ST05

THEY ASKED FOR A SEGMENTATION SCHEME, NOT CLUSTERS

Jeff Zeanah, Z Solutions, Inc. and R. Gary Huff, Flamingo Solutions, LLC

ABSTRACT

A common business request is to develop a segmentation scheme to assist in understanding customers and thus approach the market. The qualitative analyst generally converts these requests to some form of clustering using available data, because this is what the analyst knows how to do. The restrictions of clustering approaches may lead to solutions having little to do with the actual business request leading to frustration for all parties. Using Genetic Algorithm approaches in PROC GA in SAS/OR (experimental in 9.1.3, with expected release in 9.2) the analyst may optimize a segmentation scheme on any need that can be quantified. This presentation presents a simple application to introduce the concepts of Genetic Algorithms and initial approaches to accomplish an implementation in PROC GA.

INTRODUCTION

With the advent of Internet personalization and Customer Relationship Marketing (CRM) systems, significant attention has been placed on one-to-one marketing – the concept of treating each customer as an individual. Modeling activities generally labeled data mining have been employed to accomplish these tasks. Predictive models, such as logistic regression, neural networks and decisions trees, are used to estimate for each customer or each potential customer the likelihood that they will respond to a specific product offering. The techniques are employed for customized mail outs and customized web experiences.

Contrasted to this one-to-one approach are segmentation approaches to create a compact understanding of the market. There has been no decrease in the discussion or need for segmentation strategies in spite of the growth of these personalization activities. These segmentation strategies are most often satisfied with a clustering approach. Analytically, a dataset is created consisting of variables defining a multi-dimensioned data space. Clusters are typically created by finding centroids in the data space and minimizing the total of the distance of the observations from the centroid of its assigned cluster. Predictive modeling techniques offer greater analytical flexibility, offering greater predictive power than the clustering techniques. Therefore, why is there still a great demand for clustering approaches?

These two approaches offer two distinct value propositions for an organization. The predictive models make a prediction of what will happen given some action taken by the company. Through this understanding the organization can attempt to optimize their decisions. The direct marketing approaches discussed above are an example of these optimization approaches. The segmentation approaches (usually addressed through clustering) assist the organization to understand a complex system. The model serves to explain the full complexity of the system (such as millions of customers) in a way that may lead to rational discussion. Discussing plans for 20 segments is practical; discussing plans for 20 million independent customers is not. This use of models to increase understanding is the desire for segmentation, and is interpreted by analysts as a clustering need.

However, clustering is a specific minimization scheme. Although several algorithms are used to determine this minimization, the schemes are basically the same. Unfortunately, the request from the marketing department was not for a minimization scheme of the market place, but a compact understanding of the market place.

The authors of this paper through several years of consulting, teaching and discussions with colleagues have heard this problem repeatedly, “We need a better clustering technique.” The problem however, is not with clustering techniques. The problem is a disconnect between the clustering algorithms and the application needs. Analytically, the issue is the lack of flexibility in defining the objective function when using clustering techniques.

Objective functions with greater flexibility can be defined through Genetic Algorithm approaches to solve any business need that can be defined. Specifically, these techniques can deal with the nuances of the request of understanding the marketplace. This paper presents an introduction to Genetic Algorithm approaches and an example of their implementation to this marketing problem using PROC GA in SAS/OR.

GENETIC ALGORITHMS

Genetic Algorithms are an optimization approach based loosely on the concept of natural selection. A problem with a measurable objective function is defined and possible solutions to the problem are proposed and evaluated by the objective function. The objective function may be maximized or minimized. Solutions that have better scores by the objective function are considered “fit” and are passed to the next generation. New solutions are created through methods analogous to natural selection methods. A solution may “mutate” by having a component, a “gene”,

change or a new solution may be formed by "Genetic Crossover" based on the combination of two observations. The following presents simple examples of these concepts. In this example the objective function is to be maximized.

A population of four solutions are analyzed and presented on the following table.

First Generation		
Observation #	Genes	Objective Value
1	ABCD	99
2	EFGH	78
3	LMNO	56
4	WXYZ	65

The best observations survive to the next generation. In this example only the one best observation survives (observation 1). Three new solutions are generated, two are generated by crossover, one by mutation.

Second Generation			
Observation #	Genes	Objective Value	Source
1	ABCD	99	Survive
2	ABGH	101	From Parents (ABCD & EFGH)
3	EXYH	89	From Parents (WXYZ & EFGH)
4	AUCD	65	Mutation of ABCD

This process is repeated hundreds or thousands of times until convergence to a solution is found.
Clearly the process can be computationally heavy and there is no guarantee of a perfect solution. However, a good solution can be found that can fit the objectives of the marketing department. This is true even if the objective is convoluted, recognizing realities of the real world.

EXAMPLE PROBLEM

The Genetic Algorithm approach to solving a segmentation request has been used by the authors to solve proprietary requests. Presented here is a subset of those methods on a basic non-proprietary problem designed to present the basic concepts. The intent of this paper is to show the flexibility of the methods.

PROBLEM STATEMENT

A business has two variables (variables: *x1* and *x2*) known for all customers in their data that may be used to perform a segmentation strategy. For the subset of the data studied in this analysis there is a measured binary response to a marketing strategy (variable: "event"). This variable was collected through a survey and is only known for the subset of data being analyzed. It is not known for the general population. It may be used for evaluating the performance of a segmentation scheme but it cannot be used for developing the segmentation rules.

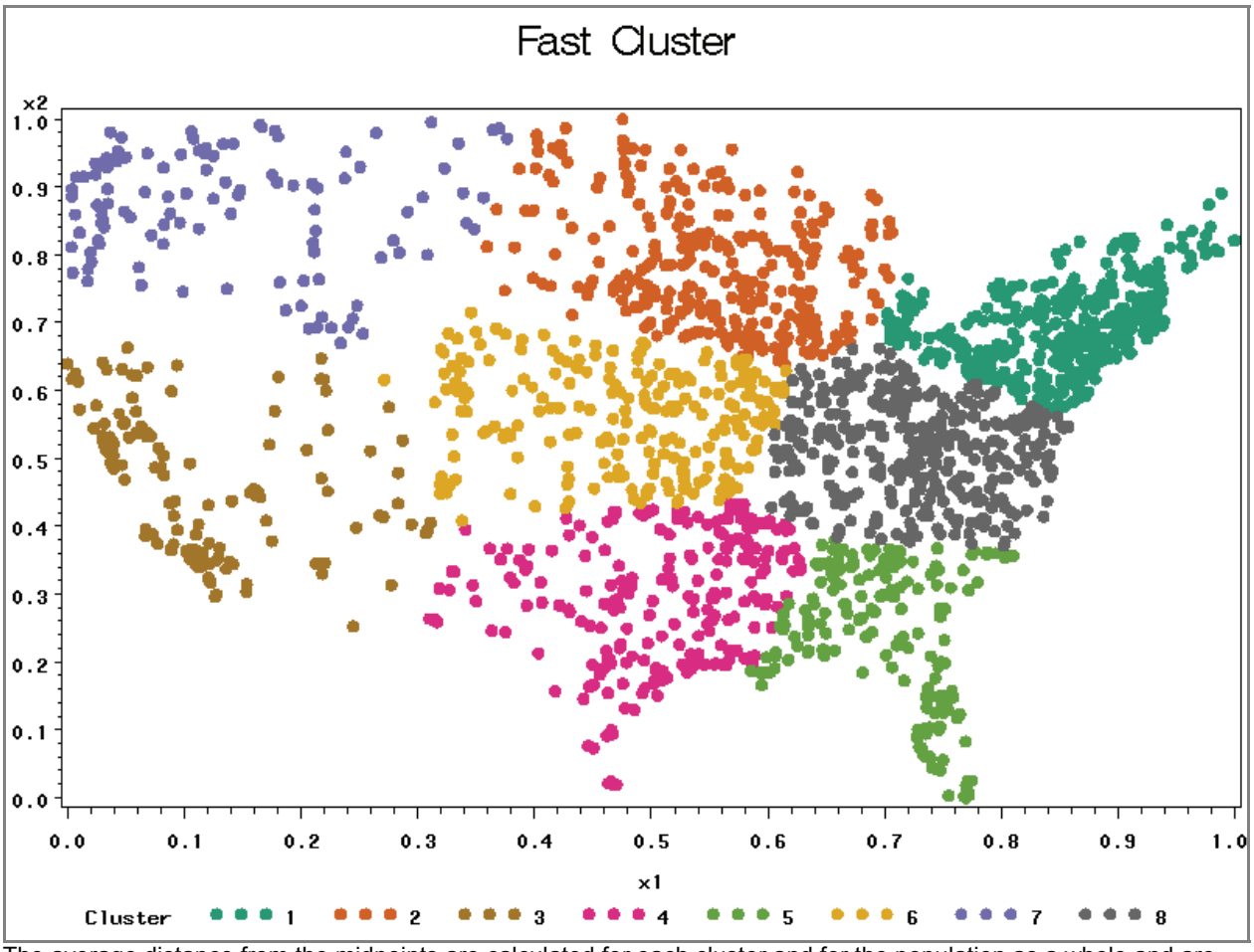
The business objective for this problem is to group the customers by common characteristics of *x1* and *x2* in such a way that the positive responses of *event* (event = 1) are concentrated into segments to the degree possible.

INITIAL CLUSTERING SOLUTION

An initial solution to this problem is found using PROC FASTCLUS.

```
proc fastclus data = paper.std1 maxclus = 8 maxiter = 500
    out= fastclusout;
    var x1 x2;
    run;
```

FASTCLUS uses an iterative method to minimize the Euclidian distance from each point to the midpoint of the cluster. The two dimensional results are graphed below presenting a familiar shape and the eight clusters.



The average distance from the midpoints are calculated for each cluster and for the population as a whole and are presented in the following table.

Segment	Number in Cluster	Avg. Distance from Midpoint
1	439	0.07902
2	305	0.10439
3	143	0.11677
4	231	0.10897
5	175	0.10735
6	233	0.10502
7	113	0.12541
8	361	0.08682
Average	2000	0.09858

INITIAL GENETIC ALGORITHM SOLUTION

The genetic algorithm methodology implemented using PROC GA in the SAS/OR package allows the user to customize the optimization function using a set of programming statements similar to those allowed in a DATA step. Through these statements the process of clustering of FASTCLUS can be replicated. The full discussion of the code required to accomplish this is beyond the scope of this paper, however, the most significant statements indicating general principles are discussed.*

PROC GA is called, indicating: the initial data set to be analyzed; the location to save the last population; a random

* All code and data used in this paper are available on-line at www.zsolutions.com/ga_paper

number seed to ensure that the results are repeatable; and a coding of the number of genes of the genetic algorithm solution. In this case a set of 16 real numbers are the genetic coding representing a pair of x1 and x2 codings for each eight clusters.

```
proc ga data1=paper.std1
    lastgen = paper.test seed=12345;
    call SetEncoding('R16');
…
```

A function is defined to evaluate each observation through each of the solutions in the population. The statement "call ReadMember" calls a solution from the current population loading the 16 genes into the array *a*. The variable test is defined calculating the Euclidian distance from each observation to the eight pairs of genes for each cluster. The observation will be assigned to the cluster that gives the minimum distance. (The reader will notice that there is nothing in this process that forces the genes to represent the midpoints of the clusters. However, the tendency will be to find the midpoint as the gene.) The minimum distance over the eight clusters is assigned to the variable *dist* and is added to the variable *sum* which is the overall sum of minimum distances to the cluster for all observations. Finally, the function returns the average distance by dividing summed distances by the total number of observations.

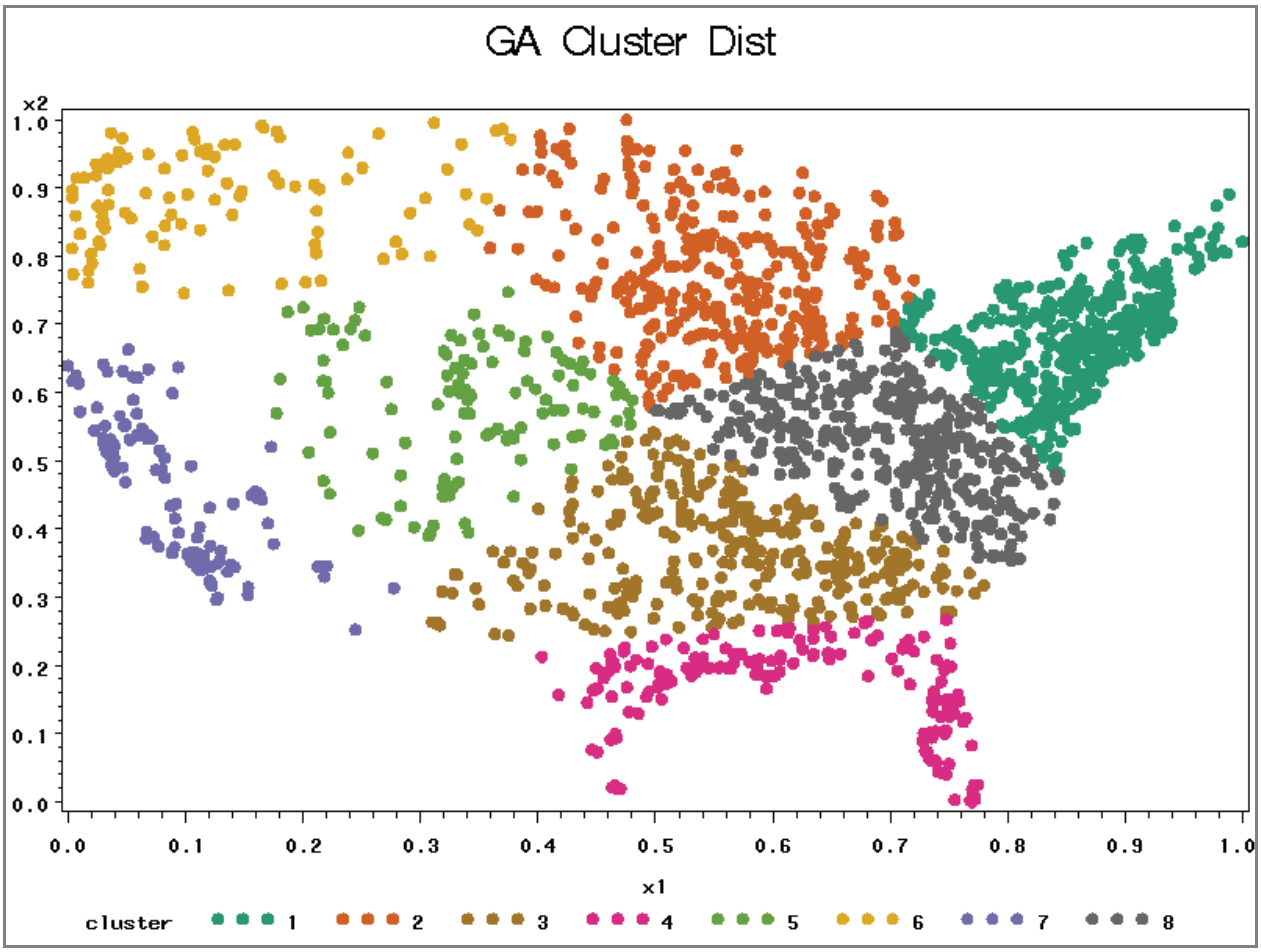
```
function analyze(selected[*],x1[*],x2[*], nbrClusters, nbrObs);
       …
   call ReadMember(selected,1, a);
       do k = 1 to nbrObs;
              dist = 999999;
              do i = 1 to nbrClusters;
                     test = sqrt((a[(i*2)-1]-x1[k])**2 + (a[i*2]-x2[k])**2);
                     if (test < dist) then dist = test;
              end; *i;
              sum = sum + dist;
       end; *k;
       return(sum/nbrObs);
endsub;
```

Several commands are needed that control genetic algorithm optimization. First the objective function "analyze" discussed above is called with the statement "Call SetObjFunc". The option "0" is to minimize the function. The statements "call SetMutProb" and "call SetMutUniform" are used to control the mutation rates. The statements "call SetCross2point", "call SetCrossRoutine" and "call SetCrossProb" control the genetic crossover options of the routines. (Please see the PROC GA documentation for the details of these operations.) Finally, two statements give an indication of the processing required. The "call Initialize" statement initializes a population of 150 solutions, in each subsequent generation there will also be 150 solutions in each population. The statement "call ContinueFor(6000)" indicates that there will be 6,000 iterations of mutation and crossover and evaluation.

```
call SetObjFunc('analyze',0);

if iter <= 1000 then do;
              call SetMutProb(0.65);
              call SetMutUniform(4);
       end;
       else if iter <= 2000 then do;
              call SetMutProb(0.65);
              call SetMutUniform(2);
       end;
       else do;
              call SetMutProb(0.50);
              call SetMutUniform(1);
       end;
call SetCross2point(1);
*call SetCrossRoutine('TwoTypes');
call SetCrossProb(0.7);
…
call Initialize('DEFAULT', 150);
call ContinueFor(6000);
```

This solution gives a result similar but not exactly the same as the FASTCLUS procedure.



Segment	Number in Cluster	Avg. Distance from Midpoint
1	479	0.08230
2	337	0.10828
3	311	0.11256
4	164	0.12258
5	120	0.10534
6	102	0.11536
7	121	0.10387
8	366	0.09535
Average	2000	0.10145

Clearly, this approach is computationally inefficient, but the results are very similar to those found by FASTCLUS. The solution was found with 6,000 generations. With more generations, a solution equal to FASTCLUS likely could have been found. The strength of the Genetic Algorithm approach is complete flexibility in how the objective function may be structured. As discussed above the variable "event" is available on the subset of data used to develop the segmentation routine. It is desired to have the common response to "event" grouped together to the greatest degree possible. The next step is to improve the segmentation with this in mind.

GINI INDEX OF HETEROGENEITY

The Gini index of heterogeneity measures the dispersion of an object of interest. In this case it is the dispersion of observations with *event = 1*. The formulation of the Gini index of heterogeneity is:

$$G = 1 - \sum_{i=1}^{k} p_i^2$$

for k groups. Null heterogeneity occurs when p_i=1 for any group *i* and p_i=0 for all other groups. Maximum heterogeneity occurs when the observations are uniformly distributed among the *k* levels, that is p_i=1/ *k* for all groups.

The Gini index can be used to evaluate our two clustering attempts with the intent to minimize the resulting value.

	p_i	p_i
Segment	FASTCLUS	GA CLUSTER
1	0.1185	0.1309
2	0.1802	0.1926
3	0.0617	0.2123
4	0.1728	0.1111
5	0.1160	0.0790
6	0.1259	0.0543
7	0.0667	0.0494
8	0.1580	0.1704
Gini Index	0.861	0.848

The maximum heterogeneity for eight clusters is when p_i=1/8

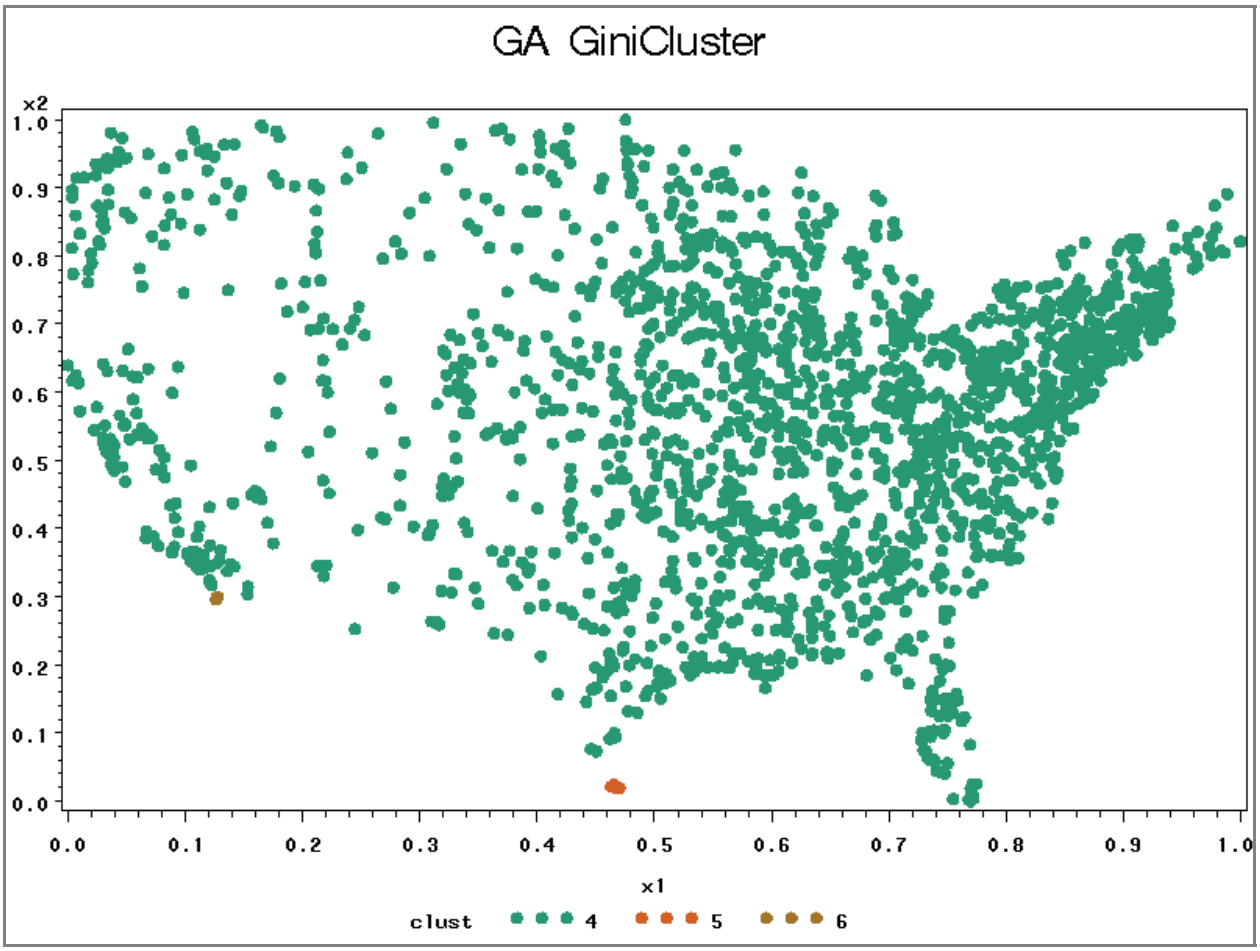
$$G = 1 - \sum_{i=1}^{8} (1/8)^2 = 0.875$$

Our results are not very different from this maximum heterogeneity, but of course there was no attempt to minimize this value.

The Genetic Algorithm function can be modified to optimize the clustering measured by Gini Index. The observations are still assigned to the clusters based on the Euclidian distance. However, p_i is calculated for each cluster and the returned value of the analyze function is now the Gini Index. Therefore, the genetic algorithm will try to minimize the Gini Index while still assigning to clusters based on distance.

```
function analyze(selected[*],x1[*],x2[*], event[*], nbrClusters, nbrObs, iter);
…
do k = 1 to nbrObs;
    dist = 999999;
    clust = 0;
    do i = 1 to nbrClusters;
           test = sqrt((a[(i*2)-1]-x1[k])**2 + (a[i*2]-x2[k])**2);
           if (test < dist) then do;
                  dist = test;
                  clust = i;
           end;
    end; *i;
    if (event[k] = 1) then events[clust] = events[clust]+1;
end; *k;
Gini = 1;
do n = 1 to nbrClusters;
    p_i2 = (events[n]/&nbrEvents)**2;
    Gini = Gini - p_i2;
end;
return(Gini);
```

The results of this approach are less than satisfactory.



Segment	p_i FASTCLUS	p_i GA CLUSTER	p_i GINI CLUSTER
1	0.1185	0.1309	
2	0.1802	0.1926	
3	0.0617	0.2123	
4	0.1728	0.1111	1.0000
5	0.1160	0.0790	0.0000
6	0.1259	0.0543	0.0000
7	0.0667	0.0494	
8	0.1580	0.1704	
Gini Index	0.861	0.848	0.0000

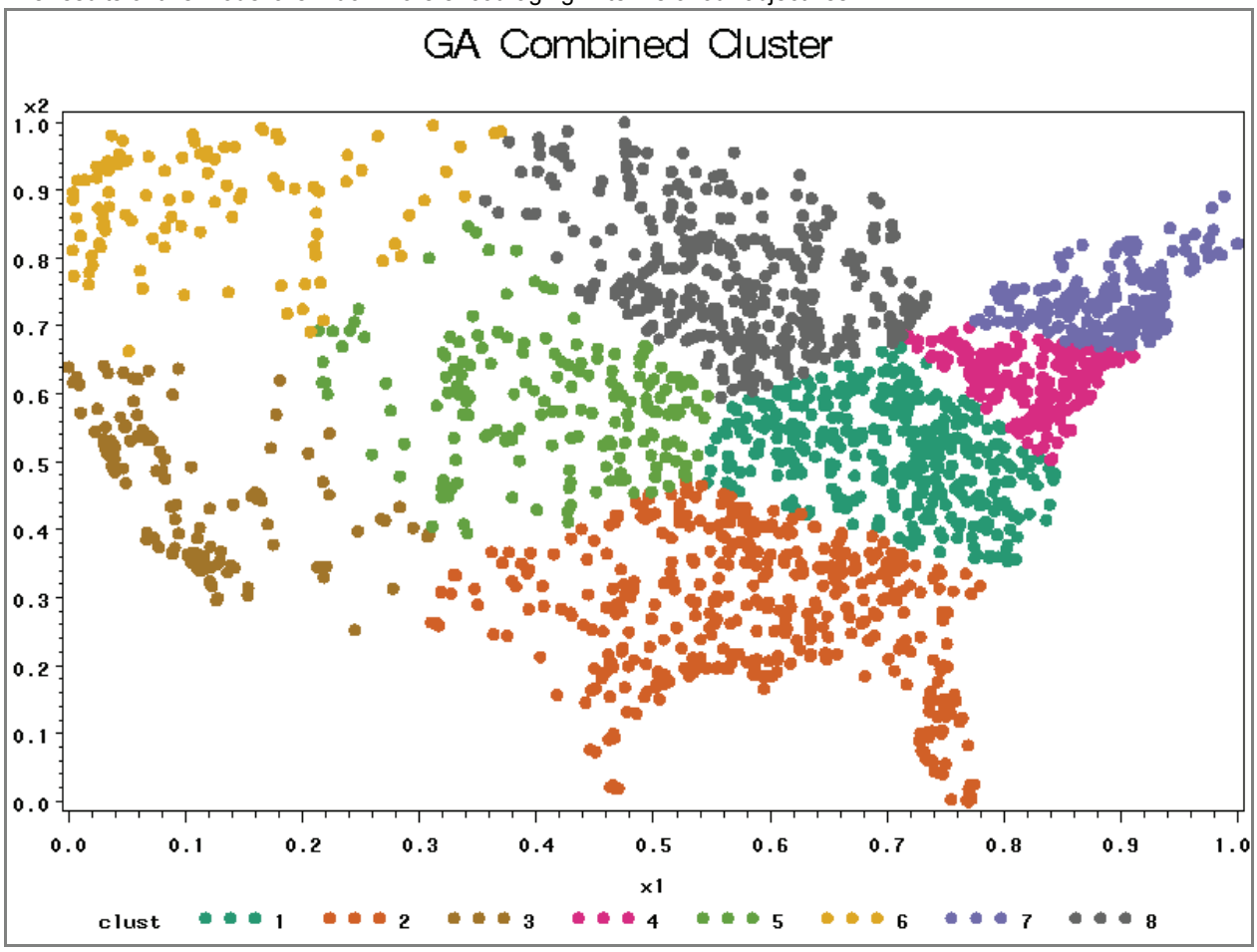
The algorithm clearly accomplished the goal, however the result is likely not useful. It should be noted that the Genetic Algorithm did find two unique spots where there are no events. Given these two groups are not contiguous this demonstrates the flexibility of the optimization.

COMBINING CONFLICTING REQUIREMENTS

The simultaneous goals of having the events clustered together while minimizing distance from midpoints of the clusters are contradictory objectives. This may be accomplished by modifying the objective function of the Genetic Algorithm using a weighted contribution of both goals. In practice the weights applied would be driven by specific business needs or costs. The following line of code gives an example with a weighting scheme of the two objectives.

```
return(0.2*Gini + sum/nbrObs);
```

The results of this model are much more encouraging in terms of our objectives.



Segment	p_i FASTCLUS	p_i GA CLUSTER	p_i GINI CLUSTER	p_i COMBINED
1	0.1185	0.1309		0.1605
2	0.1802	0.1926		0.2988
3	0.0617	0.2123		0.0519
4	0.1728	0.1111	1.0000	0.0667
5	0.1160	0.0790	0.0000	0.1210
6	0.1259	0.0543	0.0000	0.0543
7	0.0667	0.0494		0.0568
8	0.1580	0.1704		0.1901
Gini Index	0.861	0.848	0.000	0.821

This improvement in the Gini Index puts almost 30% of the events in one segment with little penalty to the best average distance found by the FASTCLUS procedure.

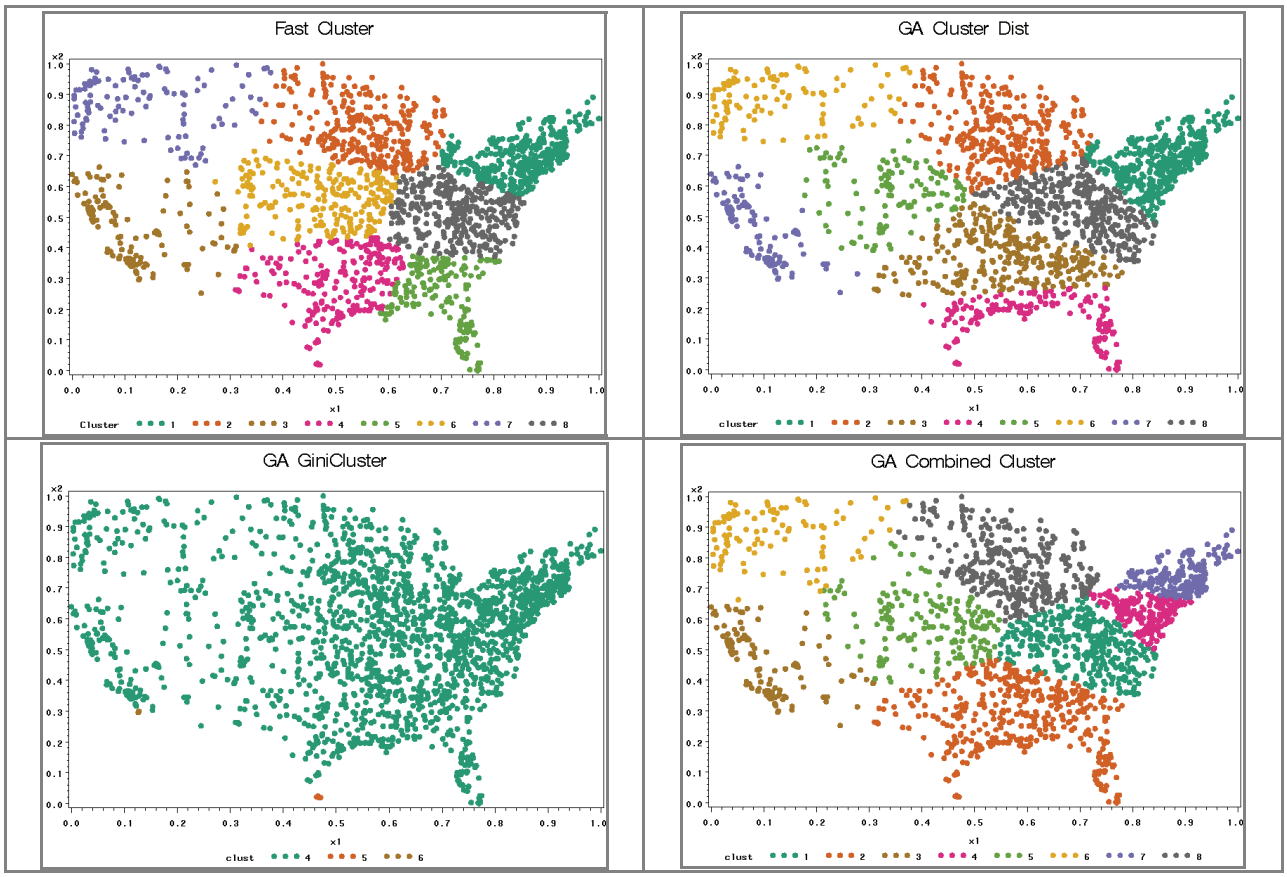
Segment	Avg. Distance Fastclus	Avg. Distance GA	Avg. Distance Combined
1	0.07902	0.08230	0.0926
2	0.10439	0.10828	0.1391
3	0.11677	0.11256	0.1097
4	0.10897	0.12258	0.0557
5	0.10735	0.10534	0.1122
6	0.10502	0.11536	0.1132
7	0.12541	0.10387	0.0576
8	0.08682	0.09535	0.1096
Total	0.09858	0.10145	0.10126

CONCLUSION

Customers are complex, markets are complex and analysts are assigned segmentation problems because of this complexity. Trying to simplify a market with a segmentation scheme – however necessary – is an unfortunate oversimplification of this complexity. The simple act of segmentation is adding enough oversimplification. Adding the limitation of minimizing only one objective such as minimizing Euclidian distance makes this oversimplification even worse. The analyst needs to recognize that the segmentation scheme needs to be driven only by business needs, not analytical constraints.

The above example is simplistic, but it demonstrates the ability to solve two factors simultaneously. The key point to gain from this exercise is that with the Genetic Algorithm any and all market or business needs can be incorporated into a segmentation scheme, no matter how complex. The only limitation is that the objective must be quantified.

The downside of Genetic Algorithm approaches is that they are very computationally inefficient. However anyone who has participated in the development of a complex segmentation scheme recognizes the extensive manpower required to develop such a scheme. Compared to this cost, computing power is very inexpensive.


Fast Cluster
GA Cluster Dist
GA GiniCluster
GA Combined Cluster
x2
x1
Cluster
cluster
clust