Automatic Clustering with Self-Organizing Maps and Genetic Algorithms

Angel Fernando Kuri-Morales
Instituto Tecnológico Autónomo de México
Río Hondo No. 1, México D.F.

Abstract. The analysis of data sets of unknown characteristics usually demands that subsets (or clusters) of the data are identified in such a way that the members of any one such cluster display common (in some sense) characteristics. In order to do this we must determine a) The number of clusters, b) The clusters themselves and c) The labeling of every element in the data set such that each element belongs uniquely to one of the clusters. We discuss an algorithm which allows us to solve (b) and (c); we assume that (a) is given. We show that the so-called labeling problem may be solved by minimizing an adequate measure of distance. We discuss several such metrics, the corresponding minimization (genetic) algorithm and offer some results derived from its application.

Keywords: Clustering, neural networks, genetic algorithms, metrics, optimization.

1 Introduction

There are several ways to attempt the identification of clusters in a set of data [1], [2], [3]. If we have information regarding the source of the data we may apply classical and/or heuristic methods with relative success [4]. Here, however, we assume that nothing is known about the data under study and apply the method originally proposed by Kohonen [5] which originates the so-called self-organizing maps (SOM). In this method, basically, a set of vectors (or "neurons") η is defined. The cardinality of η ($|\eta|$) is typically smaller than that of the objects in δ (the data set) i.e. $|\eta| \leq |\delta|$. The dimensionality of every vector in δ is determined by the number of features (φ) of each object and every such object is, thus, defined in a $\varphi - dimensional$ space. The neurons in a self-organizing map are simultaneously defined on two spaces: a) A $\varphi - dimensional$ space of features and b) A "geographic" map of γ dimensions (typically $\gamma = 2$ or $\gamma = 3$). The training algorithm then operates on the neurons in a way such that neighboring neurons in γ space (hence the name "SOM") correspond to elements which share some (possibly non-linear) attributes in δ space. Throughout this process it is relatively simple to overcome *a priori* limitations present in methods which rely on classical measures of distance (such as Euclidean or Mahalanobis' [6]). Once a set of neurons is trained (i.e. once its coordinates in δ space are determined), however, one is faced with the problem of finding the boundaries between the neurons in a SOM which distinguish one cluster from another. A simple example will illustrate this fact. Assume that $|\delta| = 200$, $|\varphi| = 4$, $\eta = 16$, $\gamma = 2$ and that the (known) number of classes (C) is 3. Let us further assume that through some yet unspecified method the neurons corresponding to each of these 3 classes has been found, yielding a map as in Figure 1. Notice that, by definition, all neurons for C = *i* (*i* = 1, 2, 3) are "physically" close in a euclidean sense. The process of assigning a label to every group of clustered neurons usually consists of setting a class number for every neuron from a previously known classification. In our example the data would assume a form analogous to the one shown in Table 1. In this table the heading "F1" to "F4" denote *feature* 1 to 4; class 1 consists of *k-1* elements, class 2 of *m-k* and class 3 of *200-m* elements, respectively.

There are alternative ways of displaying class membership but we will adhere to the one illustrated. When the C_i columns are known, other unknown elements which stem from the same source (H) giving rise to δ may be classified successfully, as has been shown in the past [7].

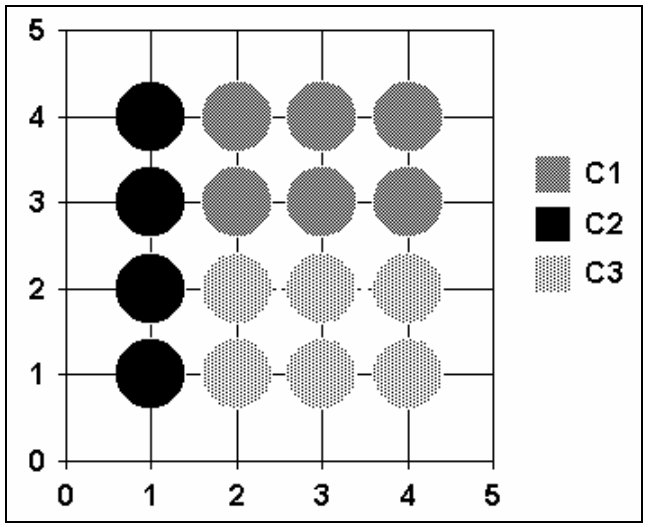

Fig. 1. A Labeled SOM with 3 Classes

F1	F2	F3	F4	C_1	C_2	C_3	
φ_{11}	φ_{12}	φ_{13}	φ_{14}	1	0	0	1
φ_{21}	φ_{22}	φ_{23}	φ_{24}	1	0	0	
...	...	...	...	1	...	...	k-1
φ_{k1}	φ_{k2}	φ_{k3}	φ_{k4}	0	1	0	k
$\varphi_{k+1,1}$	$\varphi_{k+1,2}$	$\varphi_{k+1,3}$	$\varphi_{k+1,4}$	0	1	0	
...	...	...	...	...	1	...	m-1
φ_{m1}	φ_{m2}	φ_{m3}	φ_{m4}	0	0	1	m
$\varphi_{m+1,1}$	$\varphi_{m+1,2}$	$\varphi_{m+1,3}$	$\varphi_{m+1,4}$	0	0	1	
...	...	...	...	...	...	1	200

Table 1. Labeling Data

In the absence of the C_i columns above, one must resort to some method which assigns class membership to every element in δ, so as to achieve a tabular structure similar to the one of Table 1. In the example above there are 3^{20} (roughly equivalent to 10^{90}) possible assignments. Clearly, any exhaustive enumerative approach is out of the question. We will, therefore, appeal to a genetic algorithm (GA) to find an

approximation to the best possible assignment. However, before using any optimization method we must first define a measure of fitness of an assignment. One of the contributions of this paper is the identification of a family of adequate metrics.

The rest of the paper is organized as follows. In section 2 we describe the details of the training and labeling algorithms as well as the GA we used. In section 3 we describe several metrics we tested and how we arrived at the "best" one. In section 4 we describe the application of the method to a set of simple classification problems and the results we obtained. Finally, in section 5 we offer our conclusions and point out future lines of research.

2 Algorithms

In what follows we describe the three basic algorithms on which our method relies: a) The SOM training algorithm, b) The SOM labeling algorithm and c) Vasconcelos' GA. Our description is succint. The interested reader may see [8], [9], [10] for a more detailed description.

2.1 Training Algorithm

1. By convention we assume that $\gamma = 2$ and a grid of size $\sqrt{\eta} \times \sqrt{\eta}$. All the neurons are assumed to be in the points of a grid in γ space where both coordinates are positive integers. We also assume that the horizontal and vertical distance between adjacent neurons is 1, as illustrated in figure 2. Thus, the coordinates in γ space of all neurons correspond to integers between 1 and $\sqrt{\eta}$; i.e. (1,1), ... , ($\sqrt{\eta}$, $\sqrt{\eta}$).

2. All neurons are initially assigned random coordinates (typically between 0 and 1) in φ space. Thus, the coordinates in φ space for all neurons define a $\varphi -$ dimensional vector.

3. An epoch counter *n* is initialized ($n \leftarrow 1$). One epoch is the presentation of all $|\delta|$ objects to the network.

4. A learning rate $\alpha(n)$ for epoch *n* is defined. Typically the initial learning rate is close to 1.

5. A feedback function from neuron *i* to the *winning* neuron *k* in epoch *n* is defined as $r_{ik}(n) = exp(-\frac{d_{ik}^{\,2}}{\sigma(n)^2})$; where $\sigma(n)$ is the *learning radius* for epoch *n* and d_{ik} is the *euclidean distance* between neuron *i* and neuron *k*.

6. The learning radius is initialized. Typically, in the beginning, $\sigma(n)$ takes a value larger than 3.

7. A learning factor f_α is defined. It is assigned an initial value close to 1 (say 0.99).

8. A radial factor f_σ is defined. It is, likewise, assigned an initial value close to 1 (say 0.995).

9. An object counter *t* is initialized ($t \leftarrow 1$).

10. Training object *x(t)* is presented to the network.

11. The *winning neuron* is determined by calculating the euclidean distance between all neurons and *x(t)* in φ space.

12. All the vectors in φ space are modified as per equation (1).

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(n) \cdot r_{ik}(n) \cdot (x(t) - w_{ij}(t)) \qquad (1)$$

13. $t = \delta$?

Yes: $\alpha(n+1) = \alpha(n) \cdot f_\alpha$
$\sigma(n+1) = \sigma(n) \cdot f_\sigma$
$t \leftarrow 1$

No: $t \leftarrow t + 1$

14. Is a stopping criterion reached?

No: Go to step 10

Yes: End

It should be noted that the mapping between the $\varphi - dimensional$ space to the $\gamma - dimensional$ one occurs when $r_{ik}(n)$ is found since

$$r_{ik}(n) = exp(-\frac{d_{ik}^{\,2}}{\sigma(n)^2}) \qquad (2)$$

$$d_{ik} = \| i - k \| = \sqrt{(i_x - k_x)^2 + (i_y - k_y)^2} \qquad (3)$$

Two further considerations regarding this algorithm are in order. First, the exponential form of (2) resembles a normal distribution centered in d_{ik} with standard deviation $\sigma(n)$. This implies that the influence of the winning neuron affects, initially, all neurons in the network (hence a large initial value for $\sigma(n)$); thereafter the neighborhood size of the winning neuron is made progressively smaller. Second, since the learning and radial factors are updated as per (4),

$$\sigma(n) = f_\sigma^n \cdot \sigma_0 \qquad \alpha(n) = f_\alpha^n \cdot \alpha_0 \qquad (4)$$

the adjustment on the coordinates of the neurons is

dynamically done in a way that insures that, as training advances, the changes in the coordinates are more subtle every time the data is traversed.

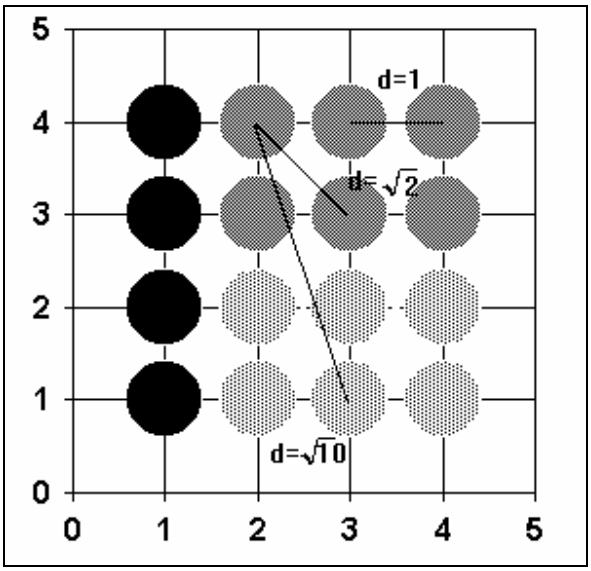

Fig. 2. Distance between Neurons in γ space

2.2 Labeling Algorithm

The algorithm just described guarantees that neurons which "point" to similar objects are close to each other in γ space. But, in order to exploit the information already contained in the network and apply it to new data (i.e. to attempt generalization), we must be able to identify the boundaries between the various groups of neurons in a way that establishes a clear distinction between the elements of a cluster and those of another. The separation that does exist between the elements of any two clusters may be derived from additional knowledge relating to elements whose membership is already known. One possible algorithm that has been used successfully in the past [11] is as follows.

Given N classes and a set of $|\delta|$ samples (objects) $\vec{O}_{jk}$ each one belonging to a class C_i, $i=1,...,N$, and making M=η for convenience, we define a matrix Δ of $M \times N$ elements. Denote the number of elements in class i with $|C_i|$. Clearly, $|\delta| = \sum_{i=1}^{N} |C_i|$.

Then,

1. Initialize all elements in $\vec{P}$ to 0.
2. For $i=1$ to N
3. For $j=1$ to M
4. For $k=1$ to $|C_j|$
5. Calculate the euclidean distance d_{ij} from neuron i $(\vec{w}_i)$ to object $\vec{O}_{jk}$:

$$d_{ij} = \sqrt{\left(O_{jk,1} - w_{i,1}\right)^2 + ... + \left(O_{jk,\varphi} - w_{i,\varphi}\right)^2} \quad (5)$$

6. Make

$$\Delta_{ij} \leftarrow \Delta_{ij} + 1/d_{ij} \quad (6)$$

7. endfor
8. endfor
9. endfor
10. Calculate

$$I_m = \max_{n=1,...,N} \{\Delta_{n,m}\} \quad for \quad m = 1,...,M \quad (7)$$

11. Assign the label of class I_m to neuron m.

□

This algorithm has two possible flaws: a) There may be no unique minimum and b) There may be classes for which no neuron is assigned. These two possible special cases are usually accounted for heuristically. For the purposes of the present discussion we shall assume that the special cases have already been satisfactorily taken care of.

A hypothetical distance matrix Δ, where M=*16* and *N=3* is shown in Figure 3.

i	j	Distance 01	Distance 02	Distance 03	Class
1	1	47.7334439790	67.1507545772	99.7589733212	3
1	2	58.2787719541	85.0103228778	79.5012882836	2
1	3	70.2683668785	105.9368680390	57.2402597886	2
1	4	69.0350546984	122.0529110845	47.7970304206	2
2	1	56.2852437939	85.5368987538	81.6109281364	2
2	2	66.5938453435	95.5350200626	67.4533501006	2
2	3	80.2785995629	108.2827110667	51.4568371983	2
2	4	77.5143303100	116.7603399523	42.2657038270	2
3	1	60.6853534276	103.4077812831	62.5830435923	2
3	2	78.9065176748	101.2960953722	56.3171749510	2
3	3	101.2399068548	96.9150402094	45.9936686770	1
3	4	110.4838966580	91.1756007678	39.9342358993	1
4	1	75.3003255408	100.8294298278	58.0891285854	2
4	2	92.6371008557	96.4477848199	49.0592256809	2
4	3	119.2449981339	84.8325461874	42.6541374145	1
4	4	133.2747783250	66.9797973986	36.1891922329	1

Fig. 3. An example of a distance matrix.

In Figure 3 we show the accumulated distances from each of the 16 neurons to the objects for each of the three classes. The last column displays the membership class according to a criterion of maximization, as per the algorithm above. The leftmost two columns (labeled "i" and "j") correspond to the coordinates of the neurons in

γ space.

2.3 Genetic Algorithm

Genetic Algorithms (GAs) are optimization meta-heuristics which differ from other optimization methods in two essential ways: a) They explore the solution landscape in several simultaneous loci and b) The algorithm explores the actual solution landscape but modifies its own behavior by changing a mapping of the space of encoded hypothetical solutions into the space of the actual solutions. They form part of what is now termed Evolutionary Computation. The "evolutionary" part of the name comes from the fact that the mentioned hypothetical solutions are refined step-by-step by preserving the most promising candidates from a so-called "population" which is, simply put, a set of viable solutions to the problem at hand (each solution in the set is an "individual" of the population). From a suggestive analogy with the living beings, the elements of the encoded solutions are called *genes* (hence the name "genetic"), the decoded solutions are the *phenotypes*, whilst their encoded counterparts are called the *genotypes*. GAs have been widely treated in the literature (see, for instance [12]) and we know that a) Elitist GAs will find the best solution given enough time, b) Such GAs may find solutions very close to the best solution in logarithmic time and c) Some "simple" problems may lead the classical GAs astray lest some modifications are introduced [13]. Previous studies [14] have statistically proven that the Vasconcelos' variation of a GA (VGA) performs very favorably and avoids the pitfalls present in more naive variations. Since the problem we are trying to tackle is of the NP kind, it is natural to use a VGA in an attempt to reach very good solutions in short time. In what follows we briefly describe the VGA.

2.3.1 VGA

1. a) $N \leftarrow 50$ (individuals in the population)
   b) $P_c \leftarrow 0.9$ (probability of crossover)
   c) $P_m \leftarrow 0.005$ (probability of mutation)
   d) $G \leftarrow 500$ (number of generations)
2. Calculate β (the number of bits to mutate) as:

$$\beta = ceiling[N \times |l| \times P_m]$$

where $|l|$ = bits in the individual's genome (throughout we assume binary encoding of the solutions).

3. *Generate* population *P(*1*)* randomly.
5. *For* $t \leftarrow 1$ *to* G
6. *Evaluate P(t).*
7. *Sort* the individuals in the population according to their fitness from best to worst.
8. *Retain* the best N individuals.
9. *For* $i \leftarrow 1$ *to* $N/2$
10. *Select* individuals i and N-i+1 (say A and Z).
11. *Cross* individuals A and Z with probability P_c. If A and Z are crossed, incorporate their offspring to the population.
12. *Endfor*
13. *Mutate* β bits (in the new individuals) randomly.
14. *Endfor*

□

In step 6, obviously, only the new individuals are evaluated. In step 11, crossover is *annular*. That is, individuals' chromosomes are considered not strings, but rather *rings* in the sense that the last bit in the genome is connected to the first. The size of the ring is $|l|/2$. Annular crossover is illustrated in Figure 4.

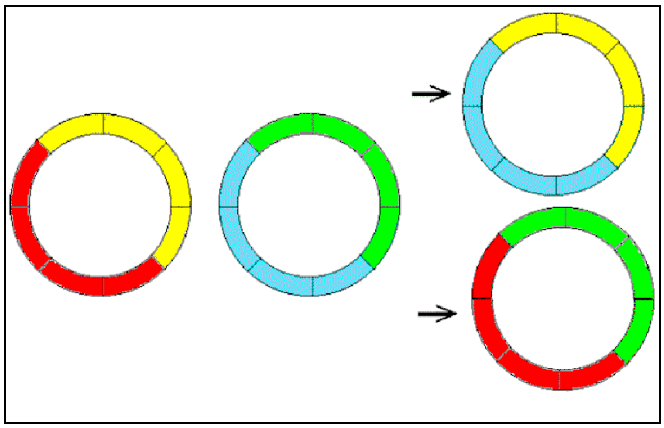

Fig. 4. Annular crossover.

3 Metrics

The problem that we would like to solve may be simply stated as follows: Given a table such as Table 1 *but lacking the data corresponding to the class columns*, what is the best way to fill in these columns such that adequate clustering is achieved? We may encode any assignment as a string of size δ consisting of a set of numbers between 1 and C. For example, if C=3 and δ = 20, the string *12312312312312332132* is a possible solution; the string *22211133333322211123* is another, and so on. What we would like to answer is which of the two solutions possible in our example is better? In order to do this we must define a measure of goodness: a

metric by which to judge the hypothetical solutions. From Figure 3 we see that we must select the column whose accumulated value is largest; from the labeling algorithm we see that the accumulated value comes from the distance between the objects in δ and the neurons. Hence, we proposed the following 4 metrics. In each, the first step is to calculate the distance matrix Δ, thus:

$$\Delta_{ij} = \sum_{l=1}^{|C(l)|} \left[1 / \sqrt{\sum_{k=1}^{\varphi} \left(w_{ik} - o_{ik} \right)^2} \right] \quad \begin{matrix} j = 1,\ldots, C \\ i = 1,\ldots, |\eta| \end{matrix} \tag{8}$$

3.1 Metric 1: Absolute Distance

$$D_1 = \sum_{i=1}^{|\eta|} \left\{ \sum_{j=1}^{C} \left| \begin{matrix} max(\Delta_{ij}) \\ j = 1,\ldots C \end{matrix} - \Delta_{ij} \right| \right\} \tag{9}$$

In D_1 we a) Select a row of Δ, b) Find its largest element, c) Get the absolute value of the difference from the largest element to the rest of the elements in the row, d) Add these differences. Repeat steps (a) to (d) for all neurons and accumulate the differences. We look for the largest possible D_1 resulting from the cluster assignment.

3.2 Metric 2: Euclidean Distance

$$D_2 = \sum_{i=1}^{|\eta|} \left\{ \sqrt{\sum_{j=1}^{C} \left(\begin{matrix} max(\Delta_{ij}) \\ j = 1,\ldots C \end{matrix} - \Delta_{ij} \right)^2} \right\} \tag{10}$$

In D_2 we a) Select a row of Δ, b) Find its largest element, c) Get the euclidean distance from the largest element to the rest of the elements in the row, d) Add these distances. Repeat steps (a) to (d) for all neurons and accumulate the distances. We look for the largest possible D_2 resulting from the cluster assignment.

3.3 Metric 3: Clustered Absolute Distance

$$D_3 = \sum_{i=1}^{C} \frac{1}{|C_i|} \left\{ \sum_{j=1}^{|\eta|} \left[\sum_{k=1}^{C} \left| \begin{matrix} max(\Delta_{jk}) \\ k = 1,\ldots C \end{matrix} - \Delta_{jk} \right| \right] \right\} \quad index[max(\Delta_{jk})] = i \tag{11}$$

In D_3 we follow a procedure similar to D_1 but we further establish that C partial values are calculated: one for every case in which the maximum value corresponds to a given cluster. Then, the average is obtained and the C partial values are finally added.

3.4 Metric 4: Clustered Euclidean Distance

$$D_4 = \sum_{i=1}^{C} \frac{1}{|C_i|} \left\{ \sum_{j=1}^{|\eta|} \left[\sum_{k=1}^{C} \sqrt{\left(\begin{matrix} max(\Delta_{jk}) \\ k = 1,\ldots C \end{matrix} - \Delta_{jk} \right)^2} \right] \right\} \quad index[max(\Delta_{jk})] = i \tag{12}$$

This metric is analogous to D_3, but the distance calculated is euclidean, rather than absolute.

□

The rationale behind all four ways of determining the adequacy of matrix Δ is similar: since labeling is achieved by selecting the largest value in Δ's rows, we reason in reverse and look for a cluster assignment such that the overall value is maximized.

3.5 Preliminary Results

Using known data we tested all four metrics. The clustered ones were found to be better than their counterparts. Furthermore, D_4's behavior originated smaller variances than D_3's and we considered, therefore, that D_4 was the best choice. We proceeded to test the GA in a way such that the populations consisted of binary strings. Our first runs yielded unsatisfactory results. The number of elements from δ which were correctly assigned to the original (known) clusters was not significant. We, therefore, modified D_4 by additionally requiring that there be no more than a pre-specified percentage (P) of elements in each cluster which exceeded proportionally equivalent cluster sizes. For example, if $|\delta| = 201$, C = 3 and P = 5, we would require that no more than 67 ± 3 elements be assigned to clusters 1, 2 and 3. We assume that, in a well conducted poll, it is reasonable to assume that samples will yield analogous cluster sizes. For the purposes of further discussion we refer to this modified metric as D_{405}; to the metric where P = 10 we refer as D_{410}; to either one of them we refer to as D_{4x}.

4 Experimental Results

As stated before, once the way of measuring a desirable assignment has been established, we may apply the VGA to obtain a reasonable approximation to the best such assignment. However, there are two further technical problems we must address.

4.1 Modifications to the Fitness Function

First, in a binary encoding, such as the one we selected, as a result of crossover and mutation, invalid genomes may arise. For example, if C = 5 and $|\delta| = 200$, the size of the genome Γ is 600 (which is derived from $\Gamma = |\delta| \times ceiling[\log_2(C)]$). However, all combinations for which $C \geq 5$ are invalid. To avoid this inconvenience, the fitness function was modified so as to detect any invalid combination and randomly replace it by a valid one. Second, usage of the D_{4x} metrics originates unfeasible genomes and finding the feasible ones which comply with the desired percentages constitutes, in itself, a thorny problem. Fortunately, a simple and effective scheme allows us to cope with it [18]. The procedure is to change a problem with constraints into an unconstrained one by applying the following transformation:

$$F(\Gamma) = \begin{cases} \left[K - \sum_{i=1}^{s} \frac{K}{p} \right] & s \neq p \\ D_{4x} & otherwise \end{cases} \qquad (13)$$

where $F(\Gamma)$ is the fitness of genome, K is a large constant [$O(10^9)$], p is the number of constraints and s is the number of these which have been satisfied. K's only restriction is that it should be large enough to insure that any non-feasible individual is graded much more poorly than any feasible one. Here the algorithm receives information as to how many constraints have been satisfied but is not otherwise affected. Once these two modifications are incorporated into the fitness calculation of VGA we are in a position to tackle the problem object of this work.

4.2 Experiments

We selected 5 functions whose characteristics were known in advance, i.e. we knew *a priori* the number of clusters into which the information was classified and which objects belonged to each of the clusters. We then run our algorithms (training and VGA). Finally, we compared the known clusters and memberships with the ones the VGA found. This was done with D_{410} and D_{405}. Table 2 shows the results from D_{405} and D_{410}.

Problem	Features	Samples	10.00%	5.00%
1	13	161	0.8419	0.7868
2	7	437	0.4766	0.4737
3	7	437	0.5948	0.8258
4	7	437	0.5763	0.5833
5	7	437	0.5571	0.7200

Table 2. Automatic Clustering Results

To appreciate the significance of the results we must analyze the type of data for every problem.

4.2.1 Problem 1

This data was taken from an actual physical sample of three different types of wines. Each wine is identified by 13 different chemical characteristics. Notice that the number of samples is the smallest of the series. There are 2^{322} or $\approx 8.54 \cdot 10^{96}$ possible configurations. The algorithm performed in the 80% range of hits.

4.2.2 Problem 2

This data was produced artificially: 6 random numbers were fed to three arbitrary functions. The seventh feature was the function of the remaining 6. The number of samples was 436. There are 2^{437} or $\approx 3.55 \cdot 10^{131}$ possible configurations. Considering the almost total randomness of the data and the huge problem size, it is to be expected that the algorithm does not perform satisfactorily.

4.2.3 Problem 3

This data was produced from the same set as the previous one. However, an important modification was introduced: every random number (between 0 and 1) was input to a trigonometric function and these values were fed to the functions. In this case, although the problem landscape is analogous to the one in 4.2.2, the behavior was significantly better. This is, of course, due to the fact that the restricted inputs defined distinguishable clusters.

For the 5% case, the algorithm performed on the 80% range of success.

4.2.4 Problem 4

In this set we, again, used the functions of 4.2.2-3. However, this time we explored the whole range of values of the trigonometric functions. Therefore, we re-assigned its randomness to the data and the clusters

remained unclear.

4.2.5 Problem 5

In this set we increased the generating functions while keeping well defined and restricted data for them. Again we obtained reasonable results: close to 72% of successes.

5 Conclusions

We have shown that it is possible to define reasonable measures of adequacy in arbitrary sets of data. Although our results are preliminary, they show an interesting promise. The algorithm, first, was tested to determine the viability of the approach. Secondly, we tried out four possible metrics and concluded that the clustered euclidean distance was the most promising. Thirdly, we discovered that even this preliminary best had to be complemented by a restriction leading to uniformity of the data. We assumed this to be a reasonable modification. Next we explored two differently stringent spreads. Our results show that the most stringent has better yields. Finally, by selecting different sets of clusters we were able to ascertain that the results are in accordance with our intuition: random data is unable to be a source of identifiable clusters. Finally, by importantly increasing the size of the genome, we were able to show that the method is applicable even in those cases where the solution landscape is quite large.

The foregoing conclusions largely depend on an efficient training algorithm and an efficient genetic algorithm. It is natural to continue our investigation in the following directions: a) Test the metrics on larger sets of functions with the aim of possibly refining these metrics; b) Cover a broader range of clusters; c) Expand the training methods. We plan to experiment with fuzzy Kohonen networks; d) Enhance the algorithm to determine, automatically, the number of clusters. Here we assumed such number was given.

We feel that this is a promising line of research. We have achieved the automatic assignment of elements of a data set to the correct unknown clusters even in the absence of information regarding the data. We believe that this will have important implications for data mining applications. It is also to be stressed that by merging two soft-computing methods (neural networks and genetic algorithms) we have shown their mutual flexibility and versatility.

References:

[1] Devijver, P.A. and Kittler, J., *Pattern Recognition: A Statistical Approach*, Prentice-Hall International, Englewood Cliffs, NJ, 1980.

[2] Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis,* Wiley-Interscience, New York, 1973.

[3] Fukunaga, K., *Introduction to Statistical Pattern Recognition, 2nd Ed.*, Academic Press, New York, 1990.

[4] Schalkoff, R., *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, New York, 1992.

[5] Kohonen, T., *Self-Organizing Maps*, Springer, Berlin, 2001.

[6] Duda, D., and Hart, P.E., *op. cit.*

[7] Kohonen, T., Barna, G. and Chrisley, R., Statistical Pattern Recognition with neural networks. Benchmarking studies, *IEEE International Conference on Neural Networks*, vol I, pp. 61-68, 1988, San Diego, CA.

[8] Haykin, S., *Neural Networks*: *A Comprehensive Foundation*, MacMillan, 1999.

[9] Kohonen, T., Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 1982, vol. 43, pp. 59-69.

[10] Kuri, A., "A Universal Eclectic Genetic Algorithm for Constrained Optimization", 1998, *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98*, pp. 518-522.

[11] MIT Staff, *Data Engine, from Data to Information*, MIT Press, 1999.

[12] Back, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[13] Kuri, A., *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning. Theory and Applications*, Vol. 1. Instituto Politécnico Nacional, pp 270, 1999.

[14] Kuri, A., A Methodology for the Statistical Characterization of Genetic Algorithms, *II Mexican International Congress on Artificial Intelligence*, submitted for publication.