A JAVA SIMULATION TOOL FOR FUZZY CLUSTERING

M.A. Egan[1], M. Krishnamoorthy[1], K. Rajan[2]

[1]Department of Computer Science
[2]Department of Materials Science and Engineering
Rensselaer Polytechnic Institute
Troy, NY

Emerging technologies on the World Wide Web promise to make program, algorithm and concept simulations universally accessible and Java appears to be the best technology available. Simulations involving animation and visualization have a tremendous benefit when applied to various algorithms. We present a simulation tool for experimenting with concepts in fuzzy clustering that has proved useful in visualizing the results and demonstrating the computation method of the algorithms. This is an asset when working with people unfamiliar with the mechanics of fuzzy clustering, such as non-computer scientists or students. This system was integral in the development of an algorithm capable of locating an unknown number of clusters embedded in background noise.

INTRODUCTION

The advantages of using graphical visualization and animation to depict the dynamic behavior of computer algorithms is widely appreciated. Studies have shown that, for many types of algorithms, animated sequences enhance learning and understanding [Lawrence, et al., 1994; Stasko, et al., 1993]. Some of the variety of algorithms that have been subject to animation include: recursion; curve fitting; sorting and searching; parsing; file compression; cryptography; network flow; dynamic and linear programming. There are other algorithms that tend to have an esoteric quality which animation and visualization could help to dispel. For example, it is only recently that work has appeared on the animation of artificial neural networks [Jackson and Morton, 1996] and genetic algorithms [Jackson and Fovargue, 1997]. This paper will present a system that demonstrates various aspects of fuzzy clustering.

The clustering of data into meaningful subsets is a major area of research and development driven by the need to process data and information. The use of fuzzy set theory is becoming widely used as the results of the algorithms are excellent, producing not only crisp decisions when necessary, but also corresponding degrees of support. The application areas of fuzzy clustering are numerous, ranging from image recognition and web searches to trend analysis and crystallographic statistics. As fuzzy clustering moves out of the realm of computer science and into other application areas, there is a need to demonstrate not only the results, but the method of computation and the role of various parameters in the search process. There is also a need when experimenting with different algorithms to compare the results directly. FCLUST (Fuzzy CLUstering Simulation Tool) allows users to do all of the above by providing real time simulation over the Internet. FCLUST allows on-line experimentation by which users may alter parameters and observe the resultant changes in clustering.

This paper will first present the background of fuzzy clustering and the techniques used in FCLUST. It will also highlight some aspects of the visualization system and detail the pros and cons of using Java in a visualization and computational tool.

FUZZY CLUSTERING

In 1965, Lotfi Zadeh introduced fuzzy sets to represent vagueness. Fuzzy sets extend to clustering in that each object of the data set may be fractionally assigned to multiple clusters. This allows for ambiguity in the data and yields detailed information about the structure of the data. The central idea in fuzzy clustering is the membership function that measures the degree to which objects satisfy imprecisely defined properties. The information contained in the fuzzy membership matrix is valuable in many types of comparisons and analyses. In data analysis, the membership values give an indication of confidence that a point belongs to a particular cluster. Although the effect of some of these variations could show up in the hard clusterings, they are not as sensitive as the fuzzy clusterings.

Most fuzzy clustering techniques minimize [Bezdek, 1973; Dunn, 1973]:

$$F = \sum_{k=1}^{c} \sum_{i=1}^{n} (u_{ik})^m (D_{ik})^2 \quad \textit{subject to } u_{ik} \in [0,1] \textit{ and } \sum_{k=1}^{c} u_{ik} = 1 \quad \forall i \tag{1}$$

In this equation, c represents the number of clusters; n, the number of data points; u_{ik}, the membership coefficient of the ith point in the kth cluster; and D_{ik}, the distance from point i to the kth cluster center, v_k. This is an iterative technique that refines the cluster centers, sizes and weights at each iteration.

Dave's noise clustering algorithm (RFC) [Dave, 1991] is incorporated into FCLUST, identifying clusters embedded in background noise. RFC identifies the noise points and groups them as the *(c+1)*st cluster. This is achieved by defining two distance metrics:

$$D_{ik}^2 = \begin{cases} (x_i - v_k)^T A_k (x_i - v_k) & \textit{for } k = 1 \textit{ to } c \\ \\ \dfrac{\lambda}{nc} \displaystyle\sum_{j=1}^{c} \sum_{i=1}^{n} (D_{ik})^2 & \textit{for } k = c+1 \end{cases} \tag{2}$$

While RFC generates good results, the number of clusters must be specified *a priori*. The clustering algorithm used by FCLUST incorporates a validity measure, giving an idea of how well the clusters fit the data set, to determine the optimum number of clusters. The general idea is to run a clustering algorithm, such as RFC, for various values of c. The statistics of the clustering are measured for each value of c and a maximum (or a minimum, depending on the measure used) indicates the number of clusters present in the data.

The measure used in FCLUST maximizes the following three criteria: fuzzy hypervolume (FHV) [Gath & Geva, 1989], partition density (PD) [Gath & Geva, 1989] and average fuzzy density (AFD) [Egan, et al., 1997]. FHV is a fuzzy measure of the volume encompassed by the clusters, while PD is measured as the total number of central members divided by total partition volume. AFD was developed to circumvent some of the shortcomings of PD and is the mean of the density of each cluster measured as central members divided by cluster hypervolume. The fuzziness in these measures refers to the fact that each of the computations is weighted by the membership or confidence value.

The validity measure used in FCLUST is a linear combination of FHV, PD and AFD. A maximum value indicates the optimal partition. FCLUST has been used in several applications and shows promising results.

THE FCLUST SYSTEM

The goal of FCLUST is to illustrate the information available through fuzzy clustering and the path by which different algorithms achieve these results. FCLUST has been used to demonstrate fuzzy cluster evolution to both students and non-computer scientists. It has also been used as a tool to develop better clustering algorithms and validity measures. The FCLUST system was originally developed using C and the Simple User Interface Toolkit (SUIT) from the University of Virginia and has recently been ported to Java. This provides accessibility to a broader range of users by allowing World Wide Web-based simulations.

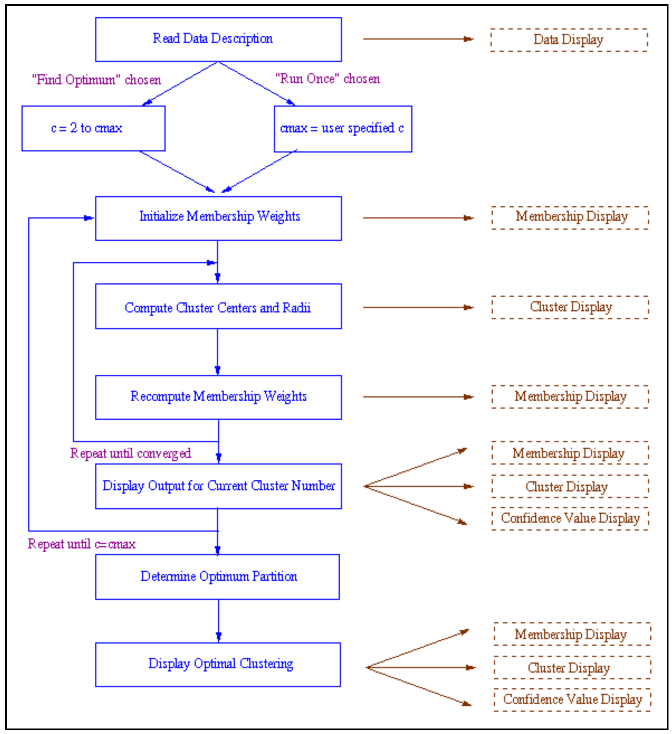
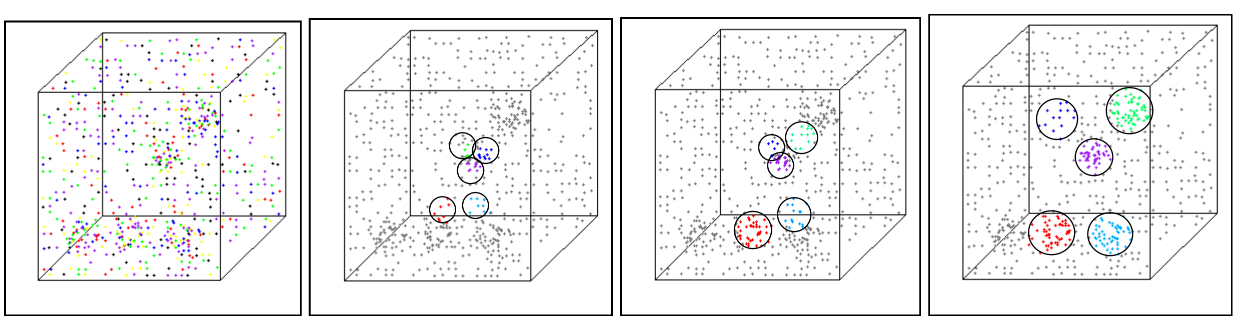


Figure 1 Outline of FCLUST System. Solid boxes represent computational flow while the dashed boxes represent visual flow.

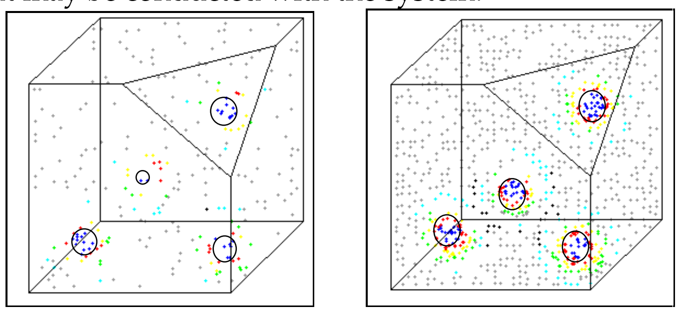
The computational and visual flow of the system may be seen in Figure 1. One of the features of FCLUST is that as the system iteratively recomputes the membership values, cluster centers and radii, the user is shown these new values. Through this animation, one may observe the migration of points as cluster centers emerge. To illustrate some of the animation, a series of membership displays is shown in Figure 2. The membership values are randomly assigned as seen in Figure 2a. As the system progresses, the cluster points migrate toward likely cluster centers. It should be noted that the light gray points represent noise (or points that do not belong to any cluster). In the application, each cluster is represented by a different color. Figure 2 depicts this information by enclosing each cluster within a circle. Each data point is considered to be a member of the cluster to which it has the highest membership value. This assumption is for display purposes only.



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

Figure 2 Sample animation. Circles represent identified clusters.

Another option, within FCLUST, displays the highest confidence value for each point (Figure 3) to give an indication of how strongly the evidence suggests clusters at particular locations. The confidence value display incorporates a color assignment ranging from red, most confident, to black, least confident, while noise points remain white. The confidence in a cluster's presence at the computed location has a direct relationship with the concentration of high confidence points within that cluster. This is indicated in Figure 3 as those points enclosed in the circles. For example, in Figure 3a, the second cluster from the left has a lower density of high confidence points when compared to the other clusters. This implies that the confidence in the location of the center and radius is lower for this cluster. With the addition of more data points (Figure 3b), the location of this same cluster migrates towards the actual cluster center and the density of high confidence points increases. This example illustrates a sample simulation that may be conducted with the system.



(a) (b)

Figure 3 Confidence value display for sample data. Circles contain points with high confidence values (> 0.9).

Another benefit of the FCLUST system is that it provides insight into the effects of the clustering parameters. With this simulation system, users can experiment with various parameters, such as the fuzziness value, m; noise parameter, λ; number of clusters, c; and initial values of the membership coefficients, u_{ik}. This allows the direct observation of the resulting cluster changes which is beneficial for educational purposes and algorithm development.

APPLICABILITY OF JAVA

This section discusses the applicability of Java to a visualization system such as FCLUST, which involves both user interaction and a high degree of computing. The user interface must be designed to allow data input and display the results in a meaningful fashion. In addition, the computational component must not be so slow as to discourage its use.

The user interface was originally written using SUIT which provides routines similar to those found in Java's Abstract Window Toolkit. Several of the differences between these two methods of producing a user interface include the fact that SUIT does not require the explicit layout of the objects using a coordinate system or grid layout. They may be interactively moved into place once the system is running. The grid layout of Java is a tedious method of creating these objects and there is no way to enforce an identical layout across architectures.

The fact that Java is an interpreted language has both advantages and disadvantages. An advantage is that while a C based language, such as SUIT, requires a recompilation for any system upgrades or architecture changes, Java is architecture independent. A disadvantage is the speed with which Java operates. While it is slower than a procedural language such as C, it is faster than past interpreted languages such as BASIC, Tcl and PERL. This is due to the fact that it is interpreting bytecodes instead of actual code.

While C and Java both have multithreading capabilities, Java has a cleaner and easier to use approach which encourages more programmers to use it. Multithreading is advantageous especially in an animation. It allows animation loops to sleep for a second between each frame without causing the whole system to pause. Multithreading is especially important in interactive, networked environments where data transmission is slow, but the CPU can still work on other things. For example, FCLUST uses multithreading to allow computation to continue while the data window is being updated.

The security features of Java confine a program to the Java execution environment and deny access to other parts of the computer. This adds the overhead that all files must be accessed through a URL and placed in a publicly readable directory. It also disables the use of temporary files for data storage by the program and makes the output of data files difficult. While creative programming resolves these problems, it is not as straightforward as one would desire.

Java and its development environments are still evolving. We expect to see many enhancements and improvements over the next few years that address the aforementioned concerns.

CONCLUSION

This paper presented a brief overview of the FCLUST simulation system for fuzzy clustering. Interactive simulations add significant value to educational material by demonstrating important concepts and allowing on-line experimentation with various aspects of the algorithms. This system has also proved useful in applications such as crystallographic statistics and economic trend analysis where the users are unfamiliar with the mechanics of fuzzy clustering.

Emerging technologies on the web promise to make program, algorithm and concept animations universally accessible and Java appears to be the best technology available. Java is still a relatively young language and we expect enhancements and improvements over the next few years to encourage the use of Java in scientific computation.

ACKNOWLEDGMENT

The authors would like to thank Michael Kondorf for porting FCLUST from C to Java.

BIBLIOGRAPHY

Bezdek, J.C., Fuzzy Mathematics in Pattern Classification, Ph.D. Thesis, Applied Mathematics Center, Cornell University, Ithaca, 1973.

Dave, R.N., "Characterization and Detection of Noise in Clustering", *Pattern Recognition Letters,* vol. 12, no. 11, pp.l 657-664, 1991.

Dunn, J.C., "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact, Well Separated Clusters", *Journal of Cybernetics*, vol. 3, pp. 32-57, 1973.

Egan, M.A., Krishnamoorthy, M. and Rajan, K., "Validity Measure for Robust Fuzzy Clustering of Crystallographic Statistics", to be submitted for publication, 1997.

Gath, I. and Geva, A.B., "Unsupervised Optimal Fuzzy Clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 11, no. 7, pp. 773-781, July 1989.

Jackson, D. and Fovargue, A., "The Use of Animation to Explain Genetic Algorithms", *Proc. ACM SIGCSE Technical Symposium on Computer Science Education,* San Jose, California, 1997.

Jackson, D. and Morton, I.G., "Algorithm Animation of Neural Networks", *Proc. ACM SIGCSE/SIGCUE Conf. On Integrating Technology into Computer Science Education*, Barcelona, 1996.

Lawrence, A.W., Badre, A.M. and Stasko, J.T., "Empirically Evaluating the Use of Animations to Teach Algorithms", *Proc. 1994 IEEE Symposium on Visual Languages*, St. Louis, pp. 48-54, 1994.

Stasko, J.T., Badre, A. and Lewis, C., "Do Algorithm Animations Assist Learning? An Empirical Study and Analysis", *Proc. INTERCHI'93 Conf. On Human Factors in Computing Systems*, Amsterdam, Netherlands, pp. 61-66, 1993.