Multiobjective Genetic Algorithm Partitioning for Hierarchical Learning of High-Dimensional Pattern Spaces: A Learning-Follows-Decomposition Strategy

Rajeev Kumar and Peter Rockett

***Abstract*— In this paper, we present a novel approach to partitioning pattern spaces using a multiobjective genetic algorithm for identifying (near-)optimal subspaces for hierarchical learning. Our approach of "learning-follows-decomposition" is a generic solution to complex high-dimensional problems where the input space is partitioned prior to the hierarchical neural domain instead of by competitive learning. In this technique, clusters are generated on the basis of fitness of purpose—that is, they are explicitly optimized for their subsequent mapping onto the hierarchical classifier. Results of partitioning pattern spaces are presented. This strategy of preprocessing the data and explicitly optimizing the partitions for subsequent mapping onto a hierarchical classifier is found both to reduce the learning complexity and the classification time with no degradation in overall classification error rate. The classification performance of various algorithms is compared and it is suggested that the neural modules are superior for learning the localized decision surfaces of such partitions and offer better generalization.**

***Index Terms*— Genetic algorithms, neural-network architecture, pattern classification, pattern clustering methods.**

I. Introduction

THE APPROXIMATION capabilities of multilayer neural networks have been investigated in the past by many authors (see Hornick [13]). In principle, it has been shown that standard multilayer feedforward networks with a single hidden layer and arbitrary bounded and nonconstant activation functions are *universal* approximators for arbitrary finite-input environment measures provided that as many hidden units as required for internal representation are employed. This does not, however, imply that it is *computationally* easy to learn a functional mapping. A monolithic network learns the *global* nature of approximations by means of incremental adaptation of connection strengths in the direction of a decrease in error function based on some learning rule. At the same time, it is believed that a *global* error surface may have extensive flat areas and significant variations in local minima. Such situations are very common in most real-function approximations and more so with high-dimensional inputs which often require very long learning times or result in unsuccessful training. Simply increasing the number of hidden units is a popular solution, but it may unnecessarily increase the number of free parameters of the net architecture which can lead to poor generalization. One major reason for this is that scaling connectionist models to larger systems is difficult because larger networks require increasing amounts of training time and data, and the complexity of the optimization task reaches computationally unmanageable proportions.

Another major phenomenon contributing to the problem of slow/difficult training is *crosstalk*, i.e., the presence of conflicting information in the training data that retards learning. Crosstalks are identified as: *temporal* [30] and *spatial* [23]. In a temporal crosstalk, a neural net receives inconsistent training information at different times in the training cycle: receiving inconsistent information at a single instant in time is treated as spatial crosstalk. By analogy, catastrophic interference [28] results from *sequential* training when the disjoint blocks of training data are presented in sequence. We are not, however, concerned with addressing the problem of sequential training in the present work.

In the context of addressing complex learning domains, two basic categories of approach—ensemble-based (e.g., [10] and [34]) and modular systems (e.g., [14] and [21])—are emerging as possible solutions to the drawback of the poor scalability of neural networks. The ensemble-based approaches rely on combining predictions of multiple models, each of which is trained on the same database: in general, the emphasis is on improving the accuracy for better generalization and not on simplifying the function approximators (see Sharkey [27] for a review).

The main characteristic of modularity is that the system can take advantage of function decomposition. A "divide-and-conquer" strategy splits a problem into a series of subproblems and then assigns a set of function approximators to each subproblem such that each module learns to specialize in a subdomain. This strategy, however, is only beneficial if an *implicit* or *explicit* way to *sensibly* decompose a complex function exists and to let the desired (sub)function mapping emerge from the learning of subtasks. For those problems where one has some prior knowledge of the pattern space and the decomposition into subtasks is explicit, modular systems are shown to give improved performance with reduced learning effort, e.g., [17] and [32].

Manuscript received March 10, 1997; revised September 24, 1997. The work of R. Kumar was supported by a Commonwealth Scholarship Commission research studentship.

R. Kumar was with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield S1 3JD, U.K. He is now with the Department of Computer Science and Information System and the Center for Robotics and Intelligent Systems, Birla Institute of Technology and Science, Pilani-333 031, India.

P. Rockett is with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield S1 3JD, U.K.

Publisher Item Identifier S 1045-9227(98)06256-0.

In the absence of any knowledge of the pattern space sampling, decomposition-through-competition has been demonstrated where the decomposition and learning phases are combined. The basic idea of competitive learning was introduced by Nowlan [21], formalized into adaptive mixtures of experts [14], and later extended to tree-structured hierarchical mixtures of experts (HME's) [15]. This architecture consists of a set of function approximators (experts) that are combined by a classifier (gating network). Expert networks compete to learn the training patterns and the gating network mediates the competition. The architecture performs task decomposition in the sense that it *learns to partition* a task into functionally independent subtasks and allocates a distinct network to learn each task. This way different experts learn different training patterns so as to (*soft*) split the input space into regions where one particular expert can specialize. At the same time, it was also inferred that function decomposition is an underconstrained problem and different modular architectures may decompose a function in different ways [14] which is certainly not a desirable situation from the point of view of generalization by a neural network with too many degrees of freedom.

In this paper, we adopt a different line of research and propose *withdrawing* the task of decomposition from the regime of modular learning, and attempt partitioning in a *generic* manner as a preprocessor to the neural domain. We argue that separating the task of decomposition from the regime of modular learning simplifies the overall architecture and this strategy of data preprocessing before its submission to a classifier considerably reduces the learning complexity. This approach of "*learning-follows-decomposition*" can be handled with a simple modular net architecture where each partition of data is mapped onto appropriate learning modules.

In terms of complexity, such a partitioning problem is NP complete, and genetic algorithms (GA's) have been shown to be powerful tools for exploring NP-complete search spaces as efficient optimizers relative to exhaustive search [4], [8], particularly where no derivative information is available. GA's typically yield *near*-optimal solutions rather than an exact solution, but have the advantage of not needing prior knowledge of the pattern space—the number of partitions that emerges from genetic search is guided solely by the optimization criteria and is not dictated by user-defined parameters. Previous work on GA partitioning has optimized only single *ad hoc* objectives and the eventual solutions have been strongly influenced by the linear coefficients used to combine the different (sub)objectives into a single scalar fitness value—for example, [1]–[4], [9], [20], and [26].

In this work, we introduce a novel approach using a multiobjective genetic algorithm to partition the pattern space into hyperspheres for subsequent mapping onto a hierarchical neural network for subspace learning. In our technique, clusters are generated on the basis of fitness for purpose—that is, they are *explicitly* optimized for their subsequent mapping onto the hierarchical classifier—rather than emerging as some implicit property of the clustering algorithm. Most traditional clustering algorithms rely on some *similarity* measure (which is usually ill defined), and the resulting clusters depend directly on the judgement of what are and what are not nominally identical patterns—they may also fail to converge to a local minimum [24]. Multiobjective genetic algorithms perform optimization on a vector space of objectives—see Fonseca and Fleming [5] for a review—and are able to explore the NP-complete search space for a set of equally viable and equivalent partitions of the pattern space.

The remainder of this paper is organized as follows. In Section II, the rationale behind our strategy is set out, and the multiobjective genetic optimization approach is briefly reviewed. The objectives used for (near-)optimal partitioning of feature spaces for hierarchical learning are identified in Section III. Implementation details are briefly described in Section IV. Results are presented for both high-dimensional synthetic and real data in Section V and discussed in Section VI. Finally, the conclusions are reported in Section VII.

II. Rationale

Conceptually, our approach has strong links to the recursive feature space partitioning algorithms of Henrichon and Fu [11] and Friedman [6] for classification using hyperplanes parallel to feature axes. The rationale for such hierarchical partitioning and their interactions and differences with artificial intelligence and pattern recognition have been summarized by Kanal [16]. In another simpler version of the algorithm, Sethi and Sarvarayudu [25] have shown the application of the algorithm to a real multifeature multiclass problem involving handprinted numeral recognition. Though partitioning using a set of hyperplanes is the simplest method conceivable, hyperplanes effectively give rise to a decision tree classifier, and a pattern is classified by following a path through the tree from the root to a leaf. Tree classifiers have the advantage that a global decision can be made hierarchically by a series of simple and local decisions, but the main drawback is that they can be brittle: a wrong decision at a higher level of the tree leads to an error which is generally unrecoverable. Second, difficult pattern recognition problems with complex decision boundaries may require the formation of a very large number of hypervolumes.

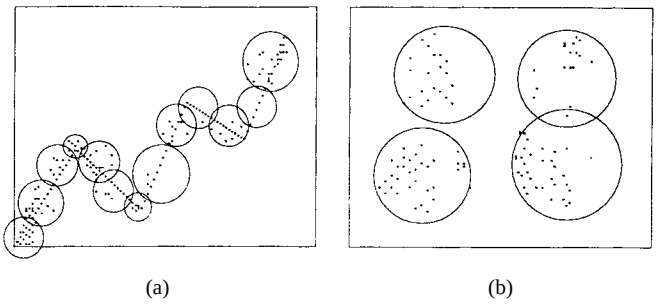
In this work, we have chosen to use hyperspheres to partition the pattern space since their chromosomal representation is comparatively compact. Partitioning using other geometric primitives (e.g., hyperellipsoids) is possible, but these require a significantly larger number of free parameters leading to much longer chromosomes and, thus, greatly increased time for genetic search.

A. Subfunction Approximation

In this work, we aim to partition feature spaces into subspaces and their corresponding mappings on hierarchical neural networks for efficient problem solving. If we consider feature partitioning as a mapping P from an N-dimensional feature space to j subspaces of dimensionality, n_j

$$\mathrm{P}: \quad R^N \rightarrow \bigcup_j R^{n_j}, \qquad n_j \leq N$$

$$\text{subject to } \mathrm{Min/Max}\ Obj\_f_i(\boldsymbol{X})$$



(a) (b)

Fig. 1. Feature space partitioning. (a) Visualization of subfunction approximation and (b) clustering into hyperspheres.

then this formulation is an N-dimensional function decomposition into many n_j-dimensional subfunctions subject to meeting certain criteria $Obj\_f_i(\boldsymbol{X})$. Since n_j represents (hopefully) a less complex domain, a classifier can approximate such a subdomain with less effort. It is more appropriate, within a learning domain, to talk about the *intrinsic* dimensionality rather than the true dimensions of a space [22]. We shall discuss in the next section that, in principle, we mean the local intrinsic dimensions rather than the true dimensions of a subspace for the mapping P from an N-dimensional feature space to n_j-dimensional subspaces.

Fig. 1(a) illustrates a simple case of decomposing a two-dimensional (2-D) function into many subfunctions. Here, the 2-D function is parameterized by a series of *intrinsically* one-dimensional (1-D) functions, and although in this illustrative example the dimensionality reduction is trivial, this will not be the case in higher dimensional spaces. Alternatively—in Fig. 1(b)—the hyperspherical clusters can enclose pattern "blobs," and two situations can arise in practice: 1) all the patterns in each hypersphere belong to a single class, i.e., partitioning alone demarcates the decision boundaries and no classification stage is needed and 2) the patterns belong to multiple classes necessitating the use of some postpartitioning classifier.

B. Multiobjective Genetic Optimization

Mathematically, a general multiobjective optimization problem containing a number of objectives to be maximized/minimized along with (optional) constraints for satisfaction of achievable goal vectors can be written as

$$\begin{aligned} \text{Minimize/Maximize} \quad & \text{Objective } f_m(\boldsymbol{X}) \\ & m = 1, 2, \cdots, M \\ \text{subject to} \quad & \text{Constraint } g_k(\boldsymbol{X}) \leq c_K \\ & k = 1, 2, \cdots k \end{aligned}$$

$$\begin{aligned} \text{where } \boldsymbol{X} &= \{x_n \colon n = 1, 2, \cdots, N\} \text{ is an } N \\ &\quad - \text{tuple vector of variables} \\ \text{and } \boldsymbol{F} &= \{f_m \colon m = 1, 2, \cdots M\} \text{ is an } M \\ &\quad - \text{tuple vector of objectives.} \end{aligned}$$

Goldberg's [8] condition of pareto optimality is stated as: in a minimization problem, an individual objective vector $\boldsymbol{F}_i$ is partially less than another individual objective vector $\boldsymbol{F}_j$ (symbolically $\boldsymbol{F}_i \prec \boldsymbol{F}_j$) *iff*

$$(\boldsymbol{F}_i \prec \boldsymbol{F}_j) \triangleq (\forall_m)(f_{mi} \leq f_{mj}) \ \wedge \ (\exists_m)(f_{mi} <, f_{mj}).$$

In this case, the individual $\boldsymbol{F}_i$ dominates the individual $\boldsymbol{F}_j$. If an individual is *not* dominated by any other member of the population, it is said to be *nondominated, noninferior* or a *pareto-optimal* solution. In a typical multiobjective optimization problem, there exists a family of such solutions which are considered *equivalent* from the perspective of simultaneous optimization of multiple and possibly competing objective functions.

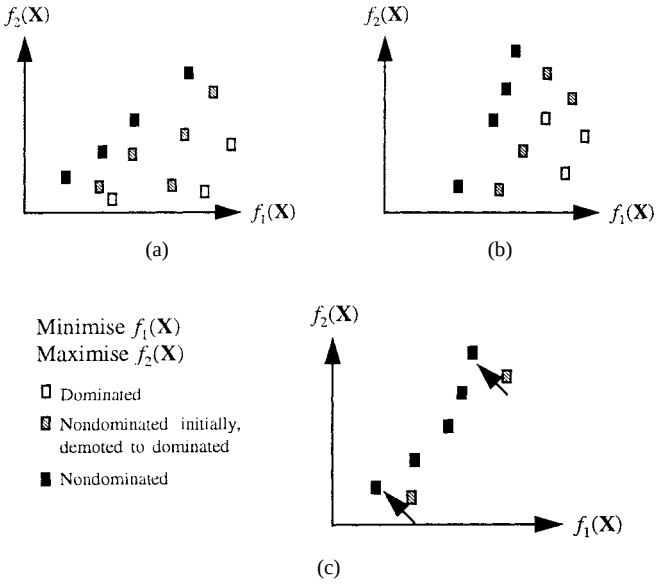


Fig. 2. Propagation of the pareto front through successive evaluations. (a) A population set. (b) Another population set. (c) Both merged together, resulting into a shift of the pareto front.

Let us consider a two objective function problem where objective f_1 is to be minimized and another objective f_2 is to be maximized. As a randomly initialized population is progressively evolved through successive generations some of the nondominated solutions become dominated. Eventually, the solution space consists of some dominated solutions and other nondominated ones. The locus in objective space made up of nondominated (pareto-optimal) solutions is referred as the *pareto front*—such a situation is illustrated in Fig. 2(a). With another independently randomized population set, the similar situation is shown in Fig. 2(b). On combining both the solution spaces, some of the nondominated solutions of either of the populations may become dominated (demoted to lower pareto ranks), and this is accompanied by a shift in the pareto front as shown in Fig. 2(c). Finally, at convergence, neither one of these objectives can be improved without degrading the other, and all the nondominated solutions constitute the final set of equivalent solutions.

In the present work, we perform optimization on a vector space of objectives and explore the NP-complete search space for a set of equally viable partitions of the pattern space.

III. Objectives for Subspace Learning

We have identified a set of seven independent objectives for partitioning the pattern space and optimizing learning effort.

1) ***Minimize*** **the number of hyperspheres** aims to exploit the modularity, but it should be based on minimizing the *overall* training effort. Alternatively, this objective can be withdrawn, and the number of partitioning hyperspheres can be specified in advance based on some prior knowledge of the problem domain.
2) ***Minimize*** **the learning complexity** which, in a feedforward network, is approximately of the order of $O(N^3)$, where N is the number of weights in the network [12]. For a given pattern space, the number of inputs and outputs remain fixed therefore the number of weights is proportional to the number of hidden units. It has been shown that the *effective* number of hidden units is (approximately) equal to the *intrinsic dimensionality* [33]. Thus, our objective is to minimize the sum of the cubes of the intrinsic dimensionality of the subspaces. We have taken a conservative estimate in determining the intrinsic dimensionality and included the components up to some proportion, say 0.95, of the total variance within a hypersphere as the determining criterion of intrinsic dimensionality.
3) ***Maximize*** **the regularity of the decision surface** aims at increasing the classification accuracy. The nearest neighbor classification error is used to indicate how well the partitions preserve the total structure of the pattern space as a separability measure [7]. This objective indirectly maximizes the correct classification probability of a k-NN classifier within the partitions where we assume that each hypersphere is an "independent" event in the statistical sense. We thus multiply their individual probabilities to combine the k-NN results to form a single measure of the regularity of the decision surface(s).
4) ***Maximize*** **the fraction of included patterns of each class** aims to include within all the partitions as many training patterns as possible from each class. Hopefully, outliers within the pattern space can be excluded because the objective does not aim to include *all* patterns.
5) ***Minimize*** **the maximum fraction of included patterns in a single hypersphere** aims to distribute the included patterns over as many partitions as possible. It discourages inclusion of all the patterns in only one or a few partitions.
6) ***Minimize*** **the overlap of partitions** aims to avoid repetition of learning effort on similar sets of patterns in different modules but allows *some* overlap of hyperspheres to prevent the formation of a "no-man's land" between the partitions. A direct calculation of the overlap between hyperspheres is possible, in principle, but it is both cumbersome and potentially misleading because volumes (and possibly multiple) intersections of N-dimensional hyperspheres do not give a clear physical picture of their geometry. We have used an alternative, simpler measure of "overlap" where we aim to "push" apart two hyperspheres with a rudimentary coulombic-type repulsion model.
7) ***Minimize*** **the surface area** attempts to produce compact solutions. The surface content of a hypersphere [29] is normalized by the number of patterns included within it so that the objective is biased toward compact solutions. This is effectively a data density measure and militates against solutions containing hyperspheres which can otherwise exist due to enclosing minimal quantities of data.

All seven elements in the objective vector are distinct and competing as well as complementary to each other. Objective 1 minimizes the number of hyperspheres whereas Objective 2 splits training data into many simpler subspaces. Thus, both together give an optimum number of clusters from a learning-time complexity point of view. Objective 3 preserves the regularity of the decision surface and hence complements Objective 2 in its attempts to minimize learning efforts—both together favor that the local structure of the clusters should be preserved for increased classification accuracy and without unnecessarily increasing the learning effort. Objective 7 (compactness of partitioning) also supports minimized learning effort with increased accuracy. At the same time, the compact solutions should be formed with the goal of including as many patterns of each class as possible (Objective 4). Objective 5 (minimizing the maximum fraction of included patterns in a single hypersphere) competes with Objective 4 in that all the patterns should not be included in a single cluster. Objectives 4 and 7 together can result in compact solutions, but with multiple instances of training patterns. Objective 6 competes with Objective 4 in that all the patterns should be included, but with minimal multiple inclusion of data points in more than one cluster. Objectives 6 and 7 are complementary to each other in that compact solutions should be formed with minimal multiple use. Objectives 1, 3, 4, and 6 may give a single-cluster solution, but are in competition with Objectives 2, 5, and 7 and so on and so forth.

All the seven objectives ensure a fair distribution of potential solutions. We do not scalarize the fitness of the solutions into a weighted sum of seven-element objectives, but compare on the scale of pareto-ranking/nondominance. At convergence, any given one of these objectives cannot be improved without degrading at least one of the others leading to a *set* of equivalent solutions.

IV. Implementation

To represent subspace partitions, our GA implementation uses variable length individuals where each subblock encodes the hypersphere center and radius. Each unit of a chromosome is a real number, and $(N+1)$ such units form a block where N is the dimensionality of the pattern space. Variable length of individuals is necessitated by the fact that the number of clusters emerging from the search is unknown and so the number of clusters in the (near-)optimal solution is also evolved genetically such that the number of blocks forming a chromosome represents the number of partitions of the pattern space. A sample chromosome is illustrated in Fig. 3.

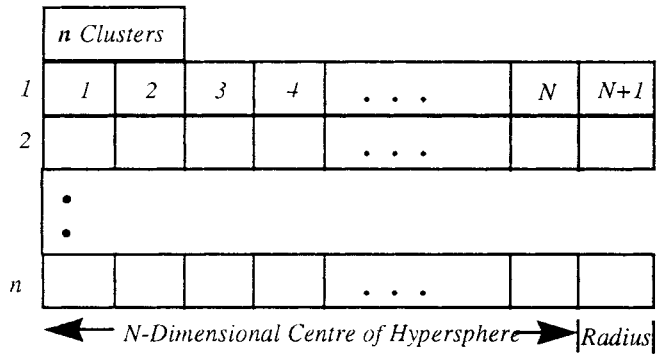


Fig. 3. A variable-length chromosome representing the partitions.

The constraints on decision variables are drawn from the bounds of the pattern space. We have investigated two approaches for initializing the chromosomes. In one approach, the cluster centers were randomly initialized and in the other a hypersphere was centered on a randomly selected data pattern. The second approach of seeding a chromosome proved particularly well suited to sparsely populated pattern spaces and thus significantly reduced the search effort. For the genetic search, we employed a single-point crossover operation on the hypersphere limits and a Gaussian mutation. The crossover point is taken on the boundaries between hypersphere description records to prevent the formation of illegal chromosomes. Apart from meaningful recombination, this has the added advantage that good clusters can be retained, but shuffled among solutions. The mutation stage involved adding zero-mean Gaussian noise to the center coordinates and the radii of the hyperspheres.

The search operations are further minimized using a few heuristics. One heuristic acts on the upper bound of the radii of a hypersphere where the upper bound is divided by the square root of the dimensionality of the space. This heuristic prevents the potential inclusion of all the patterns into each partition of the feature space. Complementary to this, another heuristic limits the minimum fraction of the patterns included in a hypersphere: partitions containing less than some fraction of the total patterns—in this work, 2%—are prevented from forming a separate cluster. One other heuristic acts on the maximum number of clusters forming a solution since preventing the number of partitions from becoming arbitrarily large helps restrict the search among the space of many (partially or wholly overlapped) clusters.

Another option is to look for some predetermined number of clusters. This can be useful if one has prior knowledge of the pattern space from, say, viewing the data with standard ordination techniques, or one can tune the computation to some fixed number of partitions after becoming acquainted with the nature of the solutions obtained during the initial GA runs.

In spite of the constraints and heuristics, exploring an NP-complete search space for a set of equally viable partitions of the pattern space is a complex optimization. We used the pareto converging genetic algorithm (PCGA) [19] which naturally performs good sampling of the solution space and ensures population advancement toward the pareto front. The added advantages of PCGA strategy are that the algorithm does not need *a priori* knowledge of the objective space and minimizes the number of heuristically chosen parameters and procedures, such as mating restrictions. From the obtained set of solutions, a small subset based on subranking of objectives were picked for hierarchical learning. The *ANCHOR* connectionist architecture [18] which we have developed for integrating multiple heterogeneous classifiers is particularly suitable for hierarchical learning of subspaces.

V. Results

We have applied our partitioning strategy to a range of synthetic problems as well as a real classification problem. First, we partitioned synthetic data from two, three-dimensional (3-D) Gaussian blobs embedded in a six space, and within this we have examined two cases: one where the two blobs are just separated and the other where they overlap. Each class contained 100 examples. For the just-separated data a large number of equivalent solutions evolved, most of which comprised two clusters of three *intrinsic* dimensions, each containing only data from the separate classes although some solutions contained exemplars from *both* classes. From the point of view of the GA, all nondominated solutions are equivalent, but some may be more desirable in practice. For the overlapped Gaussian blobs, the GA produced partitions of intrinsic dimensionality of three or four and containing 5%–10% data from the other class, both of which would be expected for this dataset. Positioning two hyperspheres on the (known) Gaussian centers and carrying out an exhaustive search for the two "best" hypersphere radii produced partitions which were comparable to the typical GA results indicating that the GA was indeed finding close-to-optimal clusters for both cases.

We have also considered the partitioning of a four-class synthetic problem in 12 variables—here each Gaussian blob was of three (mutually exclusive) dimensions and just separated from the others. Again, we generated 100 random data points from each class. Most of the family of equivalent solutions produced comprised four clusters of three intrinsic dimensions, each containing around 100 data points. A number of pareto-equivalent solutions, however, contained seven-dimensional (7-D) or eight-dimensinal (8-D) hyperspheres.

Finally, we have partitioned a benchmark problem in land-use classification of multispectral satellite image data of thirty-six dimensions.[1] The dataset description and classification results for various algorithms are given in Taylor *et al.* [31], where the best classification accuracy was obtained with a k-NN classifier. For simplicity, we reduced the original six classes to a two-class problem (the cotton crop versus all others) and randomly subsampled the original 6435 data points to give a training set of 500. To investigate the behavior of our partitioning algorithm in greater detail we fixed the number of clusters to two, four, and six in respective clustering runs—from viewing the data with standard ordination techniques it seemed that two clusters would be too few and six probably too many. Results for two hyperspheres

[1] Available from the UCI Machine Learning repository at http://www.ics.uci.edu/AI/ML/MLDBRepository.html or ELENA.

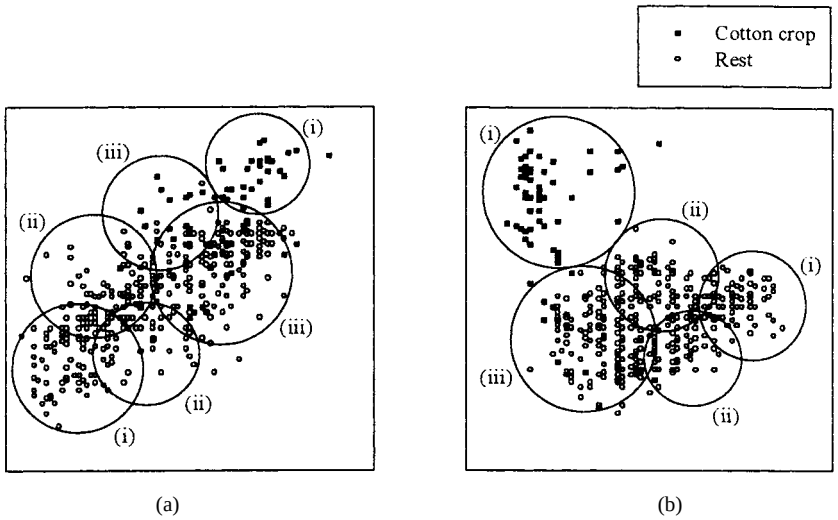


Fig. 4. Hypothetical views of land-use data partitioning—data projection on arbitrary axes in (a) and (b). Three main categories of clusters are apparent—(i) all the patterns within a hypersphere belong to a single class, (ii) a very few percent of patterns belong to the other class, and (iii) both the classes are present roughly in the 25%–75% split.

produced some partitions which contained only members of one class and a roughly 50 : 50 split in the other partition. Most solutions, however, included a hypersphere containing around ten members of the other class. This latter situation is an unattractive partitioning since *within* one of the hyperspheres, one class has a very small prior which would lead to difficulties in reliably training a neural network. Partitions based on four clusters produced mostly hyperspheres of a single class together with hyperspheres of roughly 50 : 50 membership. Six partitions produced very similar results to the four partition case except that the aggregate overlap measure was increased and a few of the hyperspheres were degenerate in that they largely or wholly overlapped other hyperspheres. We draw the conclusion that six clusters is indeed too many for this particular problem. A hypothetical 2-D view of data projection on some arbitrary axes along with the hyperspheres is shown in Fig. 4.

We obtained classification results on the land use data based on the 500 training set examples and using the remaining 5935 examples as test data. Taking the whole, unpartitioned dataset produced a 3-NN error rate of 1.11% and a 5-NN error rate of 1.23%, whereas training a feedforward neural network gave error rates of 1.23%–1.48% dependent on architecture and the (random) initialization. The slightly poorer performance of multilayer perceptrons (MLP's) relative to nearest neighbor classifiers is probably to be expected on such a dense data set. We also measured the k-NN classification rates for $k = 7$ and 9 and found that $k = 3$ gives the best results on the test data both among nearest neighbor classifiers and also compared to the other classification algorithms. (The misclassification rates are summarized in Fig. 6.) These observations are identical to those reported by Taylor *et al.* [31] that k-NN is the best for this land-use data. Since k-NN appears to give the best performance for this dataset as well as being simple (although laborious), we have used 3-NN classification as a benchmark for our partitioning approach in the rest of this paper.

Nearest neighbor error rates for the two- and four-cluster partitioned data were essentially identical to the results from treating the dataset as a monolithic block, and depending on the particular partitioning chosen, some were slightly better, but none was worse. Misclassification rates of six-cluster solutions are shown in Fig. 5(a) where the majority were slightly better and a few were worse. Assuming the results are normally distributed, the 3-NN error rate is within one standard deviation $(\sigma = 0.0021)$ of the mean $(\mu = 0.00971)$, and, thus, we view any apparent improvement in error rate as not statistically significant. The fact that the classification accuracy was not *reduced* is highly significant—although the error rate has not changed the *computational effort for both training and recall is significantly reduced.*

Obviously, the objective was not to include all training data within the clusters and so some data points were excluded from the solutions and these are potentially outliers. The distribution of the fraction of excluded points is shown in Fig. 5(b). Outliers do not tend to greatly degrade the performance of a k-NN classifier, but they can have a serious effect in the case of a feedforward network. Hence, performance improvement in the absence of outliers is not particularly marked with a k-NN classifier [Fig. 5(a)], but we believe it would be with the decision boundaries formed with feedforward networks—this is discussed further in subsequent paragraphs.

The excluded test patterns in the distribution of Fig. 5(b) did not fall inside any of the hyperspherical clusters and there are different ways of classifying such patterns. If one has some prior knowledge of the outliers in the pattern space and the equivalent number of patterns are excluded, all such exclusions can be treated as resulting from outliers and ignored in the classification process. Such knowledge of the presence of outliers guides the selection of solutions from the data distribution of Fig. 5(b) which could appropriately exclude the outlier patterns. In another treatment, each of the excluded patterns can be classified based on its distance to the nearest cluster, or, if there are sufficiently many clusters, using a k nearest- cluster "voting" strategy. Both of these treatments will

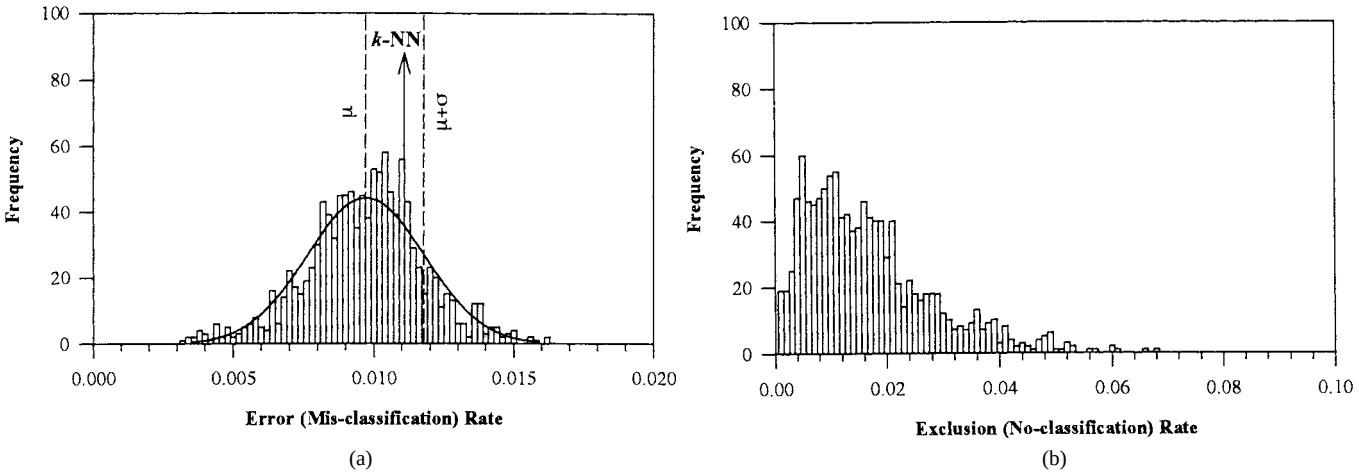


Fig. 5. (a) Misclassification rate of unseen data (5935 patterns) computed with 3-NN classifier using 500 patterns as the training data. The histogram represents the error-rates of the partitioned (six clusters each) solutions while the upward arrow indicates the error of the monolithic data. The Gaussian fitted into the histogram and the mean and standard deviation are also shown. (b) The distribution of excluded patterns from the obtained solutions.

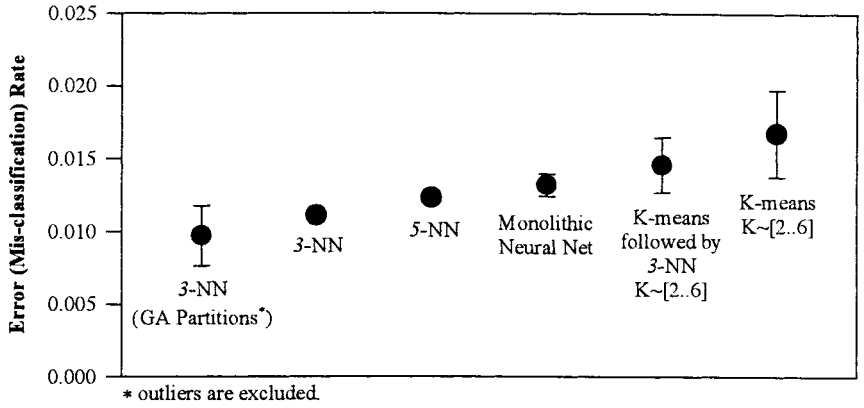


Fig. 6. A summary of misclassification rates with various models/algorithms of unseen data (5935 patterns) using 500 patterns as the training data.

militate against corruption of the decisions and also reduce the learning effort.

In an attempt to compare the genetic algorithm partitioning results with those of traditional clustering algorithms, we generated clusters in the range of [2]–[6] using the K-means clustering algorithm. For a fair comparison of the results, we generated the cluster centers with the same training data, and calculated the 3-NN error rates for each cluster. The K-means clustering followed by a 3-NN classification within the bounding hyperspheres centered on each cluster gave error rates in the range of 1.21%–1.71% dependent on the number of clusters and initial cluster centers. The traditional K-means clustering produced error rates of 1.35%–2.48%. The error rates of various models and algorithms are summarized in Fig. 6. Looking at the composition of clusters across all the pareto-optimal solutions, we observed that only one or two clusters in each solution need postpartitioning classification. This is exactly what we are aiming for in the learning-follows-decomposition strategy since the localized decision surfaces should possess considerably reduced complexity over the global decision surface. We trained such "split-class" partitions with both 3-NN and neural modules and measured the performance on the test data. The performance of the neural-modules was better than that of 3-NN classifier (Fig. 7), however, the performance improvement is small and may be problem dependent. Nonetheless, this is in keeping with the notion that on sparse datasets—and within a cluster the dataset is comparatively sparse—neural networks are better able to generalize across the limited data available. This makes hierarchical neural classifiers a natural companion for the present partitioning approach.

From the standpoint of the time required for classification, typically three out of four or four out of six partitions contained only a single class and so once inclusion within a hypersphere was established, labeling an unknown datum was trivial. Even for hyperspheres with a roughly equal split in the numbers of included classes, classification of an unknown point required far fewer nearest neighbor distance calculations than needed for classification based on the whole training set of five hundred. Such "half-and-half" clusters typically contained 100–140 total patterns. Thus, in nearest neighbor classification, our partitioned dataset gave error rates which

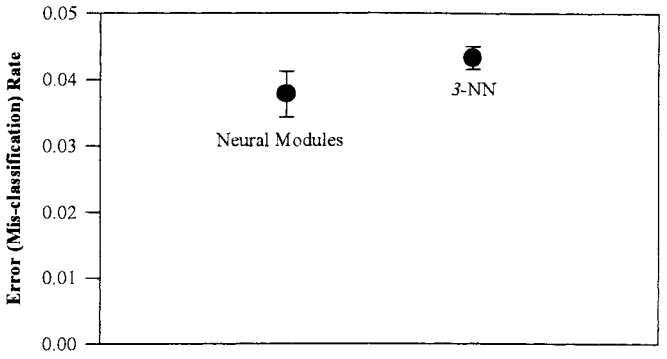


Fig. 7. Misclassification rates of trivial partitions with a neural module and 3-NN classifier.

were not degraded over a monolithic classifier, but the time to compute a label was reduced significantly. Hierarchical neural learning is at an advantage for both reduced time and improved generalization.

VI. Discussion

From the above results, we observe that three main categories of cluster emerge: 1) all patterns within a hypersphere belong to a single class; 2) a very few percent of patterns belong to the other class; and 3) both the classes are present in approximately equally numbers (Fig. 4). The first category of clusters does not require any postpartitioning effort for classification since to label an unknown point it is sufficient to determine in which cluster it is included. In the third category of clusters, mapping on to a feedforward network is fairly straightforward since the roughly equal numbers of exemplars from each class together with the reduced size of the subset to be learned both simplify training. For the second broad category of cluster, a k-NN classifier could be employed to decide the final classification within a hypersphere with less computation than would be required for nearest neighbor classification on the whole training set although clearly, unless at least k members of the minority class are included the classification effectively degenerates to the first category. Alternatively, a feedforward network could be used, but special measures are required to accommodate the unbalanced training which is well known to pose problems for learning. As a further option, a small fraction of examples within a hypersphere could be ignored at the cost of a minute increase in error rate by treating all included patterns as from the grossly dominant class; we have observed a number of examples of clusters containing 200 members from one class and a single member from the other class.

This present generic approach to partitioning as a preprocessor to the subsequent classifier is suited to a wide spectrum of complex problems, particularly where there is no prior knowledge of the pattern space which could guide clustering. Most traditional clustering algorithms rely on some *similarity* measure—which is usually ill defined—and the resulting clusters depend directly on the judgement of what are and what are not nominally identical patterns. Our multiobjective GA approach avoids any such judgement of similarity and instead forms clusters on the basis of *fitness for purpose*—namely, trying to simultaneously maximize a set of general properties we wish to emerge from a set of partitions. This property of genetic partitioning has been shown to exhibit superior results to those obtained from K-means clustering (Fig. 6). In this work, we have geared our vector of objectives to mapping onto an ensemble of MLP neural networks, but clearly any desired set of objectives could be employed to maximize fitness for some other purpose. We thus believe the present algorithm is a truly generic approach to clustering which could, if desired, incorporate traditional similarity measures as one of the objectives.

The presence of outliers in a training set is known to pose problems, particularly for neural learning. In our strategy isolated, "eponymous" outliers may well be discarded since the relevant objective tries only to maximize the number of patterns utilized, not to force the usage of all patterns. Similarly, clusters of outliers caused by some systematic measurement failure are likely to generate their own hypersphere which may well be significantly separated from other patterns with the same class label. The treatment of outliers is the subject of further research.

The strategy adopted in this work also supports the concept of ensemble-based approaches. Ensemble-based approaches rely on integrating multiple classifiers to improve prediction accuracy by repeatedly mapping the *whole* data set on multiple models. In our approach, the clusters which contain only one data class do not require any further processing and are implicitly labeled without ambiguity. In fact, we have partitioned the land usage data using 500 training patterns and the test data was approximately twelve times larger. The effectiveness of such labeling was confirmed by the fact that in all the several thousand pareto-optimal clusters we examined, we did not find any case where a cluster containing a single class of training data was subsequently found to include a single member of the other class from the test data. In a partitioned ensemble approach, we suggest that only those clusters where more than one class is represented need to be multiply mapped on suitable classifiers. Thus, the principal advantage of our partitioning approach is that only those patterns which lie near decision boundaries warrant learning effort. We also propose a way of dealing with multiple instances of patterns (which is possible because one of the objectives directly promotes some overlap): the clusters can be assigned priorities based on inclusion of patterns of a single class and if a pattern is included in more than one cluster having different priorities, it can be safely assigned to the class of the higher-priority cluster. These additions may well enhance the accuracy of ensemble-based approaches and the functional simplicity of modular systems.

VII. Conclusions

In this paper, we have presented a novel approach to partitioning pattern spaces using a multiobjective genetic algorithm for identifying (near-)optimal subspaces for hierarchical learning. learning-follows-decomposition is a generic solution to complex high-dimensional problems. The results of parti-

tioning pattern spaces have been presented. This strategy of preprocessing the data and explicitly optimizing the partitions for subsequent mapping on to a hierarchical classifier is found to both reduce the learning complexity and classification time for no statistically significant degradation in overall classification error rate. Classification performance of various algorithms have been compared and it is argued that the neural modules are superior for learning the localized decision surfaces of such partitions as well as offering better generalization than both a k-NN classifier—the best for monolithic data set—and a monolithic neural network.

REFERENCES

[1] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial Intell.*, vol. 40, pp. 235–282, 1989.

[2] F. Z. Brill, D. E. Brown, and W. N. Martin, "Fast genetic selection of features for neural network classifiers," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 324–328, 1992.

[3] E I. Chang and R. P. Lippmann, "Using genetic algorithms to improve pattern classification performance," in *Advances in Neural Information Processing System*, vol. 3, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 797–803.

[4] K. A. DeJong and W. M. Spears, "Using genetic algorithms to solve NP-complete problems," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 124–132.

[5] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[6] J. H. Friedman, "A recursive partitioning decision role for nonparametric classification," *IEEE Trans. Computers*, vol. 26, no. 4, pp. 404–408, 1977.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. San Diego, CA: Academic, 1990.

[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[9] H. A. Gvenir and I. Sirin, "A genetic algorithm for classification by feature partitioning," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 543–548.

[10] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 10, pp. 993–1001, 1990.

[11] E. G. Henrichon and K. S. Fu, "A nonparametric partitioning procedure for pattern classification," *IEEE Trans. Computers*, vol. 18, no. 7, pp. 614–624, 1969.

[12] G. E. Hinton, "Connectionist learning procedures," *Artificial Intell.*, vol. 40, pp. 185–234, 1989.

[13] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[14] R. A. Jacobs, M. I. Jordan, and A. G. Barto, "Task decomposition through competition in a modular connectionist architecture: the what and where vision tasks," *Cognitive Sci.*, vol. 15, pp. 219–250, 1991.

[15] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994.

[16] L. N. Kanal, "Problem-solving models and search strategies for pattern recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 1, no. 2, pp. 193–201, 1979.

[17] R. Kumar, W. C. Chen, and P. I. Rockett, "Bayesian labeling of image corner features using a grey-level corner model with a bootstrapped modular neural network," in *Proc. IEE 5th Int. Conf. Artificial Neural Networks (ANN 97)*, pp. 82–87.

[18] R. Kumar and P. I. Rockett, "ANCHOR—A connectionist architecture for hierarchical nesting of multiple heterogeneous neural nets," in *Working Notes AAAI Workshop Integrating Multiple Learned Models (IMLM 96)*, pp. 59–65.

[19] ———, "Improved sampling of the pareto front in multiobjective genetic optimizations by neo-stationary evolution: A Pareto converging genetic algorithm," to be published.

[20] C. A. Murthy and N. Chowdhury, "In search of optimal clusters using genetic algorithms," *Pattern Recog. Lett.*, vol. 17, no. 8, pp. 825–832, 1996.

[21] S. J. Nowlan, "Maximum likelihood competitive learning," in *Advances in Neural Information Processing Systems*, vol. 2, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990.

[22] E. Oja, "Neural networks, principal components and subspaces," *Int. J. Neural Syst.*, vol. 1, no. 1, pp. 61–68, 1989.

[23] D. C. Plaut and G. E. Hinton, "Learning sets of filters using backpropagation," *Computer, Speech Languages*, vol. 2, pp. 35–61, 1987.

[24] S. Z. Selim and M. A. Ismail, "K-means type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, no. 1, pp. 81–87, 1984.

[25] I. K. Sethi and G. P. R. Sarvarayudu, "Hierarchical classifier design using mutual information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 4, no. 4, pp. 441–445, 1982.

[26] W. Siedlecki and J. Sklansky, "Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 141–150.

[27] A. J. C. Sharkey, "On combining artificial neural nets," *Connection Sci.*, vol. 8, nos. 3/4, pp. 299–313, 1996.

[28] N. E. Sharkey and A. J. C. Sharkey, "An analysis of catastrophic interference," *Connection Sci.*, vol. 7, nos. 3/4, pp. 301–329, 1995.

[29] D. M. Y. Sommerville, *An Introduction to the Geometry of N-Dimensions*. New York: Dover, 1958.

[30] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proc. 8th Annu. Conf. Cognitive Science Soc.*, 1986, pp. 823–831.

[31] C. C. Taylor *et al.*, "Dataset descriptions and results," in *Machine Learning, Neural and Statistical Classification*, D. Michie, D. J. Spiegelhalter, and C. C. Taylor, Eds. London, U.K.: Ellis Horwood, 1994, pp. 131–174.

[32] A. Waibel, H. Sawai, and K. Shikano, "Modularity and scaling in large phonemic neural networks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 12, pp. 1888–1897, 1989.

[33] A. S. Weigend and D. E. Rumelhart, "The effective dimension of the space of hidden units," in *Proc. IEEE Int. J. Conf. Neural Networks*, 1991, pp. 2069–2074.

[34] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

Rajeev Kumar received the M.Sc. degree in physics electronics from Allahabad University, India, the M.Tech. degree in computer science and technology from Roorkee University, India, and the Ph.D. degree in machine vision from the University of Sheffield, Sheffield, U.K., in 1997.

He was with the Department of Science and Technology, Government of India, from 1983 to 1986 and the Defense Research and Development Organization, Government of India, from 1986 to 1984. Currently, he is a Member of the faculty of Birla Institute of Technology and Science, Pilani, India, the Department of Computer Science and Information System, and the Center for Robotics and Intelligent Systems. His current interests include machine vision, neural nets, genetic algorithms, and robotics.

Peter Rockett received the B.Sc. (Hon.) degree in electronics, the M.Sc. degree in solid-state electronics, and the Ph.D. degree in semiconductor physics, all from the University of Manchester Institute of Science and Technology (UMIST), Manchester, U.K.

After receiving his Ph.D. degree, he was a Research Assistant at the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, U.K. In 1990, he was appointed to tenured Lectureship in Electronic Systems at the University of Sheffield. His research interests include: machine vision, particularly, object recognition, and the application of artificial neural networks to early vision, statistical pattern recognition, the application of genetic algorithms, and visual guidance of robots.

Dr. Rockett is a Member of the International Program Committee of the IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA). He has acted as an Academic Consultant to the European Commission. In 1982, he was awarded a Science and Engineering Research Council (SERC) Post-Doctoral Research Fellowship and, in 1983, an SERC Advanced Post-Doctoral Research Fellow in Information Technology, both held at the Electronic and Electrical Engineering Department, University of Sheffield.