinality of the cluster of interest, and the weighted average of cardinalities from the point of view of feature point $\mathbf{x}_t$. For clusters with cardinality higher than average, the bias term is positive, thus appreciating the membership value. On the other hand, for low cardinality clusters, the bias term is negative, thus depreciating the membership value. This leads to a gradual erosion of the cardinality of spurious clusters. When the cardinality of a cluster drops below a threshold, we discard the cluster, and update the number of clusters. Since the initial partition has an overspecified number of clusters, each cluster is approximated by many small clusters in the beginning. As the algorithm proceeds, the second term in (1) causes each cluster to expand and include as many points as possible. At the same time, the constraint in (2) causes adjacent clusters to compete. As a result, only a few clusters will survive, while others will shrink and eventually become extinct.

It should be noted that when a feature point $\mathbf{x}_j$ is close to only one cluster (say cluster i), and far from other clusters, we have

$$N_i \approx \overline{N}_j, \qquad \text{or} \qquad u_{ij}^{\mathrm{Bias}} \approx 0.$$

In this case the membership value, u_{ij}, is independent of the cluster cardinalities, and reduces to u_{ij}^{FCM}. In other words, if a point is close to only one cluster, it will have high membership value in this cluster and no competition is involved. On the other hand, if a point is close to many clusters, these clusters will compete for this point based on cardinality. This encourages the formation of larger clusters, i.e., it promotes agglomeration.

The choice of α in (1) is important in the CA algorithm since it reflects the importance of the second term relative to the first term. To make the algorithm independent of the distance measure, α should be proportional to the ratio of the two terms. That is

$$\alpha \propto \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d^2(\mathbf{x}_j, \beta_i)}{\sum_{i=1}^{C} \left[\sum_{j=1}^{N} u_{ij}\right]^2}.$$

In all examples described in this paper, we choose α to be

$$\alpha(k) = \eta(k) \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d^2(\mathbf{x}_j, \beta_i)}{\sum_{i=1}^{C} \left[\sum_{j=1}^{N} u_{ij}\right]^2}. \tag{10}$$

In (10), α and η are functions of the iteration number k. A good choice for η is the exponential decay defined by

$$\eta(k) = \eta_0 e^{-k/\tau},$$

where η_0 is the initial value and τ is the time constant.

It should be noted that depending on the value of α, the membership values u_{ij} may not be confined to [0, 1]. In fact, u_{ij} can become negative if N_i is very small and point $\mathbf{x}_j$ is close to other dense clusters (i.e. $\overline{N}_j$ is high). This generally implies that cluster β_i is spurious. In this case, it is safe to set u_{ij} to zero to indicate that feature vector $\mathbf{x}_j$ is atypical of cluster β_i. It is also possible for u_{ij} to become larger than 1 if N_i is very large and feature point $\mathbf{x}_j$ is close to other spurious clusters (i.e. $\overline{N}_j$ is low). In this case it is safe to clip u_{ij} to 1 to indicate that feature vector $\mathbf{x}_j$ is typical of cluster β_i.

Minimization of (1) with respect to the prototypes to derive the update equation for $\mathbf{B} = (\beta_1, \ldots, \beta_c)$, varies according to the choice of the prototypes and the distance measure. Each choice leads to a different algorithm. Since the second term in (1) does not depend on the prototypes and the distance measure, the parameter update equations in the CA algorithm are the same as those in traditional objective-function-based fuzzy clustering algorithms that minimize the sum of squared distances [2].

2 Experimental results

In this section we illustrate the effectiveness of the proposed unsupervised approach for the special cases of spherical, ellipsoidal, linear, and shell clusters. In all examples shown in this section, a cluster β_i was discarded if its cardinality (N_i) is less than 5. The initial value of η (η_0) was set to 5 and the time constant τ was set to 10. The initial partition is obtained by running the FCM algorithm [1] (with the Euclidean distance measure and m=2) for 5 iterations. The FCM algorithm is then applied with the appropriate distance measure for 5 iterations. This allows the clusters to be approximated by many prototypes at the beginning. In our experience, the CA algorithm is independent of the initial number of clusters C_{max} as long as C_{max} is chosen to be much larger than the expected number of clusters. In all examples, we tried many different values for C_{max} ranging from 15 to 30, and in all cases, the algorithm converged to the same final partition. In this paper we display the intermediate partitions of the algorithm when C_{max} is chosen to be 20.

In Fig. 1 we illustrate the performance of the CA algorithm when the Euclidean distance is used. Fig. 1(a) is a synthetic data set that consists of four clusters of various sizes and densities. Fig. 1(b) shows the prototype parameters of the initial partition. The prototypes (centers) locations are shown as "+" symbols superimposed on the data set. As can be seen in this figure, large clusters are split into many clusters and small clusters are split into fewer number of clusters. Fig. 1(c) shows the intermediate results after 3 iterations where the number of clusters is reduced to 13. Fig. 1(d) shows the final results of the CA algorithm (after 15 iterations).

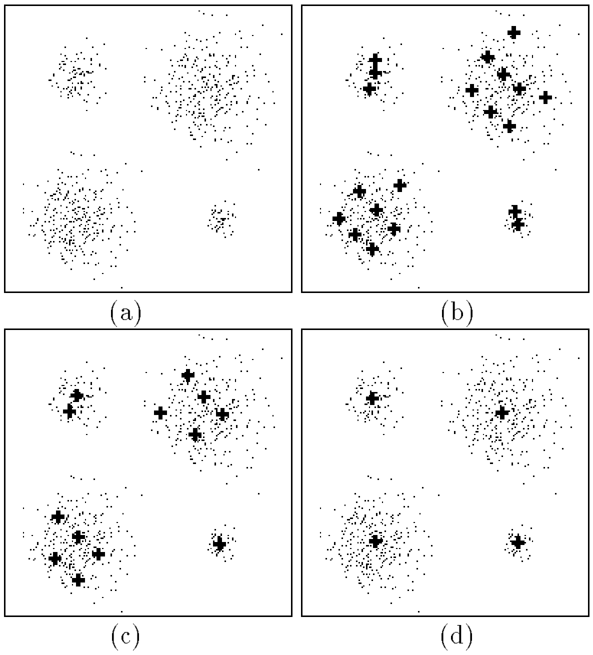
In Fig. 2 we illustrate the performance of the CA algorithm when the Gustafson-Kessel (G-K) distance [3] is used to find ellipsoidal clusters. Fig. 2(a) shows a data set containing six Gaussian clusters of various sizes, shapes, and orientations. Fig. 2(b) shows the initial prototypes. As before, The "+" signs indicate the cluster centers, and the ellipses shown in the figure enclose points having a Mahalanobis distance less than 9. Fig. 2(c) shows the prototypes obtained after 4 iterations of the CA algorithm where the number of clusters is reduced to 10. Fig. 2(d) shows the final results after convergence of the CA algorithm (after 12 iterations).

In Fig. 3 we illustrate the performance of the CA algorithm when the Adaptive Fuzzy Clustering (AFC) distance [4] is used to detect linear clusters. Fig. 3(a) shows a data set that consists of 10 line segments. The initial prototypes are displayed in Fig. 3(b). Fig. 3(c) shows the results after 2 iterations where The number of clusters is reduced to 15. The CA algorithm converged after 10 iterations, and the final results are shown in Fig. 3(d).

In Fig. 4 we illustrate the performance of the CA algorithm when the Fuzzy Quadric Shell (FCQS) distance [5] is used to detect shell clusters. Fig. 4(a) shows a synthetic image consisting of three overlapping ellipses of various sizes and orientations and a circle. Fig. 4(b) shows the initial prototypes. Fig. 4(c) shows the results after running 5 iterations of the CA algorithm. As can be seen the number of clusters drops to 9 after 11 empty clusters are discarded. Fig. 4(d) shows the final results of the CA algorithm (after 16 iterations).

3 Conclusion

In this paper, we have presented a new unsupervised and computationally attractive clustering algorithm that automatically determines the "optimum" number of clusters. Our algorithm minimizes a fuzzy prototype-based objective function, and produces a sequence of partitions with a decreasing number of clusters. The initial partition has an overspecified number of clusters, and the final one has the optimal number of clusters. Constrained fuzzy memberships are used to force the clusters to compete for data points and reduce the number of clusters from one iteration to the next. Our algorithm uses a membership update equation that takes into account relative cluster cardinalities, as well as the relative distances to the cluster prototypes. To elaborate, the CA membership consists of two components. The first component is the traditional FCM membership which takes into account only relative proximities between points and cluster prototypes. The second component is a signed bias term. This bias is negligible for a feature point that is close to just one cluster. This is the case of no competition. On the other hand, if a point is close to multiple clusters, then the bias will be positive for the clusters with larger cardinalities, and negative for sparse ones. This bias term creates a competitive environment that promotes the survival of the largest clusters, and the extinction of spurious ones. Since the CA algorithm is formulated independently of the distance measure, it can be used to find the optimal number of clusters for the special cases of spherical, ellipsoidal, linear, and shell clusters. The proposed algorithm is also relatively insensitive to initialization and local minima effects when compared with traditional objective-function-based clustering algorithms. This is due to its agglomerative property, i.e., each cluster is initially approximated by many prototypes.



(a) (b) (c) (d)

Figure 1: Intermediate results of the CA algorithm on a data set with spherical clusters
(a) original image, (b) initial prototypes,
(c) results after 3 iterations, and (d) final results.

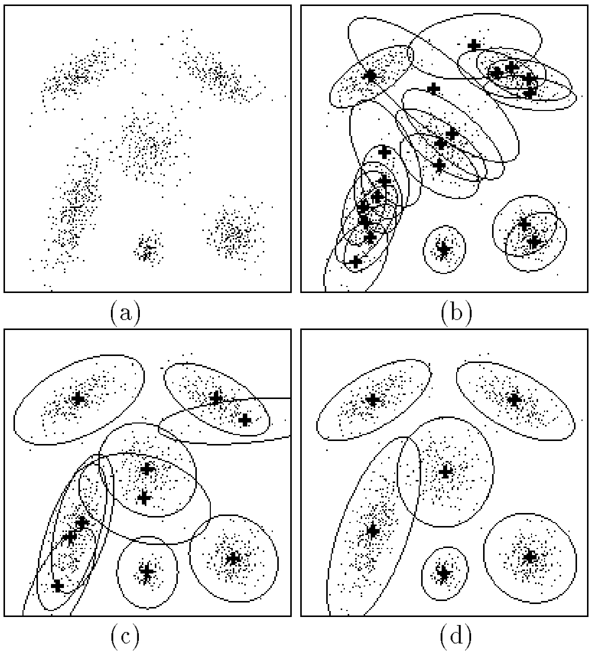


Figure 2: Intermediate results of the CA algorithm on a data set with ellipsoidal clusters
(a) original image, (b) initial prototypes,
(c) results after 4 iterations, and (d) final results.

In all the examples shown, the CA algorithm converges in less than 20 iterations. This makes our algorithm computationally attractive even when the number of clusters is unknown.

Acknowledgments

This work was partially supported by a grant from the Office of Naval Research (N00014-96-1-0439)

References

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

[2] R. Krishnapuram and J. Keller", "Fuzzy and Possibilistic Clustering Methods for Computer Vision", in *Neural and Fuzzy Systems*, S. Mitra and M. Gupta and W. Kraske (Ed.), SPIE Institute Series, pp. 133-159, 1994.

[3] E. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," *Proc. IEEE CDC*, San Diego, Calif., pp 761-766, 1979.

[4] R. N. Davé, "Use of the adaptive fuzzy clustering algorithm to detect lines in digital images," *Intelligent Robots and Computer Vision VIII*, vol. 1192, pp. 600-611, 1989.

[5] R. Krishnapuram, H. Frigui, and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation-Parts I and II," *IEEE Trans. Fuzzy Syst.*, Vol. 3, pp. 29-60, 1992.

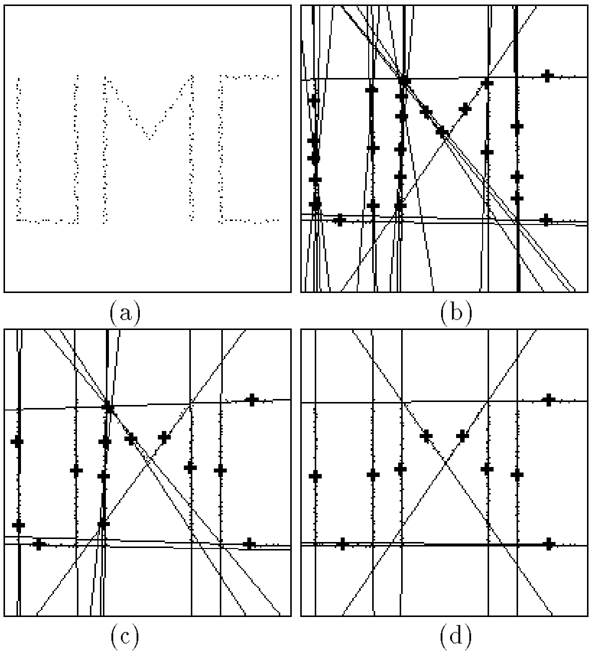


Figure 3: Intermediate results of the CA algorithm on a data set with linear clusters
(a) original image, (b) initial prototypes,
(c) results after 2 iterations, and (d) final results.

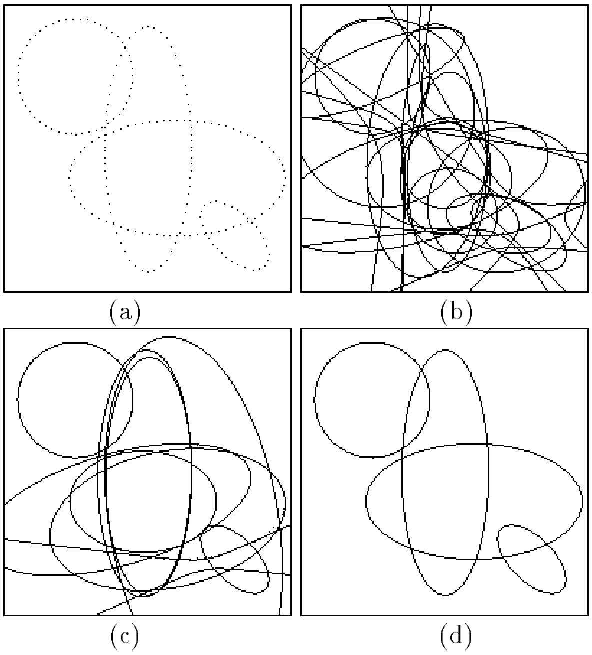


Figure 4: Intermediate results of the CA algorithm on a data set with shell clusters
(a) original image, (b) initial prototypes,
(c) results after 5 iterations, and (d) final results.