Table of Contents

1. Unsupervised Niche Clustering: Discovering an Unknown Number of Clusters in Noisy Data Sets

Olfa Nasraoui[1], Elizabeth Leon[1], and Raghu Krishnapuram[2]

[1] Department of Electrical and Computer Engineering
The University of Memphis,
206 Engineering Science Bldg.,
Memphis, TN 38152-3180
email: onasraou@memphis.edu

[2] IBM India Research Lab
Block 1, Indian Institute of Technology,
Hauz Khas, New Delhi 110016, India
kraghura@in.ibm.com

Summary.

As a valuable unsupervised learning tool, clustering is crucial to many applications in pattern recognition, machine learning, and data mining. Evolutionary techniques have been used with success as global searchers in difficult problems, particularly in the optimization of non-differentiable functions. Hence, they can improve clustering. However, existing *evolutionary* clustering techniques suffer from one or more of the following shortcomings: **(i)** they are *not robust* in the presence of noise, **(ii)** they assume a *known* number of clusters, and **(iii)** the size of the search space *explodes exponentially* with the number of clusters, or with the number of data points. We present a *robust* clustering algorithm, called the *Unsupervised Niche Clustering algorithm (UNC)*, that overcomes all the above difficulties. UNC can successfully find dense areas (clusters) in feature space and determines the *number* of clusters *automatically*. The clustering problem is converted to a multimodal function optimization problem within the context of Genetic Niching. Robust cluster scale estimates are *dynamically* estimated using a hybrid learning scheme coupled with the genetic optimization of the cluster centers, to adapt to clusters of different sizes and noise contamination rates. Genetic Optimization enables our approach to handle data with both numeric and qualitative attributes, and general *subjective, non metric, even non-differentiable* dissimilarity measures.

1.1 Introduction

Clustering [DH73, Fuk90, AH96] is an effective technique for data mining and exploratory data analysis that aims at classifying the unlabeled points in a data set into different groups or clusters, such that members of the same cluster are as similar as possible, while members of different clusters are as dissimilar as possible. Several approaches to clustering exist. For example, graph-theoretic and tree-based techniques are popular in the machine

learning community; while most objective function-driven or prototype-based clustering methods such as the $K-$Means [Mac67], its fuzzy counterpart, the Fuzzy C-Means (FCM) [Rus69, Dun74, Bez81], and Gaussian Mixture modeling, have long been used in statistical pattern recognition. Recently, data mining has put even higher demands on clustering algorithms. They now must handle very large data sets, leading to many scalable clustering techniques. Examples are CLARANS [NH94] and BIRCH [ZRL96], which assume that clusters are hyper-spherical, similar in size, and span the whole data space. Other techniques include CURE [GRS98] and the scalable K-Means and the scalable EM [BFR98a, BFR98b]. Unfortunately, all the above techniques were not designed to handle large and unknown amounts of noise in the data. *Robust* clustering techniques have recently been proposed to handle noisy data. Another limitation of most clustering algorithms is that they assume that the number of clusters is known.

1.1.1 Unsupervised Clustering

When the number of clusters is not known, the situation is sometimes called *unsupervised clustering*. Traditionally, unsupervised clustering has relied on three different approaches. The first approach is to evaluate the validity of the partition generated by the clustering for several values of number of clusters, c, and then accept the partition with optimal validity. The second approach is to seek and remove one cluster at a time provided that the found cluster passes a validity test [JMB91]. The problem with these approaches lies in the difficulty in designing validity measures that perform well on a variety of data sets encountered in practice. Also, most validity measures either assume a known underlying inlier or noise distribution or are very sensitive to noise, and hence are not appropriate for general robust clustering. The third approach consists of starting the clustering process with an overspecified number of clusters, and then merging similar clusters and eliminating spurious clusters until the correct number of clusters is left as in Compatible Cluster Merging [KF92].

1.1.2 Motivations for Evolutionary Clustering

Inspired by nature, Genetic Algorithms (GAs) [Hol75] and evolutionary strategies [FOW66] constitute a powerful set of global search techniques that have demonstrated good performance on a wide variety of problems. GAs search the solution space of a fitness function to be optimized using a simulated "Darwinian" evolution that favors survival of the fittest. Whereas traditional search techniques rely on characteristics of the objective function to be optimized (such as differentiability for gradients and Hessians, and sometimes linearity and continuity) to determine the next sampling point, GAs make no such assumption. Instead these points are determined based on stochastic

sampling rules. This means that GAs can optimize fitness functions of many forms, subject to the minimal requirement that the function can map the population into a partially ordered set [Gol89].

The Simple Genetic Algorithm (SGA) has been successfully used to search the solution space in clustering problems with a *fixed* number of clusters [HOB99], and for robust clustering [NK97]. However, in practice, the number of clusters may not be known.

1.1.3 Why Genetic Niching?

There is a *symbiosis* between the way *niches* evolve in nature and the way data is partitioned into optimal *clusters*. In the Evolutionary Computation area, Genetic Niching (GN) techniques have emerged, largely inspired by nature, to solve the multimodal function optimization problem and to *counteract premature population convergence* by encouraging genetic diversity. In unsupervised learning, a wide variety of clustering techniques have been developed to *cluster heterogenous data into different components*.

We propose a novel approach to unsupervised robust clustering using Genetic Algorithms. We start by modifying our objective from searching the solution space for c clusters to searching this space for any one cluster. Accordingly, we need to optimize an appropriate objective function that simply measures the goodness of fit of a model to just part of the data. We formulate a density based fitness function that reaches a maximum at every good cluster center. This means that the fitness landscape will consist of multiple peaks with one at each cluster location. We use Deterministic Crowding (DC) [Mah92] as the genetic niching optimization tool. To alleviate the problem of crossover interaction between distinct niches, we propose an improved restricted mating scheme which relies on an accurate and assumption-free estimate of the niche radii which are not restricted to be equal for all peaks. The resulting unsupervised clustering algorithm is called *Unsupervised Niche Clustering* (UNC). Since UNC is based on Genetic Optimization, it is much less prone to suboptimal solutions than traditional techniques. Moreover, because UNC uses robust weights, it is less sensitive to the presence of noise. Also, relying on Genetic optimization to search for the solution obviates the need to analytically derive the update equations for any prototypes. Thus, our approach is able to handle data with both numeric and qualitative attributes, and it can work with general subjective non metric dissimilarity measures. Handling such dissimilarities has traditionally been tackled by clustering relational data consisting of all pairwise dissimilarity values. In this case, the complexity of most algorithms, such as the Linkage type hierarchical clustering techniques can be prohibitively expensive (in the order of $\mathcal{O}\left(N^2 \log_2 (N)\right)$).

The remainder of this chapter is organized as follows. In Section 1.2, we review niching methods. In Section 1.3, we survey some evolutionary clustering techniques. In Section 1.4, we describe our new approach to unsupervised

clustering based on genetic niching. In Section 1.5, we present our synthetic experimental results. In Section 1.6, we apply our unsupervised clustering algorithm, UNC, to the problem of image segmentation. Finally, we present our conclusions in Section 1.7.

1.2 Genetic Niching

Traditional Genetic Algorithms have proved effective in exploring complicated fitness landscapes and converging populations of candidate solutions to a single global optimum. However, some optimization problems require the identification of global as well as local optima in a multimodal domain. As a result, several population diversity mechanisms have been proposed to delay or counteract the convergence of the population to a single solution by maintaining a diverse population of members throughout its search. As for the case of GAs, these diversity enhancing methods have turned to nature for ideas and analogies. An analogy to multimodal domains exists in nature in the form of "ecological niches" which are subspaces that can support different types of species or organisms. This has inspired the consideration of each peak in a multimodal domain as a niche in the framework of what has come to be called niche formation methods. In nature, the fertility of the niche as well as the efficiency of each organism at exploiting that fertility is what determines the number of organisms that can be contained in a niche. This principle is at the base of how a GA should maintain the population diversity of its members in a multimodal domain. Thus, the niches should be populated in proportion to their fitness relative to other peaks. This concept is known as niche proportionate population. This aim has been very difficult to realize because of the uncertainty in the niche location (peak *location* and niche *boundary*). The two most popular niche formation methods are sharing and crowding.

Sharing methods [Hol75, GR87] attempt to maintain a diverse population by reducing the fitness of individuals that have highly similar members within the population. This in turn discourages redundant solutions from overtaking the entire population, while rewarding individuals that uniquely exploit specific areas of the domain. Sharing methods rely strongly on correct estimates of the niche counts, i. e., the number of individuals in each niche. The niche counts themselves depend on a parameter, σ_{sh}, which ideally, should approximate the widths of the peaks. Deb and Goldberg [DG89] suggested ways of determining the appropriate value for σ_{sh} based on the expected number of peaks and the hypervolume of the entire domain space. Unfortunately, in many real applications, the number of peaks may not be known. Moreover, a single value of σ_{sh} may not be sufficient when peaks differ vastly in their widths. In addition to the above drawbacks, sharing adds an additional $\mathcal{O}(N_P^2)$ complexity per generation to compute all the pairwise distance calculations among all the members of a population of size N_P.

Crowding methods, proposed by De Jong in [Jon75], try to form and maintain niches by replacing population members preferrably with the most similar individuals. Unfortunately, stochastic "replacement errors" prevented this method from maintaining more than two peaks in a multimodal fitness landscape. Mahfoud [Mah92] proposed an improved crowding mechanism, called "deterministic crowding" (DC), which nearly eliminated replacement errors and proved more effective in maintaining multiple niches. DC is presented below:

Deterministic Crowding (DC)

Repeat for G **generations** {
  Repeat $\frac{N_P}{2}$ **times** {
    Select two parents p_1 and p_2 randomly without replacement;
    Cross them to produce children c_1 and c_2;
    Optionally apply mutation to produce children c_1' and c_2';
    IF $[d(p_1, c_1') + d(p_2, c_2')] \leq [d(p_1, c_2') + d(p_2, c_1')]$ THEN {
      IF $f(c_1') > f(p_1)$ THEN replace p_1 with c_1'
      IF $f(c_2') > f(p_2)$ THEN replace p_2 with c_2'
    }
    ELSE {
      IF $f(c_2') > f(p_1)$ THEN replace p_1 with c_2'
      IF $f(c_1') > f(p_2)$ THEN replace p_2 with c_1'
    }
  }
}

Unlike sharing methods, DC is free of any parameters relying on assumptions about the number of peaks or their widths. Also, unlike sharing, the expected distribution of the population at convergence is independent of fitness. Instead, the cardinality of each niche is expected to be proportional to the fraction of the population from the search space that falls within this niche, or in other words the prior probability of the niche. However, a dominated niche can, with another niche's assistance, cross to form members from a fitter (dominant) niche. This causes a migration of members from the dominated peaks to the dominant peaks that will only come to a halt when one of the dominated niches is depleted of its members. This crossover interaction problem can be critical for clustering because of the arbitrary density and position of clusters.

1.3 An Overview of Existing Evolutionary Clustering Techniques

Clustering techniques which use mathematial optimization methods or local search strategies are very sensitive to initialization, and can be trapped in local extrema. GAs perform a globalized search for solutions that can alleviate this problem. One of the earliest attempts at using a GA for clustering

was made by Raghavan and Birchand [RB79]. In their approach, the GA was used to optimize the square error of clustering (similar to K Means's criterion). Each chromosome represented a possible partition of the entire data set consisting of N objects. Hence the chromosome consisted of N substrings, with each substring encoding one of c cluster labels. Obviously this encoding led to an explosion of the search space size as the data set got larger, and assumed a known number of clusters. The square error based fitness function also meant that the approach was sensitive to noise. Most importantly, the encoding scheme was poor, since n-point crossover frequently resulted in meaningless or lethal partitions. Bhuyan et al. [BRV91] proposed an improved encoding that used a separator symbol ($*$) to separate the clusters, consisting of a string of data object labels, and Goldberg's permutation crossover [Gol89] to yield valid offspring. However, the solution suffered from en explosion in permutation redundancy because of the arbitrary order of data labels. Babu and Murty [BM93] used the GA only to find good initial solutions, and then used K-Means for the final partition. This was the first hybrid clustering approach that obviously outperformed the use of either K-means or GA alone. However, it is not resistant to noise, and assumed a known number of clusters. Fogel and Simpson [FS93] use Evolutionary Programming to solve the fuzzy min-max cluster problem by directly encoding the centroids (instead of the entire partition) in the chromosome string. This was undoubtedly one of the first *efficient* encodings for the clustering problem, and it influenced most subsequent evolutionary clustering methods. Hall et al. [BBHB94, HOB99] proposed a genetically guided approach (GGA) to optimizing the reformulated Hard and Fuzzy C-Means (HCM and FCM) objective functions. The chromosome strings encode the c center vectors (of p individual features per vector) of the candidate solutions, and a standard GA evolves the population. However, GGA is *not robust* in the face of noise, and it *can not determine the number of clusters automatically.* Also, because all c cluster centers are encoded in each chromosome string, the size of the search space *explodes exponentially with the number of clusters.*

In [NK97], we proposed a robust estimator based on the LMedS that can simultaneously partition a given data set into c clusters, and estimate their parameters, for the case when the number of clusters, c is known a priori. In addition to its limitation of estimating the parameters of a single structure, the LMedS suffers from a major drawback in that it has a nonlinear, and nondifferentiable objective function that is not amenable to mathematical or numerical optimization. For this reason, we proposed the integration of a genetic algorithm to the partitioning and estimation process, in order to search the solution space more efficiently. This resulted in a new approach to *robust genetic clustering* based on LMedS. However, the technique still assumed that the number of clusters was *known in advance*, and that the *noise contamination rate was* 50%. Also, because all c cluster centers are

encoded in each chromosome string, the size of the search space explodes exponentionally with the number of clusters as in the case of GGA.

Lee and Antonsson present an algorithm for unsupervised clustering using Evolutionary Strategies (ES) in [LA00]. In their approach, all cluster centroids are coded into a variable length chromosome, and only crossover is used to vary the number of clusters. However this coding scheme suffers from an exponential increase in the complexity of search with the number of clusters, and is also not robust to noise and outliers because its fitness measure is based on a classical sum of errors.

Rousseeuw's original robust K-Medoid criterion [RL87] was optimized in [ECY00] using a hybrid GA approach. Though more robust to noise, this approach assumes a known number of clusters. Also, because the cluster representatives are medians, it is most efficient when the rate of noise is exactly 50%, and cannot adapt to various noise contamination rates. It also does not have any provision for clusters of different sizes since it has no notion of scale, and the size of the search space explodes exponentionally with the number of clusters.

In [NK00], we presented preliminary results of Unsupervised Niche Clustering (UNC). In this chapter, we present detailed derivations and discussions of the theoretical properties of UNC, as well as thorough evaluation experiments.

Table 1.1 compares some evolutionary clustering techniques including UNC with respect to important features such as robustness to noise, automatic determination of the number of clusters, $\cdots$, etc, showing that UNC has all the desired features lacking in other approaches. Note that a *partitional* approach relies on a sum of error type objective function that requires encoding of c cluster centers or a long c-ary partition string of length N. Partitional approaches also require the additional costly overhead of *re-partitioning the data points into c clusters with each fitness computation.* On the other hand, a *density* based approach directly optimizes each cluster density independently of other clusters, hence eliminating the need to partition data. Density is also a more sensible measure of cluster validity, and is naturally resistant to noise. Complexity is listed per generation. Also hybrid approaches tend to converge in fewer generations compared to purely evolutionary search methods, and are therefeore faster.

1.4 New Approach to Unsupervised Robust Clustering using Genetic Niching

1.4.1 Representation

The solution space for possible cluster centers consists of n-dimensional prototype vectors. These are represented by concatenating the Gray codes of the

Table 1.1. Comparison of UNC with Other Evolutionary Clustering Algorithms for data of size N, population of size N_P, and c clusters

Approach →	*UNC*	*GGA* [HOB99]	*Lee* [LA00]	*G-C-LMedS* [NK97]	*k-d-Median* [ECY00]
Search Method	GA	GA	ES	GA	GA
Robustness to noise	yes	no	no	yes	yes
Automatic Scale Estimation	yes	no	no	no	no
Complexity: $O()$	NN_P	CNN_P	CNN_P	CNN_P	$N_P CN \log(N)$
Hybrid	yes	no	no	no	yes
Does not require No. of Clusters	yes	no	yes	no	no
Handles ellipsoidal clusters	yes	no	no	no	no
Density/Partition	Density	Partition	Partition	Partition	Partition

individual features for *a single* cluster center into a binary string. Paradoxically, this means that the search space is much smaller than the one corresponding to using the SGA to search for c cluster centers as in the case of the Fuzzy C-Means based GGA [BBHB94] and the Genetic C-LMedS [NK97]. If S is the search space size for our niching based unsupervised approach to clustering, then the search space size for the SGA based C-means clustering is S^c. As expected, the savings in the size of the search space translate into savings in the population size.

1.4.2 Fitness Function

Since in general, we identify dense areas of a feature space as clusters, we will define the fitness value, f_i, for a candidate center location, $\mathbf{c}_i$, as the density of a hypothetical cluster at that location. For the case of an n-dimensional data set, $\mathcal{X}$, with N data points, $\mathbf{x}_j, j = 1, \cdots, N$ the density of the i^{th} cluster can be defined as follows

$$f_i = \frac{\sum_{j=1}^{N} w_{ij}}{\sigma_i^2}, \tag{1.1}$$

where

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma_i^2}\right), \tag{1.2}$$

is a *robust* weight designed to be high for inliers (typical points falling within the boundaries of a cluster) and low for outliers (points that fall beyond the cluster boundaries), and d_{ij}^2 can be any distance measure, such as the Euclidean distance from data point $\mathbf{x}_j$ to cluster center $\mathbf{c}_i$,

$$d_{ij}^2 = \|\mathbf{x}_j - \mathbf{c}_i\|^2 = (\mathbf{x}_j - \mathbf{c}_i)^t (\mathbf{x}_j - \mathbf{c}_i). \tag{1.3}$$

σ_i^2 is a robust measure of scale (dispersion) for the i^{th} cluster, which is crucial for determining its boundaries. In fact, the niche radius can be written as $K\sigma_i^2$, where K is close to $\chi^2_{n,.995}$ for spherical Gaussian clusters.

For the case of general multivariate hyper-ellipsoidal clusters, the fitness measure is evaluated as

$$f_i = \frac{\sum_{j=1}^{N} w_{ij}}{|\Sigma_i|^{\frac{1}{n}}}, \tag{1.4}$$

where

$$w_{ij} = \exp\left(-\frac{d_{Mij}^2}{2}\right), \tag{1.5}$$

In (1.4), $|.|$ is the determinent and Σ_i is a dynamically estimated dispersion matrix, which is used to compute the Mahalanobis distance values that serve to compute the weights in (1.5). That is

$$d_{Mij}^2 = (\mathbf{x}_j - \mathbf{c}_i)^t \Sigma_i^{-1} (\mathbf{x}_j - \mathbf{c}_i), \tag{1.6}$$

It is important to observe that for the case of a diagonal dispersion matrix where all the diagonal entries are equal, both the fitness and robust weight in (1.4) and (1.5) reduce to (1.1) and (1.2) respectively. Even though (1.4) is more general than (1.1), when the dimensionality is high, there are several benefits in estimating scalar dispersion factors as opposed to dispersion matrices and their required inversions for each individual of the population. These benefits clearly come in the form of computational savings, but also in the form of better estimates because of the fewer number of parameters that have to be estimated. For general high dimensional data sets with arbitrary distributions, it may be preferrable to assume a simpler model. Also, the simpler model with scalar dispersion allows more general dissimilarity measures to be used, even non-differentiable and non-metric ones. For this reason, we present the analysis and results of the approaches based on both the scalar and matrix dispersion models. For the case of $n = 2$, it can be seen that f_i is a measure of density since it measures the ratio of the robust cardinality, $\sum_{j=1}^{N} w_{ij}$, of the i^{th} cluster or niche to its volume. For the case of $n > 2$, the niche volume is proportional to $|\Sigma_i|^{\frac{1}{2}}$, Hence, the fitness is the density of the projection of the data points on a 2-dimensional subspace, where the resulting dispersion is proportional to the geometrical mean of the dispersions along the individual dimensions of the data set. In this case, σ_i^2 and $|\Sigma_i|^{\frac{1}{n}}$ become analogous, i.e., they are both approximately proportional to the square of the projected niche radius for the i^{th} cluster. In both cases, the normalization of the cardinality of the niche by its size gives preference to individuals located in denser niches.

1.4.3 Analogy between Density and Scale in Artificial Niches and Fertility and Barrier Constraints in Natural Niches

One way to dynamically estimate the niche sizes or scales is to compute the optimal scale values that will maximize the proposed density fitness functions above. This is because the scale that will maximize cluster or niche density corresponds to the optimal niche size that achieves the right balance between niche compactness (as measured by the within niche dispersion in the denominator) and niche count (as measured by the soft cardinality in the numerator). Minimizing the dispersion (denominator) *encourages proximity* of members of the same niche, hence promoting the establishment of *barriers* between distinct niches; while maximizing the soft cardinality in the numerator avoids the degenerate case that would result from optimizing compactness alone, and that can result in a niche accepting no more than a single member. Hence it encourages the *inclusion of as many members as possible* to populate the niche as long as enough resources are available. Hence, optimizing the above density fitness functions is consistent with the way that ecological niches are formed. They are areas that attract relatively higher members of the population compared to their surroundings.

1.4.4 A Baldwin Effect for Scale Estimation

The scale parameter or niche radius that maximizes the fitness value in (1.1), for the i^{th} cluster can be found by setting $\frac{\partial f_i}{\partial \sigma_i^2} = 0$ to obtain

$$\sigma_i^2 = \frac{1}{2} \frac{\sum_{j=1}^{N} w_{ij} d_{ij}^2}{\sum_{j=1}^{N} w_{ij}}. \tag{1.7}$$

The centers at which density is maximized are unbiased estimates of the cluster centroids(this can easily be verified by mathematical optimization of the fitness criterion). However, the scale values in (1.7) tend to be underestimated as seen for the simple case of a single cluster, in Figure 1.1(a).

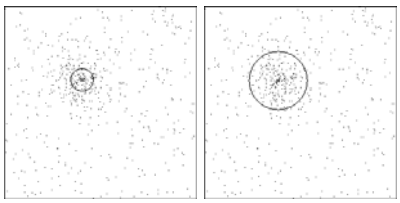


Fig. 1.1. Effect of bias adjustment on scale estimates:(a) results with a biased estimate (b) results with the compensated unbiased estimate

We compensate for this underestimation by adjusting the bias for the scale so that, if the statistical measure of variance, $\sigma_s^2 = E(\mathbf{x}_j - \mu_i)^2$, is considered as the reference, the bias would be zero (i.e., the estimate becomes unbiased). The adjustement of bias of σ_i^2 with respect to σ_s^2, denoted as $B_{\sigma_s^2}(\sigma_i^2)$ below, proceeds by adjusting the scale estimates σ_i^2 by a multiplying factor, α that will make its bias zero, as follows:

$$B_{\sigma_s^2}(\alpha\sigma_i^2) = E(\alpha\sigma_i^2) - \sigma_s^2 = E(\frac{\alpha}{2}\frac{\sum_{j=1}^N w_{ij}d_{ij}^2}{\sum_{j=1}^N w_{ij}}) - \sigma_s^2 = 0$$

Fixing the weights w_{ij} as was done before deriving the optimal scale values in each iteration, we obtain

$$B_{\sigma_s^2}(\alpha\sigma_i^2) = \frac{\alpha}{2}E(\sum_{j=1}^N \frac{w_{ij}}{\sum_{k=1}^N w_{ik}}d_{ij}^2) - \sigma_s^2 = \frac{\alpha}{2}\sum_{j=1}^N \frac{w_{ij}}{\sum_{k=1}^N w_{ik}}E(d_{ij}^2) - \sigma_s^2 = 0$$

$$B_{\sigma_s^2}(\alpha\sigma_i^2) = \frac{\alpha}{2}E(d_{ij}^2) - \sigma_s^2 = \frac{\alpha}{2}E(\mathbf{x}_j - \mu_i)^2 - \sigma_s^2 = 0$$

This reduces to

$$B_{\sigma_s^2}(\alpha\sigma_i^2) = \frac{\alpha}{2}E(\mathbf{x}_j - \mu_i)^2 - \sigma_s^2 = \frac{\alpha}{2}\sigma_s^2 - \sigma_s^2 = 0$$

which is satisfied iff.

$$\alpha = 2$$

Hence, the unbiased scale parameter or niche radius ($\alpha\sigma_i^2$) for the i^{th} cluster becomes

$$\sigma_i^2 = \frac{\sum_{j=1}^N w_{ij}d_{ij}^2}{\sum_{j=1}^N w_{ij}}. \tag{1.8}$$

Therefore, the σ_i^2 will be updated using an iterative hill-climbing procedure, using the previous values of σ_i^2 to compute the weights w_{ij} in (1.2). Note that this approximation is reasonable because by virtue of the replacement scheme in DC, the centers for an individual are not expected to change drastically from one generation to the next. Even though the exponential weights w_{ij} decrease with distance, points from other clusters can still exert an adverse influence on a given cluster's parameter estimates in the early generations, particularly when the data set contains a large number of clusters. For this reason, before updating the scale parameters, the weights will be mapped to a binary range. The resulting hard rejection will gradually allow only the core members of the cluster to be involved in the estimation of its parameters. The binarization of the weights is done as follows

$$w_{ij} = \begin{cases} 1 \text{ if } w_{ij} > T_w \\ 0 \text{ otherwise} \end{cases} \tag{1.9}$$

where T_w is a suitable threshold value.

The dispersion matrix that maximizes the fitness value in (1.4), for the i^{th} cluster can be found by setting $\frac{\partial f_i}{\partial \Sigma_i^{-1}} = 0$ and solving for Σ_i to obtain

$$\Sigma_i = \frac{\sum_{j=1}^{N} w_{ij}(x_j - c_i)(x_j - c_i)^T}{\sum_{j=1}^{N} w_{ij}}. \tag{1.10}$$

Therefore, when the fitness measure depends on a disperion matrix, the Σ_i will be updated using an iterative hill-climbing procedure, using the previous values of Σ_i to compute the weights w_{ij} in (1.5). As in the case of scalar dispersions, the weights are also binarized to counteract the initial influence of points from different niches.

The proposed approach to estimating the niche sizes makes our approach a hybrid genetic optimization technique. Unlike *Lamarckian learning* [WGM94], our approach to estimate the scale *does not disrupt the genotype* of the candidate solutions. However, it improves *individual learning* in the evolutionary process by dynamically modifying the *fitness landscape* (better scale estimates will result in more accurate cluster/niche density estimates) in a way that will make it easier to maintain diversity and to converge closer to the niche peaks. This can be seen as introducing a *Baldwin effect* [HW87] to the evolutionary process.

1.4.5 Mating Restriction

We impose some restrictions on the DC selection and replacement procedure described in Section 1.2 to prevent the production of lethal offspring or extinction of certain dominated niches. Let s_i denote a measure of the current scale estimate of individual P_i, where $s_i = \sigma_i^2$ if the scalar dispersion model is used and $s_i = |\Sigma_i|^{\frac{1}{n}}$ if the matrix dispersion model is used. Two population members P_i and P_j, with corresponding centers $\mathbf{c}_i$ and $\mathbf{c}_j$, are considered to be coming from different niches if neither one of them is within the other member's niche. Then P_i and P_j are considered to be from distinct niches if P_j is not within P_i's niche, or $dist(P_i, P_j) > Ks_i$; and P_i is not within P_j's niche, or $dist(P_i, P_j) > Ks_j$, where $dist\,(P_i, P_j) = \|\mathbf{c}_i - \mathbf{c}_j\|^2$. These two conditions are equivalent to $dist(P_i, P_j) > K \max(s_i, s_j)$. If parents from different niches are forbidden from mating regardless of their fitness values, then mediocre individuals located in non-niche areas of the search space will not be allowed to improve their genotype. As a result, these undesirable individuals which can result from the random initialization process, particularly when the data set contains outliers, will remain in the population until convergence. Therefore, mating restriction should be relaxed by allowing unconditional mating between two individuals if at least one of them has low

fitness. The resulting mating restriction rule for individuals P_i and P_j can be summarized as

$$\begin{aligned} &\text{IF } (dist\,(P_i, P_j) > K \max\,(s_i, s_j) \text{ and } f_i > f_{min} \text{ and } f_j > f_{min}) \\ &\text{THEN Restrict Mating.} \end{aligned} \tag{1.11}$$

1.4.6 Scale Inheritance

When mating takes place, each child inherits the scale parameter, σ_i^2, or dispersion matrix, Σ_i, of the closest parent as its initial scale. This inheritance is possible because in DC, children are most likely to replace similar parents. This makes it possible to integrate a hill-climbing step that encompasses all generations for the scale parameter updates. Our *one-step* local Baldwin Effect learning approach, enabled by a cumulative *memory* of past generations, is more efficient than traditional hybridizations of GA and hill-climbing procedures because the latter iteratively update the parameters within each generation. However, these updates cannot be used as starting points for the next generation due to arbitrary changes in the population distribution between generations, particularly in the early generations (no memory).

The *full* or *from-scratch* learning of existing hybrid approaches, while making evolution faster (i.e., fewer generations are needed to converge), suffers from an added heavy computational cost in *each* generation for *each* individual. Our approach has the advantage of not requiring more than a single step of updating the scales, and hence adapting the environment (fitness) to accelerate evolution. Following their inheritance by both children, the scale parameters are updated using (1.8). Similarly, the parents' scale parameters are updated so that both children and parents' scale parameters would have implicitly undergone the same number of updates starting from the initial generation. This is essential to make a fair comparison of their fitnesses for the replacement decision in DC. When mating is not allowed, the parents' scale parameters are updated, their fitness values are recomputed, and they remain in the population for the next generation.

1.4.7 Crossover and Mutation

When mating between two individuals is allowed, one-point crossover is performed independently on each of the string sections representing the individual feature dimensions of the candidate cluster centers. This leads to n independent crossovers per offspring, each with a crossover probability P_c. After mutation, each bit in an offspring individual's chromosome string can be inverted with a small mutation probability P_m.

1.4.8 Incorporating Constraints Against Degenerate and Spurious Solutions

When an individual in the population is initialized at a remote location in a data set, its scale parameter will either increase excessively so that more

inliers are included within its boundaries, or conversely shrink towards zero to achieve minimumn size. These degenerate solutions result in high or invalid fitness respectively. We can limit this behavior by ensuring that the scale parameter does not exceed a theoretical upper bound on σ_i^2. For example, if the data set consists of only one Gaussian cluster, with center $\mathbf{c}$ and scale parameter σ^2, then we have $\max_{i=1}^{N} dist^2(\mathbf{x}_i, \mathbf{c}) = \frac{\max_{i,j=1}^{N} dist^2(\mathbf{x}_i, \mathbf{x}_j)}{4}$. Since we can approximate $\frac{\max_{i=1}^{N} dist^2(\mathbf{x}_i, \mathbf{c})}{\sigma^2}$ with $\chi^2_{n,0.995}$, it follows that

$$\frac{\max_{i,j=1}^{N} dist^2(\mathbf{x}_i, \mathbf{x}_j)}{(2\sigma)^2} \approx \chi^2_{n,0.995}. \tag{1.12}$$

Based on (1.12), an upper bound for σ_i^2 can be computed easily using a rough estimate of the diameter of the entire data set or equivalently the maximum distance between any two data points in $\mathcal{X}$. This results in

$$\sigma^2_{max} = \frac{\sum_{p=1}^{n} \left(\max_{j=1}^{N} x_{jp} - \min_{j=1}^{N} x_{jp}\right)^2}{4\chi^2_{n,.995}}. \tag{1.13}$$

Therefore, σ_i^2 will not be updated if the updated value will exceed σ^2_{max} or falls below $\sigma^2_{min} = \frac{\sigma^2_{max}}{100}$. In the case of ellipsoidal clusters, we have noticed that poor individuals will have their scale matrix, Σ, either shrink or expand in only one dimension to yield extremely elongated clusters. While $s = |\Sigma|^{\frac{1}{n}}$ would shrink toward zero in the first case, it tends to remain within normal bounds in the second case. Hence, we need to rely on $\max_{k=1}^{n} \Sigma_{kk}$ to detect the case of an expansion situation. Hence, we will suppress the fitness of an individual that satisfies $\max_{k=1}^{n} \Sigma_{kk} > \sigma^2_{max}$ In addition to degenerate solutions with an invalid scale parameter, the population may contain spurious clusters with very few points. In general, they will form their own niches in the sparse areas of the feature space, and remain in the population owing to the diversity-maintaining nature of DC. For these individuals, the scale may also shrink toward zero. In order to eliminate these individuals, we start by forcing the fitness of an individual to be zero when its variance falls below the bound, $\sigma^2_{min} = \frac{\sigma^2_{max}}{100}$, or when its robust cardinality, $\sum_{j=1}^{N} w_{ij}$, falls below an application dependent lower limit, N_{min} (lowest acceptable cluster cardinality). Therefore we modify the fitness function to become

$$f_i = \begin{cases} \frac{\sum_{j=1}^{N} w_{ij}}{\sigma_i^2} & \text{if } \sigma^2_{min} < s_i < \sigma^2_{max} \text{ and } \sum_{j=1}^{N} w_{ij} \geq N_{min} \\ 0 & \text{otherwise} \end{cases} \tag{1.14}$$

Consequently, an individual with zero fitness can never survive into the next generation because in DC, a better fit child will always replace its parent. In addition, we prevent the child that is most similar to a zero fitness parent from replacing this parent to limit the propagation of mediocrity into future generations. This is done by modifying the DC replacement rule whenever a zero fitness or invalid parent is selected to contribute in a crossover operation.

In this case, regardless of whether crossover takes place, the invalid parent is replaced by the other parent. However, if the latter is also invalid, then it is replaced by a valid member that is randomly selected from the population. Note that we allow zero fitness parents to reproduce when mating is not restricted because by mating with individuals from different niches, it is possible to generate individuals in new unexplored areas of the feature space, particularly in the early generations. In other words, they are allowed to mate to enhance the exploration capabilities of the genetic optimization.

1.4.9 Selecting the Initial Population

The most rudimentary approach to initialization would be to select bit values randomly to build the chromosome strings of the initial individuals. However, this does not exploit the special characteristics of the clustering problem. Since the optimal solution consists of a cluster center, it is natural to select the initial centers randomly from the set of feature vectors. This results in a population of N_P individuals, P_i, $i = 1, \cdots, N_P$. The initial scale parameters are initialized using a fraction of the upper bound in (1.13), $\frac{\sigma^2_{max}}{K_\sigma}$. When dispersion matrices are used, they are all initialized to $\frac{\sigma^2_{max}}{K_\sigma} I_n$, where I_n is the $n \times n$ identity matrix. $K_\sigma = 10$ was used in all our experiments.

1.4.10 Extracting Cluster Centers From the Final Population

After convergence of the population, almost all individuals converge to the niche peaks or cluster centers. At this point, we extract the best individual from each good niche to obtain the set of final cluster centers, $\mathcal{C}$. We start by sorting the members of the final population in descending order of their fitness values, so that the better fit individuals are extracted before the lesser fit ones. Then, the best individual is extracted as the first cluster center. Subsequently, each good member of the sorted population is considered as a candidate for inclusion in the list of final cluster centers. For this purpose, the candidate center is compared to all the cluster centers that have already been extracted. The candidate center, P_i is considered to be similar to one of the previously extracted centers, P_k, if P_k is within P_i's niche and P_i is within P_k's niche. If the fitness measures depend on scalar dispersion as in 1.1, then this means that $dist(P_i, P_k) \leq K\sigma_i^2$ and $dist(P_i, P_k) \leq K\sigma_k^2$. These two conditions are equivalent to $dist(P_i, P_k) \leq K \min\left(\sigma_i^2, \sigma_k^2\right)$. In other words, the following rule can be used to decide wether two individuals are form different niches

$$\begin{aligned} &\text{IF } \left(dist(P_i, P_k) > K \min\left(\sigma_i^2, \sigma_k^2\right)\right) \text{ THEN} \\ &\quad P_i \text{ and } P_k \text{ are from different niches.} \end{aligned} \tag{1.15}$$

If the fitness measures depend on dispersion matrices as in (1.4), then the following equivalent rule can easily be derived to decide whether two indi-

viduals are form different niches (using the Mahalanobis distance measures d^2_{Mij} and d^2_{Mji} using P_i and P_j as reference niche peaks, respectively)

$$\begin{array}{l}\text{IF } \left(\max\left(d^2_{Mij}, d^2_{Mji}\right) > K\right) \text{ THEN} \\ \quad P_i \text{ and } P_k \text{ are from different niches.}\end{array} \tag{1.16}$$

When a candidate cluster is deemed to be similar to one of the extracted clusters, it is discarded and its comparison to the remaining extracted clusters is abandoned.

The extraction procedure is presented below:

Final Cluster Center Extraction

Sort individuals P_i in descending order of their fitness values to obtain $P_{(i)}$, $i = 1, \cdots, N_P$, such that $f_{(1)} \geq f_{(2)} \geq \cdots \geq f_{(N_P)}$;
Initialize set of cluster centers $\mathcal{C} = \emptyset$;
FOR $i = 1$ **TO** N_P **DO** {
  IF $f_{(i)} > f_{min\_extract}$ AND P_i and P_k are from different niches
    $\forall P_{(k)} \in \mathcal{C}$ THEN
  {
    $\mathcal{C} \leftarrow \mathcal{C} \cup P_{(i)}$;
  }
}

1.4.11 Refinement of the Extracted Prototypes

It is recommended that a local search be performed in the neighborhood of each solution provided by Genetic Optimization (GO) to increase accuracy. Fortunately, initialization becomes no longer an issue for local search methods, because GO is expected to yield a solution that is very close to the global optimum. To make the local refinement of the parameters of each cluster independent of other clusters, the data set is partitioned into c clusters before performing the local search, such that each feature vector is assigned to the closest prototype. Subsequently, the i^{th} cluster is given by $\mathcal{X}_i = \left\{ \mathbf{x}_{(k)} \in \mathcal{X} \mid d^2_{ik} < d^2_{jk} \ \forall j \neq i \right\}$, for $1 \leq i \leq c$. In [NK96], we presented a new iterative robust estimator, called the Maximal Density Estimator (MDE) which can estimate the center and scale parameters accurately and efficiently (with linear complexity). MDE uses an alternating optimization of the centers using

$$\mathbf{c}_i = \frac{\sum_{\mathbf{x}_{(j)} \in \mathcal{X}_i} w_{ij} \mathbf{x}_j}{\sum_{\mathbf{x}_{(j)} \in \mathcal{X}_i} w_{ij}}, \tag{1.17}$$

and the following update equation for the scale parameters

$$\sigma^2_i = \frac{\sum_{\mathbf{x}_{(j)} \in \mathcal{X}_i} w_{ij} d^4_{ij}}{3 \sum_{\mathbf{x}_{(j)} \in \mathcal{X}_i} w_{ij} d^2_{ij}}. \tag{1.18}$$

We have noticed that final refinement using MDE yields center and scale estimates that are significantly more accurate than the pure evolutionary search with UNC. On the other hand, UNC can be considered to provide a reliable initialization to MDE which is based on a local iterative search.

1.4.12 The Unsupervised Niche Clustering Algorithm (UNC)

The Unsupervised Niche Clustering Algorithm

Compute scale upper bound, σ^2_{max} using (1.13);
Initialize population with N_P individuals, P_i, selected randomly from the data set;
Initialize scale parameters or dispersion matrices for $i = 1$ TO N_P:
    $\sigma_i^2 = \frac{\sigma^2_{max}}{K_\sigma}$, or $\frac{\sigma^2_{max}}{K_\sigma} I_n$ respectively;
Update scale values, σ_i^2 or dispersion matrices Σ_i, using (1.8) or (1.10) respectively
Repeat for G **generations** {
  Repeat $\frac{N_P}{2}$ **times** {
    Select two parents p_1 and p_2 randomly without replacement;
    Decide whether Mating Restriction is to be imposed using (1.11);
    IF Mating NOT Restricted THEN {
      Cross them to produce children c_1 and c_2;
      Optionally apply mutation to produce children c_1' and c_2';
      Update scale values, σ_i^2 or dispersion matrices Σ_i, using (1.8) or (1.10) respectively, for parents and children
      Compute fitness values, f_i, using (1.14), for parents and children
      IF $f_i = 0$ for any of the parents or children THEN
        Replace zero-fitness individual with another individual randomly selected from population
      Use Deterministic Crowding as a replacement strategy;
    }
    ELSE {
      Update scale values, σ_i^2 or dispersion matrices Σ_i, using (1.8) or (1.10) respectively, for restricted parents
      Compute fitness values, f_i, using (1.14), for restricted parents
    }
  }
}
Extract final cluster centers and scales from the final population;
Refine final cluster centers and scales locally using (1.17) and (1.18)

1.4.13 Computational Complexity

In each generation, the most extensive computational requirement for UNC consists of computing the distance values involving N data points, fitness, and scale for each of the N_P individuals in the population, and $\frac{N_P}{2}$ inter-niche distances, resulting in $\mathcal{O}(N_P \times N)$ computations. The extraction step (done only once) requires sorting the fitnesses and computing the inter-niche

distances, resulting in $\mathcal{O}\left(N_P\left(\log N_P + N_P\right)\right)$. From our experience, the population size in UNC, N_P, tends to be a very small fraction of the size of the data, N. Hence the complexity is approximately linear ($\mathcal{O}\left(N_P \times N\right)$) with respect to the number of points. The number of iterations to converge is generally small (within 30 generations) due to the hybrid optimization, which makes it competitive even with respect to well known iterative methods such as the K-Means.

1.5 Simulation Results on Synthetic Examples

The examples used in this section illustrate the performance of UNC on spatial data in 2D, hence they are easier to inspect visually.

1.5.1 Detailed Phases of Cluster Evolution

Fig. 1.2 shows the evolution of the population (denoted by square symbols) using UNC for a noisy data set with 5 clusters. The initial population is chosen randomly from the set of feature vectors. This explains the higher concentration of solutions in the densest areas, which converge toward the correct centers in subsequent generations. The cluster center coordinates found using UNC, and refined using MDE, are shown in Figs. 1.2(e) and 1.2(f) respectively. The leveled contours correspond to boundaries including increasing quantiles ($\alpha = 0.25, 0.5, 0.75, 0.99$) of each cluster as derived from the normalized distance values $\frac{d_{ij}^2}{\sigma_i^2} = \chi_{2,\alpha}^2$. Hence, they indicate the accuracy and robustness of UNC's cluster scale estimates σ_i^2.

1.5.2 Sensitivity to GA Parameters, Noise, and Effect of Final Refinement

Our simulation results using various population sizes showed similar results as long as the population size exceeded 50 individuals. Convergence of the population occured relatively fast, after about 30 generations. Hence, we show the effect of the only GA parameters that seemed to affect the results: crossover probability (P_c), and mutation probability (P_m). With each parameter combination, we performed 100 runs and show the averages of the following clustering quality measures: **(i)** Number of final extracted clusters, **(ii)** Average normalized centroid error. To compute the error measure in (ii), we pair each discovered cluster with the closest true cluster center, compute the Euclidean distance between them, normalize this distance by the maximum possible distance in a 256×256 image (i.e., $\sqrt{256^2 + 256^2}$), and finally average these pairwise distances over all discovered clusters. The GA parameters throughout all the experiments in this paper were: population size = 80, number of generations = 30, $K = \chi_{2,.999}^2 = 13.8$, $f_{min} = 0.6$,

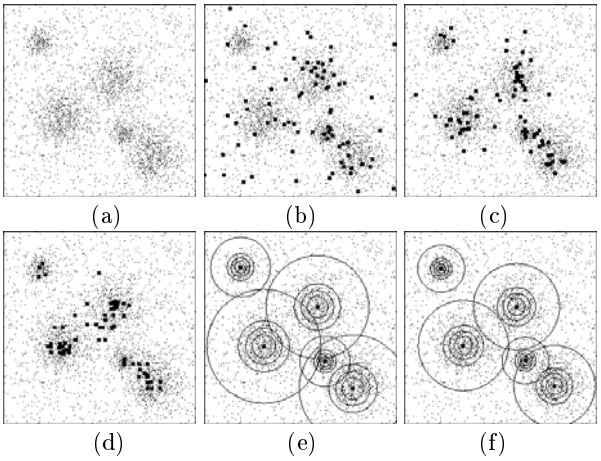


Fig. 1.2. Evolution of the population using UNC: (a) original data set (b) Initial population, (c) population after 10 generations,(d) population after 30 generations, (e) extracted centers and scales, (f) final centers and scales after refinement

$f_{min\_extract} = 0.25$, and $T_w = 0.3$. We start by studying the effect of crossover and mutation as well as refinement on the clean and noisy data sets depicted in Fig. 1.3. The GA operator parameters were varied between 0.5 and 1 for P_c and between 0.001 and 0.2 for P_m. Figs. 1.4(a) and 1.5(a) show that for a wide range of crossover and mutation rates, UNC can discover the correct number of clusters in the vast majority of the experiments. Figs. 1.4(b) and 1.5(b) show that the accuracy of the center estimates, *without any final local refinement*, is within 1.5% of the range of the images, and that this accuracy improves with an increased crossover rate, while being stable for a wide range of mutation rates. The former is due to the fact that crossover encourages local exploration and recombination of good solutions to improve converged solutions. While the latter is due to Deterministic Crowding's replacement strategy which prevents lethal children resulting from mutation, from replacing their parents. The top two curves in each of Figs. 1.4(c) and 1.5(c) show that UNC is quite robust to noise, yielding comparable unrefined center estimate accuracies for both clean and noisy versions of the data sets, while the bottom two curves confirm that final local refinement dramatically improves this accuracy, as expected. Finally Fig. 1.6 shows how each iteration of the final local refinement improves the center estimates, hence reducing the bias, while also reducing the variance of the estimates, hence improving consistency across a whole range of experiments.

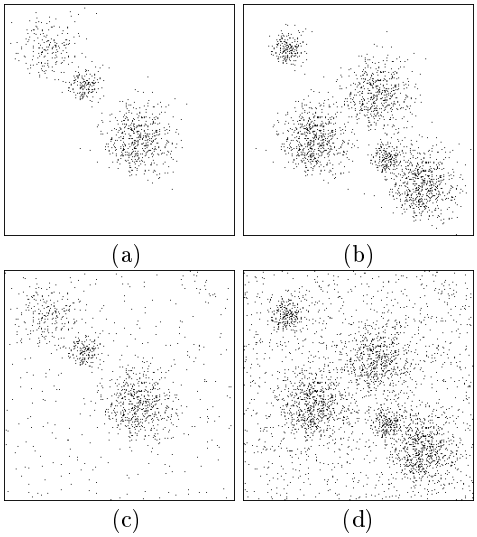


Fig. 1.3. Data Sets with 3 and 5 clusters: (a-b) Clean data sets (c-d) Noisy data sets

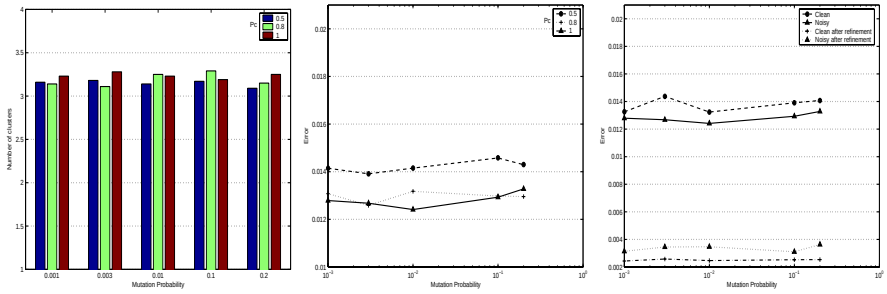
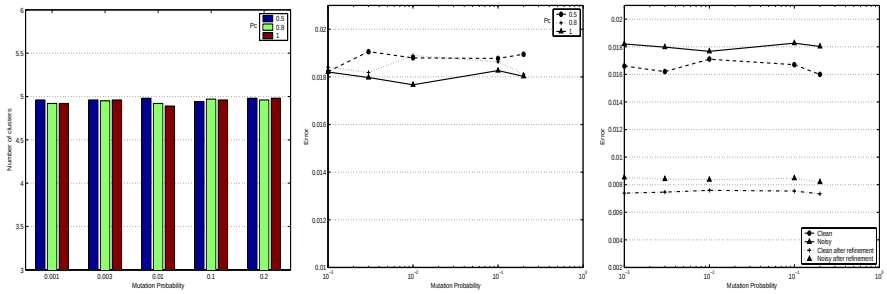
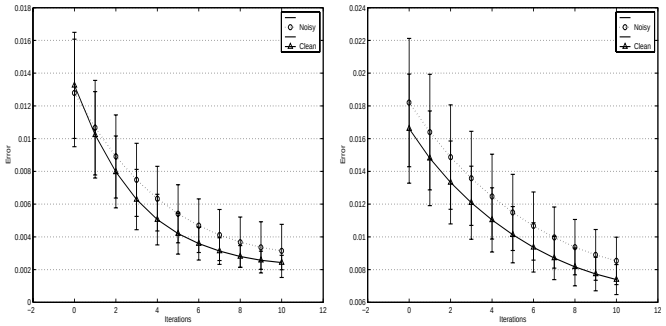


Fig. 1.4. Results of UNC for 3 clusters with noise, averaged over 100 runs:(a) Average number of cluster for different crossover and mutation probabilities (b) Center Error, (c) Center Error After Refinement ($P_c = 1$)



(a) (b) (c)

Fig. 1.5. Results of UNC for 5 clusters with noise, averaged over 100 runs:(a) Average number of cluster for different crossover and mutation probabilities (b) Center Error, (c) Center Error After Refinement ($P_c = 1$)



(a) (b)

Fig. 1.6. Effect of Final Refinement (averaged over 100 runs with $P_c = 1, P_m = 0.001$) on Average and Standard Deviation of Center Error for data sets with:(a) 3 clusters (b) 5 clusters

1.5.3 Simulation Results for Data with Varying Number of Clusters, Cluster Sizes, Densities, and Noise Contamination Rates

To illustrate UNC's performance under different conditions related to cluster size, density, noise contamination, orientation, and number of clusters, we present its results on the nine noisy data sets shown on Figs. 1.9(a) and 1.10(a) (first columns). The GA parameters were: population size = 80, number of generations = 30, crossover probability, $P_c = 1$, mutation probability, $P_m = 10^{-3}$, $K = \chi^2_{2,.999}$, $f_{min} = 0.6$, $f_{min\_extract} = 0.25$, and $T_w = 0.3$. The refinement of the centers and scale parameters is performed using 2 iterations of MDE for spherical clusters and 4 iterations of MMDE for ellipsoidal clusters.

The final centers computed using the Euclidean distance under the scalar dispersion model by UNC, K-Means [Mac67] and the Possibilistic C-Means (PCM) [KK93, KK94] are displayed in bold in Figs. 1.9(b) (second column), 1.9(c) (third column), and 1.9(d) (fourth column) respectively. Except for K-means which is not robust, and involves no scale estimation, the outermost circular contour around each cluster center depicts the normalized distances, $\frac{d^2_{ij}}{\sigma^2_i}$, corresponding to $\chi^2_{0.999}$, and hence encloses 99.9% of the data estimated to be inliers in each cluster. For PCM, σ^2_i is considered to be the cluster scale parameter η_i which is initialized using the fuzzy average distance. For UNC, the leveled contours correspond to boundaries including increasing quantiles ($\alpha = 0.25, 0.5, 0.75, 0.99$) of each cluster as derived from the normalized distance values $\frac{d^2_{ij}}{\sigma^2_i} = \chi^2_{2,\alpha}$. Hence, they indicate accuracy and robustness of UNC's cluster scale estimates σ^2_i. Therefore, it reflects *robustness*. Note that unlike K-Means and PCM which are *provided with the correct number of clusters beforehand*, UNC succeeds in determining this number *automatically*. K-Means centers were initialized using randomly selected seeds from the data set, and PCM centers were initialized using randomly selected seeds from the data set which are then refined using 5 iterations of the Fuzzy C-Means. Both K-Means and PCM are sensitive to initialization and local minima, while K-Means suffers from a blatant lack of robustness. PCM suffers from another problem, that of often yielding *identical* clusters while missing other clusters. Fig. 1.7 illustrates the case where *no good* cluster exists in the data set. Note how only UNC succeeds in recognizing this situation. Fig. 1.8 illustrates the case where the whole data set consists of a dense uniform distribution, corresponding to a single cluster, and how UNC and PCM are able to detect a single cluster, while K-Means would naively partition the data into as many clusters as pre-specified.

Figs. 1.10(b) (second column) illustrate UNC's performance, using scale dispersion *matrices*, under different conditions related to *multivariate* cluster size, orientation, density, noise contamination, and number of clusters. For comparison purposes, Figs. 1.10(c) and (d) (third and fourth columns)

show the results of K-Means and the Possibilistic C-Means (PCM) clustering algorithms, using the Gustafson-Kessel (GK) distance measure [GK79], on the same data sets. Note that unlike UNC which *determines the number of clusters automatically*, K-Means and PCM were *provided* with the correct number of clusters in advance. In these figures, the outermost ellipsoidal contour around each cluster center depicts the Mahalanobis distance, $d^2_{Mij} = \chi^2_{2,0.999}$; and the inner contours enclose increasing quantiles of each cluster as in the spherical case. Therefore, they reflect the accuracy of the final scale estimates.

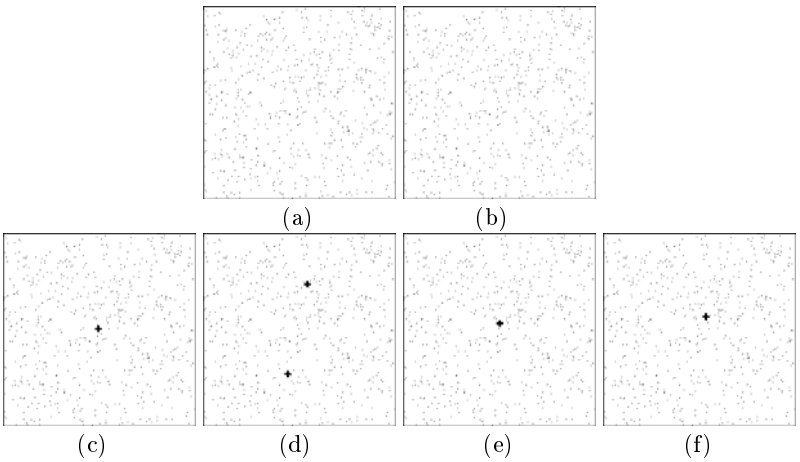


Fig. 1.7. Data set with no cluster: (a) original data set (b) Results of UNC, (c) Results of K-Means with $C = 1$, (d) Results of K-Means with $C = 2$, (e) Results of PCM with $C = 1$, (f) Results of PCM with $C = 2$

1.6 Application to Image Segmentation

Image segmentation is a typical Multimedia data mining task, where different regions of an image are to be identified, for different applications ranging from image summarization, annotation, object recognition, motion analysis, to Content Based Image Retrieval. We use UNC to cluster the feature data extracted from the images shown on Fig. 1.11(a). Six features (3 color and 3 texture) are extracted from the original images in the COREL Image Database [CTB+99]. The Euclidean distance with scalar dispersion model is used. The GA parameters were: population size = 80, number of generations = 40, $P_c = .9$, $P_m = 5 \times 10^{-6}$, $K = \chi^2_{6,.995}$, $f_{min} = 1.0$, $f_{min\_extract} = 1.0$, and $T_w = 0.4$. Figs. 1.11(b) (second column) show the segmented images with a different color assigned to each cluster. The white pixels correspond

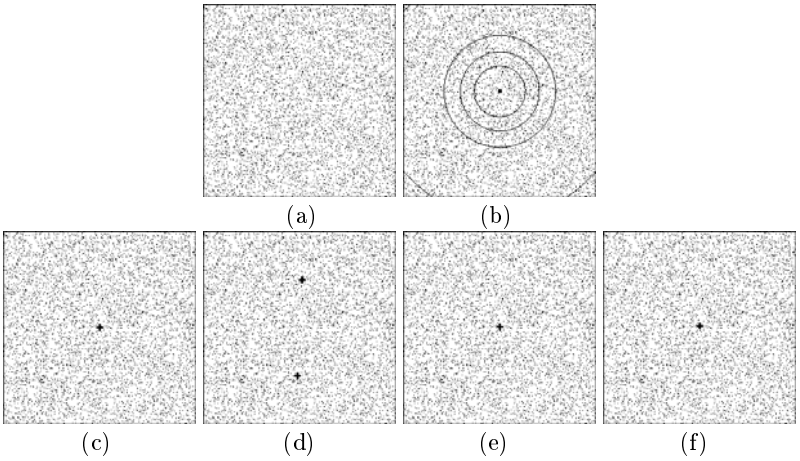


Fig. 1.8. Data set with uniform distribution:(a) original data set (b) Results of UNC, (c) Results of K-Means with $C = 1$, (d) Results of K-Means with $C = 2$, (e) Results of PCM with $C = 1$, (f) Results of PCM with $C = 2$

to noise (robust weight $< w_{min} = \exp\left(\frac{-\chi^2_{6,.995}}{2\sigma_i}\right)$). These are pixels that are either outliers or simply not at the core of a cluster. Also, often, the pixels lying on the boundaries between different regions end up being classifed as noise (since they have low weight in either cluster), due to sampling and to the types of texture features which are region based. This is a natural side-effect of region based features, and can be improved using a variety of post-processing techniques.

Identifying noise or non-core pixels allows the delineation of cleaner (more homogenous) regions, and hence makes the computation of region validation measures more reliable and accurate. From a theoretical standpoint, noise identification is paramount to Content Based Image Retrieval and Object Recognition Systems, since this allows the computation of cleaner and more accurate similarity measures, and indirectly results in more accurate scale measures, which in turn result in more accurate confidence intervals, and more robust statistical tests.

Figs. 1.11(c) (third column) show the boundary images, where the black pixels correspond to pixels on the boundary betwen distinct regions. This illustrates another (dual) by-product of the segmentation process which is edge or boundary detection. The experiments are repeated with the matrix dispersion model for scale and fitness, and the segmentation and boundary results are shown in Figs. 1.12(b) and (c) respectively, using $T_w = 0.5$. Clearly, the results obtained using Euclidean distance and a scalar dispersion show more region fragmentation (more clusters per region) than their corresponding Mahalanobis distance based and dispersion matrix segmentations. This is because even naturally elongated clusters in feature space tend to be split

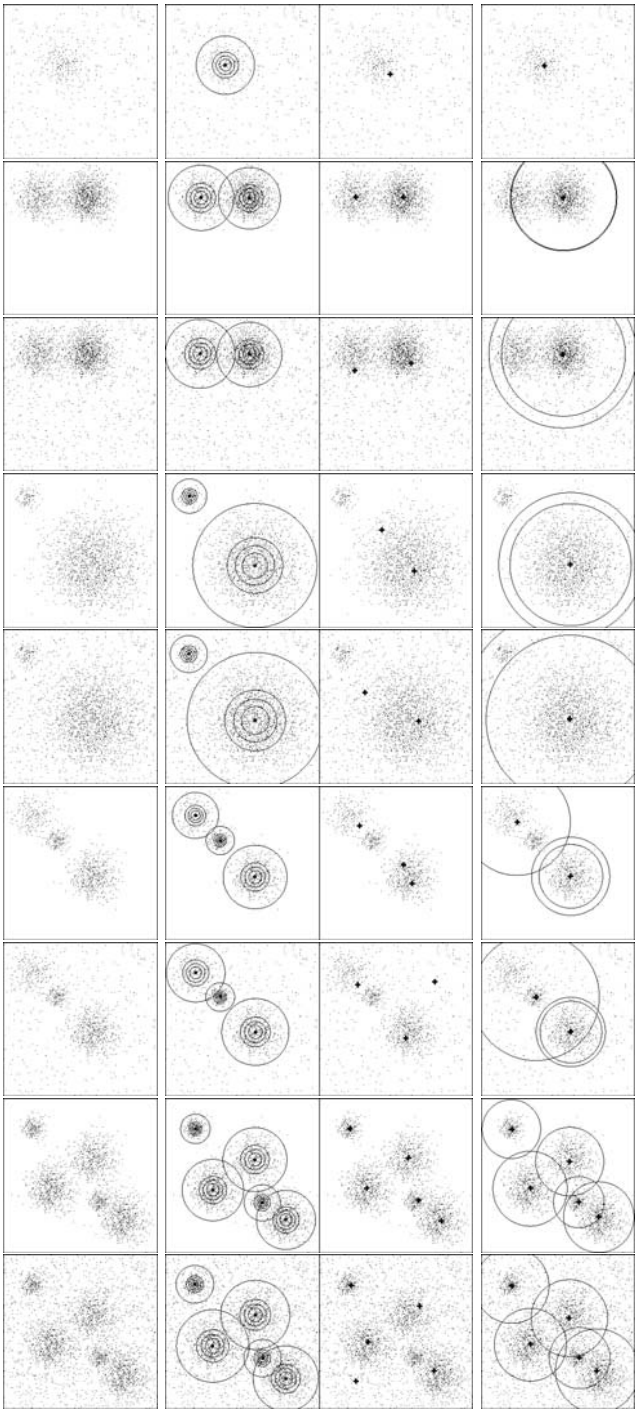


Fig. 1.9. Clustering spherical clusters: effect of different number of clusters, densities, sizes, and noise contamination rates: (a) original data set (b) Results of UNC, (c) Results of K-Means with prespecified correct c, (d) Results of PCM with prespecified correct c

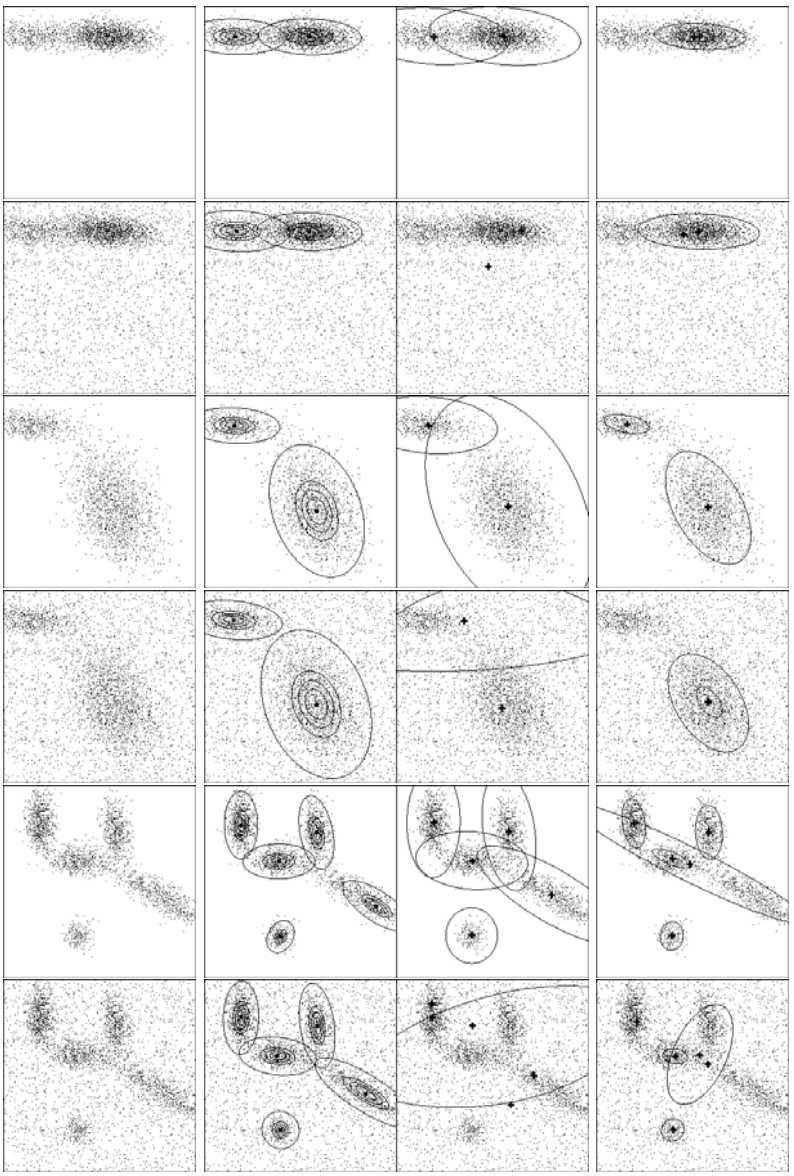
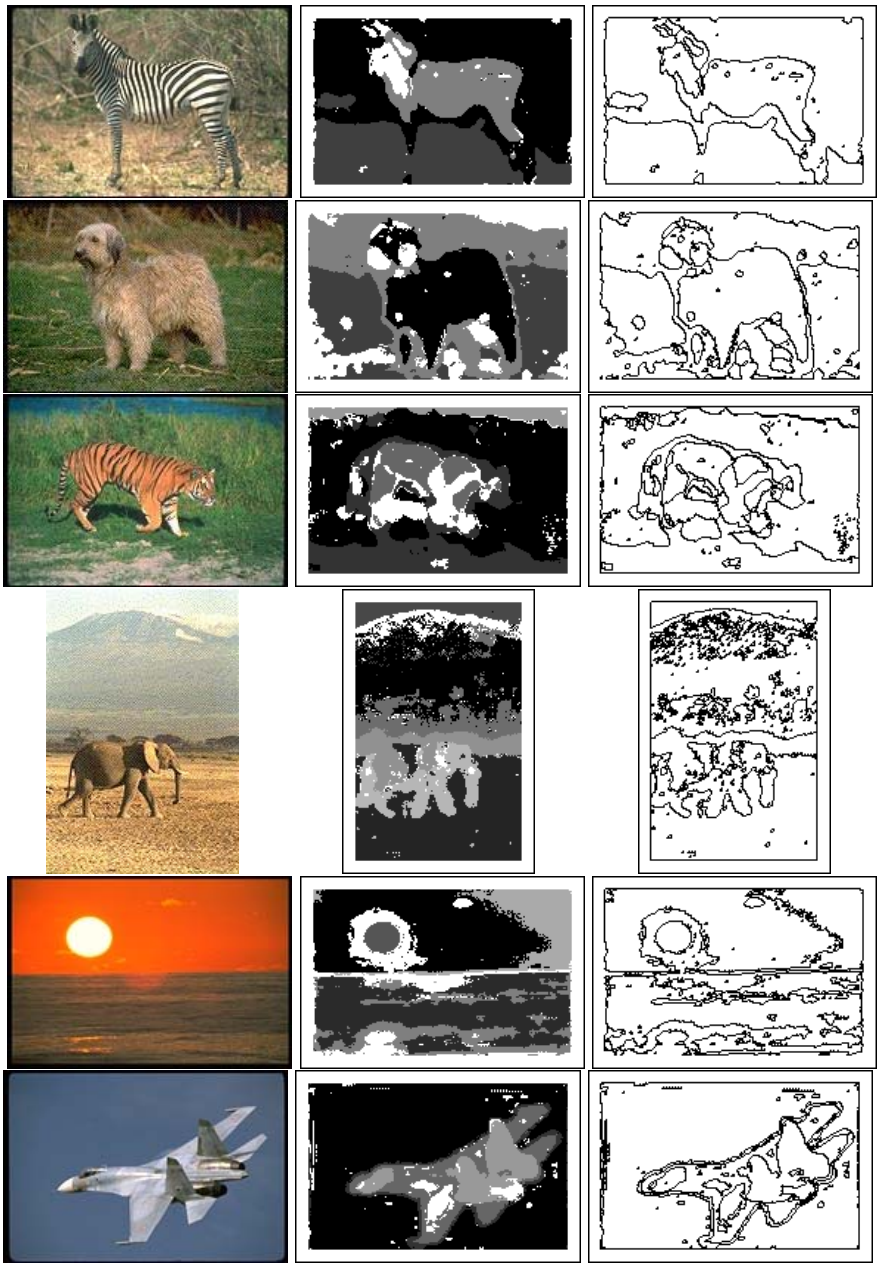
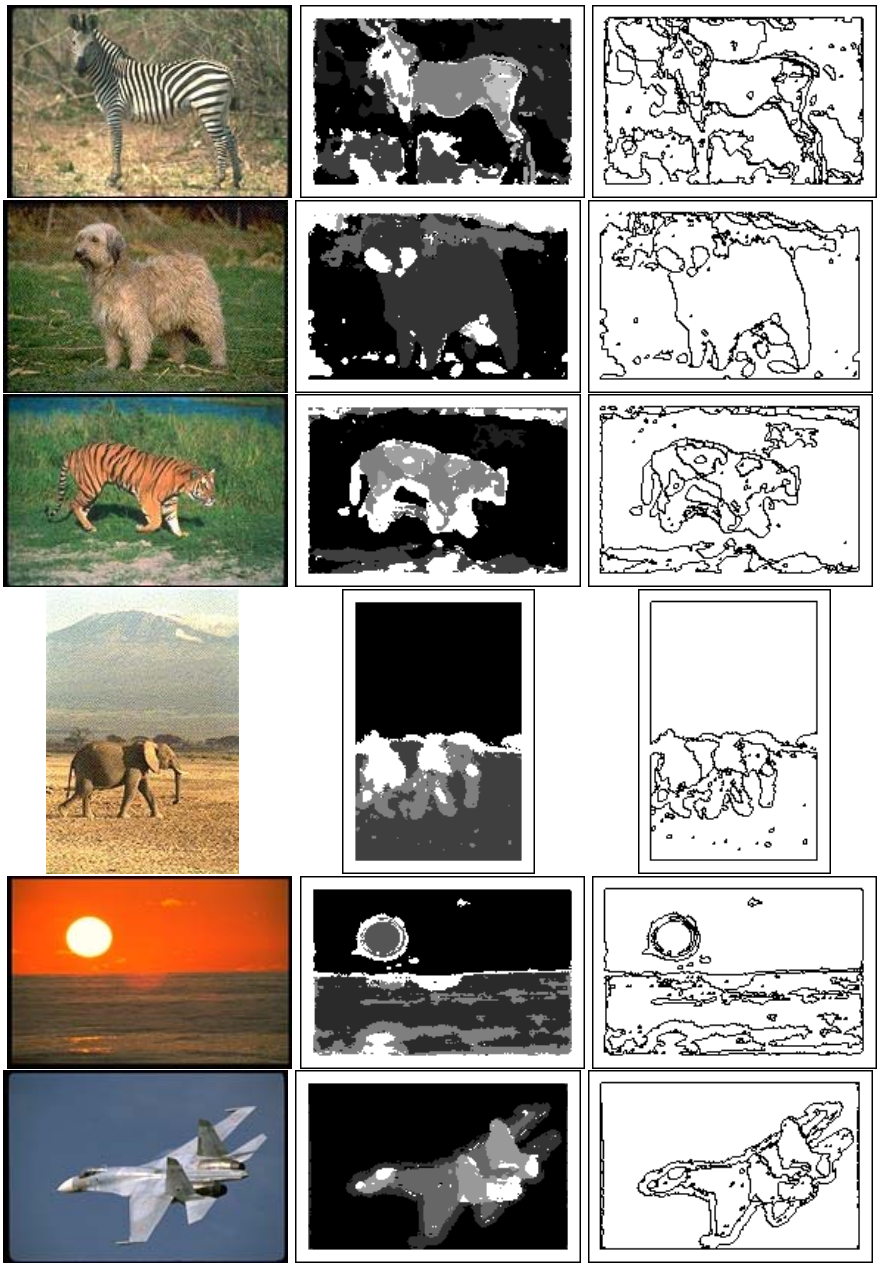


Fig. 1.10. Clustering ellipsoidal clusters: effect of different number of clusters, densities, sizes, and noise contamination rates: (a) original data set (b) Results of UNC,(c) Results of K-Means with prespecified correct c, (d) Results of PCM with prespecified correct c



(a) (b) (c)

Fig. 1.11. Image segmentation with UNC-Scalar dispersion: (a) Original image, (b) Segmented image (non-core pixels in white) (c) boundary image



(a) (b) (c)

Fig. 1.12. Image segmentation with UNC-Dispersion Matrix: (a) Original image, (b) Segmented image (non-core pixels in white) (c) boundary image

into many smaller hyper-spherical clusters when Euclidean distance is used. This problem is less present in Fig. 1.12.

1.7 Conclusion

The Unsupervised Niche Clustering (UNC) algorithm is an unsupervised robust clustering technique based on genetic niching that is applicable to a variety of data mining applications. Using an appropriate density fitness measure, clusters in feature space are transformed into niches in the fitness landscape. The niche peaks, which represent the final cluster centers, are identified based on Deterministic Crowding (DC).

We eliminate the problem of crossover interactions in DC by allowing mating only between members within the same niche. For this purpose, we estimate the robust niche radii by using an iterative hill-climbing procedure coupled with the genetic optimization of the cluster centers. An upper bound on these radii is used to disqualify invalid solutions.

Our *one-step* local Baldwin Effect learning approach, enabled by a cumulative *memory* of past generations, is more efficient than traditional hybridizations of GA and hill-climbing procedures because the latter iteratively update the parameters within each generation. However, these updates cannot be used as starting points for the next generation due to arbitrary changes in the population distribution between generations, particularly in the early generations (no memory). Our approach has the advantage of not requiring more than a single step of updating the scales, and hence adapting the environment (fitness) to accelerate evolution. This makes convergence even faster compared to conventional hybrid algorithms.

UNC outperformed K-Means and to the Possibilistic C-Means clustering (PCM) algorithms on several examples with clusters varying in size, density, orientation, and noise contamination rates.

Compared to other unsupervised clustering techniques, UNC is a good candidate for clustering in many real world problems. This is because most other techniques are either object (feature) based, and thus necessitate the derivation of the optimal prototypes by differentiation to guarantee convergence to a *local* optimum, which can be impossible for most subjective and non-metric dissimilarity measures; or work on relational data consisting of a matrix of all pairwise dissimilarity values as in the case of the Graph based and Linkage type hierarchical clustering techniques. For these techniques, both the storage and computational cost can be prohibitively high (complexity in the order of $\mathcal{O}\left(N^2 \log_2 (N)\right)$).

To summarize, this new approach to genetic clustering has the following advantages over previous methods: **(i)** It is *robust* in the presence of outliers and noise; **(ii)** it can *automatically determine the number of clusters*; **(iii)** because of the single cluster representation scheme used, *the size of the search space does not increase with the number of clusters or the number of data*;

(iv) It is *generic* enough that it can handle any type of distance/dissimiliarity measure and any type of input data (numerical, categorical, etc); and **(v)** it takes advantage of both evolutionary optimization for better *global* search, and *local Baldwin type learning* to accelerate and improve the accuracy of the search.

Acknowledgment

This work is supported by a National Science Foundation CAREER Award IIS-0133948 to O. Nasraoui.

References

[AH96] P. Arabie and L. J. Hubert. An overview of combinatorial data analysis. In P. Arabie, L. J. Hubert, and G. D. Soete, editors, *Clustering and Classification*, pages 5–63. World Scientific Pub., New Jersey, 1996.

[BBHB94] J. C. Bezdek, S. Boggavarapu, L. O. Hall, and A. Bensaid. Genetic algorithm guided clustering. In *First IEEE conference on evolutionary computation*, volume 1, pages 34–39, Orlando, Florida, June. 1994.

[Bez81] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[BFR98a] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of the 4th international conf. on Knowledge Discovery and Data Mining (KDD98)*, 1998.

[BFR98b] P. Bradley, U. Fayyad, and C. Reina. Scaling em (expectation-maximization) clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, 1998.

[BM93] G. P. Babu and M. N. Murty. A near-optimal initial seed value selection in the k-means algorithm using a genetic algorithm. *Pattern Recognition Letters*, 14(10):763–769, Oct. 1993.

[BRV91] J. N. Bhuyan, V. V. Raghavan, and K. E. Venkatesh. Genetic algorithms for clustering with an ordered representation. In *Fourth International Conference on Genetic Algorithms*, pages 408–415, 1991.

[CTB+99] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *VISUAL*, pages 509–516, Amsterdam, The Netherlands, 1999.

[DG89] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *3rd Intl. Conf. Genetic Algsorithms*, pages 42–50, San Mateo, CA, 1989.

[DH73] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley Interscience, NY, 1973.

[Dun74] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact, well separated clusters. *J. Cybernetics*, 3:32–57, 1974.

[ECY00] V. Estivill-Castro and J. Yang. Fast and robust general purpose clustering algorithms. In *Pacific Rim International Conference on Artificial Intelligence*, pages 208–218, 2000.

[FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated evolution*. Wiley Publishing, New York, 1966.

[FS93] D. B. Fogel and P. K. Simpson. Evolving fuzzy clusters. In *International Conference on Neural networks*, pages 1829–1834, San Francisco, CA, 1993.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[GK79] E. E. Gustafson and W. C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *IEEE CDC*, pages 761–766, San Diego, California, 1979.

[Gol89] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, New York, 1989.

[GR87] D. E. Goldberg and J. J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *2nd Intl. Conf. Genetic Algsorithms*, pages 41–49, Cambridge, MA, Jul. 1987.

[GRS98] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large data databases. In *Proceedings of the ACM SIGMOD conference on Management of Data*, Seattle Washington, 1998.

[HOB99] L. O. Hall, I. O. Ozyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Trans. Evolutionary Computations*, 3(2):103–112, July 1999.

[Hol75] J. H. Holland. *Adaptation in natural and artificial systems.* MIT Press, 1975.

[HW87] G. Hinton and D. Whitley. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[JMB91] J. M. Jolion, P. Meer, and S. Bataouche. Robust clustering with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):791–802, Aug. 1991.

[Jon75] K. A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. *Doct. Diss., U. of Michigan.*, 36(10-5140B):29–60, 1975.

[KF92] R. Krishnapuram and C.-P. Freg. Fitting an unknown number of lines and planes to image data through compatible cluster merging. *Pattern Recognition*, 25(4), 1992.

[KK93] R. Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Trans. Fuzzy Syst.*, 1(2):98–110, May 1993.

[KK94] R. Krishnapuram and J. M. Keller. Fuzzy and possibilistic clustering methods for computer vision. In S. Mitra, M. Gupta, and W. Kraske, editors, *Neural and Fuzzy Systems*, pages 135–159. SPIE Institute Series, 1994.

[LA00] C.-Y. Lee and E. K. Antonsson. Dynamic partitional clustering unsing evolution strategies. In *3rd Asia Pacific Conf. on simulated evolution and learning*, Nagoya, Japan, 2000.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symp. on Math. Statist. and Prob.*, pages 281–297, Berkeley, California, 1967. University of California Press.

[Mah92] S. W. Mahfoud. Crowding and preselection revisited. In *2nd Conf. Parallel problem Solving from Nature, PPSN '92*, Brussels, Belgium, Sep. 1992.

[NH94] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the VLDB conference*, Santiago Chile, 1994.

[NK96] O. Nasraoui and R. Krishnapuram. A robust estimator based on density and scale optimization, and its application to clustering. In *IEEE International Conference on Fuzzy Systems*, pages 1031–1035, New Orleans, Sep. 1996.

[NK97] O. Nasraoui and R. Krishnapuram. Clustering using a genetic fuzzy least median of squares algorithm. In *North American Fuzzy Information Processing Society Conference*, Syracuse NY, Sep. 1997.

[NK00] O. Nasraoui and R. Krishnapuram. A novel approach to unsupervised robust clustering using genetic niching. In *Ninth IEEE International Conference on Fuzzy Systems*, pages 170–175, San Antonio, TX, May 2000.

[RB79] V. V. Raghavan and K. Birchand. A clustering strategy based on a formalism of the reproductive process in a natural system. In *Second International Conference on Information Storage and Retrieval*, pages 10–22, 1979.

[RL87] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, 1987.

[Rus69] E. Ruspini. A new approach to clustering. *Information Control*, 15:22–32, 1969.

[WGM94] D. Whitley, S. Gordon, and K. Mathias. Lamarckian evolution, the baldwin effect and function optimization. In Y. Davidor, H. Schwefel, and R. Manner, editors, *Parallel Problem Solving From Nature-PPSN III*, pages 6–15. Springer Verlag, 1994.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 103–114, New York, NY, 1996. ACM Press.

Author Index

Subject Index