Fast Fuzzy Clustering of Infrared Images

Steven Eschrich, Jingwei Ke, Lawrence O. Hall and Dmitry B. Goldgof
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave.
Tampa, Fl 33620
{hall,eschrich,goldgof,ke}@csee.usf.edu

Abstract

Clustering is an important technique for unsupervised image segmentation. The use of fuzzy c-means clustering can provide more information and better partitions than traditional c-means. In image processing, the ability to reduce the precision of the input data and aggregate similar examples can lead to significant data reduction and correspondingly less execution time. This paper discusses brFCM, a data reduction fuzzy c-means clustering algorithm. The algorithm is described and several key implementation issues are discussed. Performance speedup and correspondence to a typical FCM implementation are presented from a dataset of 172 infrared images. Average speedups of 59 times traditional FCM were obtained using brFCM, while producing identical cluster output relative to FCM.

We demonstrate the effectiveness of the algorithm, both in speedup over FCM and how well its partitions correspond with FCM cluster output.

In order to provide speedup and FCM correspondence results, we test against a set of synthetic infrared images of natural scenes. These images were generated for an Army Research Lab project and were used for automatic target recognition (ATR). The natural scenes were observed to cluster well into trees and grass. Each image is a 480x830 8-bit grey-level image and was clustered into 5 clusters. There are 172 images in the dataset. One of Laws' texture energy features [7] was generated to create a two-dimensional vector (pixel intensity and texture). The execution time and cluster correspondence between brFCM and FCM are presented based on these images.

1. Introduction

Clustering is an unsupervised learning technique for grouping like data [5]. In the context of image processing, clustering can be used to group similar features within the image. This segmentation of the image can be done with little a priori knowledge of the image characteristics. A c-means clustering algorithm attempts to separate the data into c distinct clusters. The method for partitioning data is generally a minimization of square error of distance from the cluster center to example [1]. The fuzzy c-means (FCM) algorithm broadens the notion of cluster membership. Each example in the dataset is assigned a membership value in $[0, 1]$ for each cluster.

The use of FCM has been shown to be effective in image segmentation, including medical imaging [3]. However, large image sizes require significant amounts of computation. In [6], a clustering method called 2rFCM is introduced. This algorithm reduces the image precision in order to cluster more efficiently. In this paper, we restate the algorithm and discuss some of the implementation issues that significantly impact performance.

2. brFCM

In [6], a description is given of a modified FCM algorithm known as 2rFCM. The algorithm reduces the number of feature vectors to be clustered, possibly reducing the precision of the data, in order to speed up the clustering. We present an alternate view of the algorithm, generalizing it to arbitrary numeric data. This algorithm is discussed within the image processing domain, however the technique can be applied to many other clustering problems.

The brFCM algorithm consists of two phases: data reduction and fuzzy clustering using FCM. The data reduction phase consists of an optional precision reduction step and an aggregation step. Both steps attempt to reduce the number of feature vectors presented to the FCM algorithm. Specifically, we attempt to reduce the number of distinct examples to be clustered from n to n_0, for some $n_0 << n$. At the same time, we want to preserve partition "goodness."

2.1. Data Reduction: Overview

The first step in data reduction for the brFCM algorithm is quantization. When continuous data of any type is measured and translated into the digital domain, some level of quantization occurs. This quantization is often a reduction in precision and therefore a potential loss of information. However, an assumption is generally (and we feel appropriately) made:

> Small, often human-imperceptible changes in values do not affect the classification of the object in question.

This can be seen as a tolerance to noise or simply the extent of distinguishibility within the human brain. From an analytic point of view, we are clustering examples using a geometric distance metric. Therefore, small changes in values lead to correspondingly small changes in distances. Clustering makes the assumption that groups of examples exist within close distance in feature space.

The second step of data reduction in brFCM is aggregation. For this algorithm, aggregation combines identical feature vectors into a single, weighted example. The existence of identical feature vectors is not common in all data, however where it does exist there is often significant complexity reduction seen by aggregating the duplicate examples. Again, consider the image processing domain and an image with 256 grey-level values of intensity as the only feature. As long as the image contains more than 256 pixels, some level of aggregation can occur. An image of size 800x600 would provide a 99.9% reduction in feature vectors (from 480,000 to 256) at a minimum.

When both quantization and aggregation are used, significant data reduction can be seen. Quantization forces different continuous values into the same quantization level, creating identical feature vectors from "similar" ones. In brFCM, aggregation creates a single example representing the quantization level. The value of this example is taken as the mean value of all full-precision features vectors quantized to this level.

The data reduction step of brFCM attempts to create truly representative examples from the original feature space. The mean value of a set of examples retains some information. In addition, each representative feature vector has an associated weight, corresponding to the number of full-precision examples within the quantization level.

Once the data reduction has been accomplished using quantization and aggregation, the resulting dataset of examples is then clustered using a modified FCM clustering algorithm. Once clustering is complete, the representative feature vector membership values are distributed identically to all members of the quantization level.

Data reduction using quantization will necessarily lose information about the dataset. There is no a priori method of determining an appropriate level of reduction; acceptable precision loss must be empirically determined. As will be shown later in this paper, small precision reductions can produce clusters that closely correspond to FCM.

It should also be noted that quantization is an optional step in data reduction. The brFCM algorithm with only aggregation is functionally equivalent to traditional FCM. If data redundancy is significant, the dataset can be represented in a more compact form for clustering. The brFCM algorithm can then be used with significant computational savings vs. traditional FCM with no difference in clustering output.

2.2. brFCM Details

Once the data reduction phase of brFCM has been performed, the reduced precision image can be presented to the FCM clustering algorithm. FCM is modified to include support for weighted feature vectors. Recall that the aggregation step of data reduction creates representative examples. The weights correspond to the number of aggregated feature vectors.

In more formal terms, consider the set X' of example vectors representing a reduced-precision view of the dataset X. There are n_0 such vectors, such that $n_0 \leq n$. Each X'_q represents the mean of all full-precision members in the given quantization level. In addition, X'_q has an associated weight, w_q, representing the number of feature vectors aggregated into X'_q. Clustering is done through alternating calculations of membership values and cluster centroids. The cluster centroids are calculated by

$$V_i = \frac{\sum_{k=1}^{n_0} w_k \left(u_{ik}\right)^m X'_k}{\sum_{k=1}^{n_0} w_k \left(u_{ik}\right)^m}\ ,\ 1 \leq i \leq c\,; \qquad (1)$$

The cluster membership values are calculated by

$$u_{ik} = \left[\sum_{j=1}^{c} \left(\frac{\|X'_k - V_i\|}{\|X'_k - V_j\|}\right)^{\frac{2}{m-1}}\right]^{-1} \qquad (2)$$

where $1 \leq i \leq c$ and $1 \leq k \leq n_0$.

It is worth noting two particular features of this algorithm:

- When no quantization occurs and the aggregation step does not reduce the dataset, $n_0 = n$ and $w_i = 1$ for all i. The algorithm reduces to traditional FCM.

- When the aggregation step is used by itself, the algorithm also reduces to traditional FCM. However, it is a more efficient calculation since identical terms in the summation are grouped together. This formulation can significantly improve the speed of clustering, *without a loss of accuracy*.

2.3. Examples

As an example of the brFCM algorithm, consider an image consisting of 4 pixels as listed in Table 1, with the values representing intensity levels. For this example, we will quantize the feature space by masking the lower r bits of the feature value.

Feature Vector	Value	Binary Value	Quantized Value	Binary Quantized Value
I_1	25	011001	24	011000
I_2	26	011010	24	011000
I_3	32	100000	32	100000
I_4	32	100000	32	100000

Table 1. Example - 6 Pixel Image

Aggregation by itself would produce one reduced vector, $I' = 32$ with a weight of 2, since I_3 and I_4 are identical. The other two vectors are represented in the reduced dataset as they are, with a weight of 1.

We can also quantize the feature space using a bit mask of (111100), that is $r = 2$, where the width of the quantized bin is 2^r. After quantization, the pixels have the values as seen in Table 1, columns 4 and 5. Aggregation can then be applied to the dataset to provide data reduction. Note the mean value that is used in the clustering step is computed from the full precision value.

Feature Vector	Mean Value	Weight (w_i)
I'_1	25.5	2
I'_2	32	2

Table 2. Example - 2 Pixel Reduced Image

2.4. Image Characteristics

One question that arises from the brFCM algorithm is: "For what type of image is brFCM suited?" At present there are no direct tests to indicate the usefulness of the algorithm on a particular set of images. However, as discussed earlier, many identical feature vectors do indicate successful data compression. When multiple dimensions are considered in an image (e.g. multi-spectral, RGB) the feature space size increases and the possibility of identical vectors is decreased.

The brFCM algorithm can be an effective replacement for traditional FCM if the feature space defined by the data set is similar in size or smaller than the number of vectors in the dataset. As an example, consider an RGB image consisting of $2^8 \times 2^8 \times 2^8 = 2^{24}$ possible values. With no prior knowledge of the image, the probability of any particular RGB value would be $\frac{1}{2^{24}}$. Therefore, aggregation will not significantly compress the data for images containing less than 2^{24} pixels (a 4096x4096 pixel image). Fortunately, many RGB images contain identical color values and therefore compression remains practical. In addition, quantization can be used to reduce the overall size of feature space. Consider quantizing RGB space by $r = 2$ – this will create a space of size $2^6 \times 2^6 \times 2^6 = 2^{18}$. An image of size 2^{18} would correspond to a 512x512 square color image, a reasonable image size.

3. brFCM Implementation

In [6], the feasibility of the brFCM algorithm was demonstrated. However the implementation was inefficient, making the data reduction step costly. As with any implementation, careful attention can significantly improve performance. A literal implementation of FCM leads to many unnecessary calculations; several modifications can be used to reduce the number of operations needed. In addition, quantization and aggregation can be expensive. For this work, quantization was implemented via bit-masking and aggregation was done using a hashing scheme.

3.1. Formula Implementation

Literal translation of both the cluster membership and centroid calculations in FCM lead to inefficient code. Several simple modifications can avoid many unnecessary computations.

Consider first the calculation of the cluster centroids in Equation 1. In a literal implementation, the value of u_{ik}^m would be calculated both for the numerator and denominator. By calculating the value once and temporarily storing it, we save n calculations. In addition, recall the X_k is a s-dimensional vector – the literal implementation calculates u_{ik}^m s times for each example. Thus by utilizing the cached value, there are $n_0 \times s$ calculations saved per cluster centroid, or $c \times n_0 \times s$ operations during each iteration.

Next, consider the calculation of the membership values u_{ik} as given in Equation 2. In this equation, when $i = j$, the fraction evaluates to 1. By adding a special

case check for this condition, we save $c \times n_0$ divisions and $c \times n_0$ exponentiations. Additionally, by arranging the outer loop to be over the index k (representing the particular example in the dataset), we can cache the value of $\|X_k - V_j\|$ for all j values. Each distance is used c times, therefore a savings of $c \times (c-1)$ operations can be realized. Thus a total of $2 \times c \times n_0 + c^2 - c$ calculations can be avoided in the membership function updates during each iteration.

3.2. Quantization

Quantization of a feature space can be done either using fixed-size ranges or variable-sized ranges. The benefit of variable-sized ranges for quantization is that known sparse areas of feature space can be reduced to a single "outlier" example. However, this method will be computationally expensive as the dataset must be examined for such regions. Instead, the brFCM algorithm can be implemented efficiently using fixed-size ranges. For example, image intensity space can be quantized with ranges of size 2^r for some small value r. Using this representation, values can be quickly quantized by masking the lower r bits of an intensity value.

The use of bit-masking for quantization is an efficient method for efficiently implementing brFCM, however this technique can only be applied to integer-valued data. A more general approach to quantization can be

$$X_q = Q_r \times \left\lfloor \frac{X}{Q_r} \right\rfloor \tag{3}$$

where Q_r is the quantization size, and $\lfloor X \rfloor$ is the integer floor function.

3.3. Aggregation using Hashing

In [6], the authors introduced the bit-reduction of the example space, however the implementation of the algorithm required significant time overhead in bin creation (the aggregation step in Section 2). Ke's algorithm involved a forward search of all examples looking for duplicate examples. Even for moderate-sized number of bins the creation overhead is large. A more sophisticated aggregation method utilizing hashing was implemented.

A hash table is a generalized array data structure in which the index to a data element is computed as a function of the key provided [2]. In the best case, one data access is required to find the data location. Collisions, or the existence of multiple data items at the same hashed array location, are resolved using the chain method. This is a linked list of elements at a particular hash location. Searching this linked list can be costly, therefore minimizing the number of collisions that occur is an important goal in designing a hash table.

The hashing function is an important design consideration in creating a hash table. For brFCM, a universal hashing function as described in [2] is used. The function is given by

$$h_a(x) = \left(\sum_{i=0}^{s} a_i x_i \right) \; mod \; m \tag{4}$$

The s denotes the number of features in the example (for multidimensional data). The values a_i are randomly chosen from the range $0 \leq a_i \leq m$. The value m is defined by

$$m = \frac{Expected\ Number\ of\ Items}{Expected\ Number\ Of\ Collisions} \tag{5}$$

We use an expected number of items, corresponding to the dataset reduction rate, of 75%. The expected number of collisions is set to 3. For more details on hashing, refer to [2].

4. Experiments

In this section, we detail the experimental results of using brFCM on a set of infrared images. Data reduction is the key step in the brFCM algorithm and as the experiments below demonstrate, significant reduction is possible. The corresponding speedups in brFCM demonstrate dramatic reduction in clustering time. Finally, brFCM is only a useful enhancement to FCM if it produces similar cluster results. We present a correspondence metric relative to FCM in which the tradeoff between speedup and correspondence with FCM can be clearly seen.

4.1. Data Reduction

Our 172 ATR images are 8-bit (256-value) infrared images of size 398400 pixels. We use two features: intensity and one Laws' Texture Energy feature [7] (see [4] for more details). The feature space is smaller than the number of pixels, therefore we expect significant data reduction even with no quantization performed (i.e. the aggregation step only). Table 3 shows the remarkable level of reduction seen in these images. When $r = 4$, there are only 16 pixel intensity quantization levels (2^4), helping create an extremely small dataset to cluster.

4.2. Performance Speedups

Once data reduction is done, the brFCM algorithm should generally run faster since there are not as many

r	Mean Number of Examples After Reduction	Mean Reduction Rate
0	13533	96.60 %
1	4744	98.81 %
2	1605	99.60 %
3	544	99.86 %
4	193	99.95 %

Table 3. Reduction Results for Infrared Images

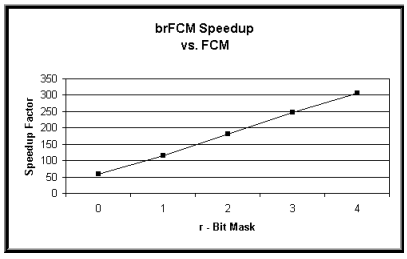

Figure 1. Speedup vs. Traditional FCM

examples to cluster. However, as discussed earlier, the data reduction step can potentially swamp any clustering speedup. We examine the speedup between a literal implementation of FCM and the brFCM implementation described above.

Both the literal FCM and brFCM implementations were run against 172 infrared images. Timing was started when the clustering function was called and stopped when the function returned. No input or output is done in the clustering function. On return, the timing is calculated and the results are printed. In the case of brFCM, the timing includes the data reduction phase, the clustering phase, and the distribution of cluster membership values to all members of the original dataset. The results presented below consist of timing results and relative speedup values. All tests were run on a Sun Ultra 10 workstation with 256MB RAM. The timing results are reported as the mean accumulated CPU time in seconds, across the entire dataset of images. The speedup results are presented as the mean of individual speedups, for all images in the dataset.

As can be seen in Table 4, the speedups are remarkable. Fuzzy clustering requires a significant amount of work proportional to the number of examples clustered. Therefore, the significant reduction in number of examples accounts for much of the speedup seen.

r	FCM Time (sec)	brFCM Time (sec)	brFCM Speedup
0	791.42	16.60	58.90
1	791.42	7.57	113.82
2	791.42	4.42	180.67
3	791.42	3.22	246.89
4	791.42	2.61	306.17

Table 4. Timing Results for Infrared Images

4.3. Correspondence with FCM

The brFCM algorithm can produce significant speedup vs. FCM. However, do the clustering results vary significantly? Clustering is usually an unsupervised learning technique, therefore pixel-level ground truth generally does not exist and hence accuracy cannot be measured. This algorithm is a faster variant of FCM, so we would like to measure the correspondence in clustering results with FCM. We adopt the metric used in [6]. In that work, Ke used a cluster discrepancy measure. In this paper, we use a related measure, the cluster correspondence.

The cluster correspondence between two partitions can be found using the following algorithm [6]. Consider the set of image pixels $X = \{x_1, x_2, ..., x_n\}$. Clustering creates a partition P of X into c clusters. In the case of fuzzy clustering, we harden a partition by assigning x_i to a cluster via the maximum fuzzy membership. Consider two partitions of X: $P^1 = \{C_i^1 | i = 1, 2, ..., c\}$ and $P^2 = \{C_i^2 | i = 1, 2, ..., c\}$.

First, we define the maximal intersection of $C_i^1 \in P^1$ and $C_j^2 \in P^2$ as the cluster in P^2 with the largest number of matches to C_i^1:

$$C_i^1 \cap_{max} C_j^2 = max \{|C_i^1 \cap C_j^2| \, | j = 1, 2, ..., c\} \quad (6)$$

The correspondence mapping $P^1 \rightarrow P^2$ can then be defined as the mapping of cluster C_i^1 to C_j^2 such that $C_i^1 \cap_{max} C_j^2$, for all clusters in P^1.

The algorithm for calculating the cluster correspondence is given below.

- Find correspondence mapping $P^1 \rightarrow P^2$ and $P^2 \rightarrow P^1$.
- Correspondence rate $Corr_1$ is the sum of all maximal intersections in the correspondence mapping, divided by number of examples in X.

$$Corr_1 = \left(\frac{\sum_{i=1}^{c} |C_i^1 \cap_{max} C_j^2|}{|X|} \right) \quad (7)$$

  .
- Repeat for $Corr_2$ (using $P^2 \rightarrow P^1$).
- Correspondence rate $CR = max(Corr_1, Corr_2)$.

See [6] for more discussion and details on the need for calculating the cluster correspondence mapping in both

directions. Using this technique, we can measure the correspondence between two partitions of a dataset. In particular, the correspondence of brFCM results can be compared to FCM results using this technique. The correspondence rate was calculated between FCM and brFCM on all 172 images, for $r = 0, 1, 2, 3, 4$ (Table 5).

The question arises: how significant are the brFCM-FCM correspondence rates as r increases? FCM is sensitive to the initial values of the cluster centroids. We can generate 30 FCM partitions of the same image by choosing 30 random cluster centroid initializations. We can then compare the brFCM-FCM correspondence rate for an image to the mean FCM-FCM correspondence rates among the 30 partitions. For $r = 1$, brFCM-FCM correspondence rates in 171 (of 172) images were higher than one standard deviation above the mean FCM-FCM rate. At $r = 3$, 163 of the images had higher brFCM-FCM rates. Once $r = 4$, the brFCM-FCM correspondence is expected to drop since 256 grey-levels are quantized to only 16 different values. However, the brFCM-FCM correspondence rates for 137 of 172 (80%) images were higher than one standard deviation above the mean FCM-FCM rate for that image. This indicates that brFCM generally creates partitions very similar to FCM, given the same centroid initializations.

r	Mean Correspondence Rate	Standard Deviation
0	100%	0
1	97.33%	6.53
2	92.22%	13.16
3	85.31%	18.56
4	62.45%	27.58

Table 5. FCM Correspondence

4.4. Discussion

The brFCM algorithm generates significant speedup over literal FCM in the infrared image dataset. However, the correspondence to FCM drops when too much quantization occurs. A tradeoff exists between the FCM correspondence and speedup, as can be seen by Figure 2. The level of quantization is domain-dependent, and therefore must be considered for each new problem.

It should also be noted that aggregation-only brFCM reduces the average number of pixels in the image by 96%. As discussed earlier, brFCM is functionally equivalent to FCM when $r = 0$. Therefore, this technique can be valuable even when quantization is not desired and full precision is needed. For identical results, brFCM was on average 58.9 times faster than FCM on the infrared images.

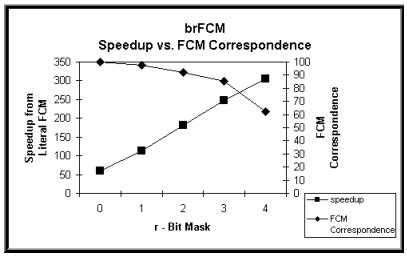

Figure 2. brFCM Speedup vs. FCM Correspondence

5. Conclusion

The brFCM algorithm described in this paper was shown to provide a substantial speedup versus the traditional FCM implementation for the 172 infrared images considered. Through the use of quantization and aggregation, the number of examples to be clustered can be dramatically reduced. With an efficient implementation of brFCM, the average speedup was 59 times vs. FCM with identical cluster output for these images. Quantization of the feature space can further improve speedup rates vs. FCM. For many image clustering problems, brFCM is a fast alternative to traditional FCM.

6. Acknowledgment

This work was partially supported by the U.S. Army Research Laboratory under grant DAAD17-00-P-1451.

References

[1] J. C. Bezdek and S. K. Pal, editors. *Fuzzy Models For Pattern Recognition*. IEEE Press, New Jersey, 1991.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1998.

[3] L. O. Hall, A. M. Bensaid, L. P. Clarke, et al. A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. *IEEE Transactions on Neural Networks*, 3(5):672–682, Sep 1992.

[4] L. O. Hall and S. Eschrich. Robust recognition of interesting objects in images. Technical report, University of South Florida, 2000.

[5] A. Jain and R. Dubes. *Algorithms That Cluster Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[6] J. Ke. Fast accurate fuzzy clustering through reduced precision. Master's thesis, University of South Florida, 1999.

[7] K. I. Laws. Texture energy measures. In *DARPA Image Understanding Workshop*, pages 47–51. DARPA, Los Altos, CA, 1979.