6

An Efficient Nearest Neighbor Classifier

R. Frédéric and G. Serge

Summary. In order to build an efficient nearest neighbor classifier three objectives have to be reached: achieve a high accuracy rate, minimize the set of prototypes to make the classifier tractable even with large databases, and finally, reduce the set of features used to describe the prototypes. Irrelevant or redundant features are likely to contribute more noise than useful information.

These objectives are not independent. This chapter investigates a method based on a hybrid genetic algorithm combined with a local optimization procedure. Some concepts are introduced to promote both diversity and elitism in the genetic population. The prototype selection aims to remove noisy and superfluous prototypes and selects among the others only the most critical ones. Moreover, the better the selection the faster the algorithm.

The interest of the method is demonstrated with synthetic and real chemometric data, involving a large number of features. The performances are compared to those obtained with well known algorithms.

Key words: Feature selection, Genetic algorithm, Hybrid algorithm, classification, k nearest neighbors

6.1 Introduction

The k nearest neighbor algorithm is widely used in classification. Given a database with training samples, whose class label is known, it consists in finding the k nearest neighbors of a pattern in order to classify and assign the unknown pattern a label which takes into account the labels of its neighbors. The neighborhood is defined by a distance function in the input, or feature, space.

When choosing $k = 1$, the classification rule consists in assigning the unknown pattern its nearest neighbor label. Compared to other well known classifiers, based for example on radial basis function or back propagation neural networks, neighborhood techniques remain very attractive thanks to their easy use. No assumption on class shape is made and the training procedure is quite simple.

R. Frédéric and G. Serge: *An Efficient Nearest Neighbor Classifier*, Studies in Computational Intelligence (SCI) **75**, 127–145 (2007)
www.springerlink.com

In order to build an efficient nearest neighbor classifier three objectives have to be reached: achieve a high accuracy rate, minimize the set of prototypes to discard the noisy and superfluous ones and make the classifier tractable even with large databases, and finally, reduce the set of features used to describe the prototypes. Irrelevant or redundant features are likely to contribute more noise than useful information.

One of these objectives is feature selection, a topic which, while remaining a challenging issue, has been extensively researched [1–3]. The selection can be done either by considering each feature independently, the filter approach [4], or by managing a subset of the available features, as in wrapper approaches [5]. The feature selection step is of prime importance as the distance function works in the feature space. The neighborhood of a given pattern is highly dependent on this distance function, i.e., the selected features.

Another objective is related to instance selection problems. These problems are managed using editing and condensing techniques [6]: The goal is to select the most critical patterns in order to make the classifier faster and more relevant. This objective is of prime concern when dealing with large databases as finding a pattern neighborhood requires as many distance computations as there are items in the reference data set.

A prototype selection procedure combines editing [7] and condensing [8,9] techniques. The former technique aims at cleaning the data by removing irrelevant patterns while the latter's goal is to find representatives. Both contribute to reducing the set cardinality. Many methods have been proposed by different scientific communities. The oldest, and probably most famous one, is the "condensed nearest neighbor rule" presented by Hart [10]. More recently, Gates introduced the "reduced nearest neighbor rule" [11] and Swonger proposed the "iterative condensation algorithm" [12]. A survey of the available selecting methods can be found in [13]. The DROP family methods [14] are the most popular within the pattern recognition community. They aim at discarding the noncritical instances. Starting from the original set, it removes the instances step by step, in an ordered way. An item is removed if at least the same number of its well-classified neighbors can be correctly classified after the item's removal. This process is decremental.

Feature selection and prototype reduction are not independent, as the result of the classification depends on the neighborhood, which depends on the selected features. The main approaches to deal simultaneously with this twofold objective are detailed in Sect. 6.2. They are all based on genetic algorithms (GA). GA are one of the best-known techniques for solving optimization problems. The tuning of the genetic parameters, mutation and crossover rates, population sizes, is still an open research field [15–17]. Many specific techniques have been used within selection problems. Niching methods and particularly various sharing and crowding strategies [18] have been developed to keep the population diverse under the constraint of the selection pressure. They contribute to avoiding premature convergence.

Despite real progress in the GA field, recent literature advocates the use of hybrid algorithms [19–22], as GA may fail in finding a good solution in high dimensional systems. Hybrid algorithms, often called memetic algorithms [23, 24], combine GA

with a local search procedure. A huge number of papers [25–28] has appeared in the last 10 years, most of them dealing with quite small systems. Finding the balance between accuracy and tractability is not easy.

In this chapter, a hybrid genetic approach is proposed to design a nearest neighbor classifier. This approach proposes a specific hybrid procedure, based on a genetic algorithm, adapted to the efficient nearest neighbor classifier design. The goal is to provide the user with all the basic elements needed to easily implement an operational system suitable for medium databases (several hundred patterns described by up to two hundred features) such as those found in the chemometric field. Section 6.2 summarizes the existing dual selection approaches. The hybrid genetic algorithm is introduced in Sect. 6.3. Its application on several databases, as well as the comparison with alternative methods, is carried out in Sect. 6.4. Finally, Sect. 6.5 reports some concluding remarks.

6.2 Problem Statement

Let $Z = z_1, \ldots, z_p$ be a set of samples described by a set of features $X = x_1, \ldots, x_f$. Each item, $z_j \in R^f$, is labeled, $L = 1, \ldots, l$ being the set of available labels. Given C_{1nn} a nearest neighbor classifier, the optimization problem consists in finding the smallest subsets $S1 \subseteq X$ and $S2 \subseteq Z$ such that the classification accuracy of C_{1nn} over Z is maximal.

The formulation is then the following, $|.|$ standing for set cardinality:

$$\begin{cases} C_{1nn}(Z) & \text{is maximal} \\ |S_1| & \text{is minimal} \\ |S_2| & \text{is minimal} \end{cases} . \tag{6.1}$$

As these objectives are conflicting ones, the optimization problem is usually managed by heuristic approaches. Various techniques are involved such as quantization techniques or more recently the well known Tabu search algorithm [29]. They rather focus on the classification objective by optimizing the prototype location. Most of the methods dealing with the three objectives combine condensation techniques and feature selection separately or are GA based.

The oldest, and undoubtedly the best-known studies, are those by Shalak [30] and Kuncheva [31]. Shalak's pioneering work addresses the problem in a partial way. The fitness function manages only the classification accuracy, the number of prototypes, $|S2|$, being a priori given by the user. Dealing with well known (and quite easy to manage) databases, he aims to show that only one prototype by class is needed to obtain good classification results. Computation costs are thereby hugely reduced. This approach is obviously limited to a small range of applications.

Kuncheva [31] proposed a complete approach, optimizing the sizes of both prototype and feature sets. She applied a standard elitist GA well designed to perform in a single step edition and selection by aggregating the objectives. Her work shows that a good trade-off, high discrimination rate with a moderate number of features, can be

reached in real-world applications. The results presented are better than all the others obtained by procedures that manage feature selection and edition in different steps.

Recently Ho [32] and Chen [33] have proposed new derived approaches. The first paper is mainly dedicated to the genetic part of the process: It proposes an intelligent crossover (IGA) based on an orthogonal experimental design. This paper shows this approach is likely to improve the results gained by existing GA-based and non-GA-based classifiers in terms of classification accuracy and prototype number. In Chen, the same genetic concept holds, but they propose a new multiobjective management. Their claim is that an aggregative fitness function may not be suited as it is difficult for practitioners to determine the appropriate weights. Instead of aggregating the objectives in a single function, chromosome comparison is achieved in the three spaces, using the well known Pareto strategy [34]. According to this strategy, a chromosome is said to be better than another if it is strictly better in at least one criteria and no worse, better or equivalent, in all the other objectives. Then the chromosomes can be ordered and each of them can by characterized by its position: How many chromosomes does it dominate and by how many other ones is it dominated? This combination of intelligent crossover and multiobjective fitness seems to yield quite good results. In Chen, the ones presented are better than those gained by each of these approaches independently, either IGA or strength Pareto evolutionary algorithm (SPEA).

To the best of our knowledge, no hybrid method has been applied to design an efficient nearest neighbor classifier. Our proposal is hybrid. It is based on self-controlled phases with dedicated objectives combining crowding and elitist strategies. Elitism and pressure preservation are reinforced by a mechanism involving a breaking process and an evolutionary memory. The genetic exploration is driven by an aggregative fitness assignment strategy. The GA is hybridized via forward and backward local procedures. The hybridization is structured in such a way that the classifier tractability and efficiency are optimized. Some neighborhood concepts related to the prototype nature are also incorporated in the local procedures. By progressively filtering useless and noisy prototypes, they contribute to facilitating and improving the natural selection of GA. Let us now go into detail.

6.3 The Hybrid Algorithm

The whole procedure comprises two distinct steps. The first one, which can be called a preliminary phase, is a pure GA. The goal is to promote diversity within the chromosome population in order to remove the unused features and to prepare the second step, called the convergence phase. Then the objective is to find a set of possible solutions. Instead of diversity, the internal mechanisms favor elitism and some local tuning is combined with the GA. It should be noted that the transition between the preliminary and the convergence phases is automatic. Furthermore, additional neighborhood considerations are introduced to select critical prototypes and discard superfluous or noisy ones. The first subsection goes into the GA details while the second is dedicated to the hybrid component.

6.3.1 The Genetic Algorithm

As the optimization procedure deals with two distinct spaces, the feature space and the pattern space, both are managed by the GA. A chromosome represents the whole solution. It is encoded as a string of bits, whose length is $f+p$, f being the number of available features and p the number of patterns in the training set. In a chromosome a 1 for the ith feature or pattern stands for its selection, while a 0 means it is not taken into account.

As the number of features is likely to be smaller than the number of patterns, in order to speed up the procedure and to improve the exploration power of the algorithm, the two spaces are managed independently at each iteration by the genetic operators such as crossover and mutation. This means the whole chromosome is the union of two distinct subchromosomes, the first one to encode the feature space and the second one the pattern space. In each subchromosome a classical one-point crossover is applied.

We superimpose some restrictions for a chromosome to represent a valid solution. The first one is obvious: the number of selected features is not zero, $|S_1| \geq 1$. Otherwise, no input space would be defined. The other condition aims at ensuring that all the classes are managed by the system, whatever their cardinality. The number of prototypes of a given class has to be greater than a defined proportion. Without this kind of constraint, bad results with small classes could be compensated by good results with larger ones.

The initial chromosomes are not generated in a completely random way: The number of active bits is limited for both spaces. The intervals are $[a_1 p, a_2 p]$ and $[1, min(a_3, f)]$ (typical values are $a_1 = 0.1$, $a_2 = 0.5$, and $a_3 = 50$).

The choice of the fitness function is of prime importance in a GA design. The one we propose takes into account the three conflicting objectives: maximize the classification results, and minimize both the number of features and prototypes.

It is, of course, defined for valid chromosomes only. Its analytical expression, to maximize, is as follows

$$F = w_{\mathrm{c}} C_{1\mathrm{nn}}(Z) + w_{\mathrm{f}} \lambda_{\mathrm{f}} + w_{\mathrm{p}} \lambda_{\mathrm{p}} \tag{6.2}$$

with $w_{\mathrm{c}} + w_{\mathrm{f}} + w_{\mathrm{p}} = 1$. The weight values stand for the importance of the corresponding objectives. In our case, $w_{\mathrm{f}} \approx w_{\mathrm{p}}$ to mean that the reduction is of comparable importance for feature and pattern spaces, and $w_{\mathrm{c}} > w_{\mathrm{f}}$ to enhance the accuracy objective.

$C_{1\mathrm{nn}}(Z)$ is usually evaluated over the training set. It should be noted that the combined use of a validation set, if available, is likely to improve the generalization capability of the classifier. Note that this validation set is different from the test set used to assess the final performance.

The λ parameters, λ_{f} and λ_{p}, have to be maximal for a small number of selected features or patterns. We used

$$\lambda_{\mathrm{f}} = \begin{cases} 1 - \frac{|S_1|}{f_{\max}} & \text{if } |S_1| \leq f_{\max} \\ 0 & \text{else} \end{cases} \tag{6.3}$$

$$\lambda_{\mathrm{p}} = \begin{cases} 1 - \frac{|S_2|}{p_{\max}} & \text{if } |S_2| \leq p_{\max} \\ 0 & \text{else} \end{cases} . \tag{6.4}$$

Let us underline the difference between our proposal and the one proposed in [31] or [32]. In their function, the last two objectives are managed using a single term:

$$\frac{|S_1| + |S_2|}{f + p}.$$

This compensation is likely to favor the pattern space as it is usually of higher dimension. By the use of two separate terms, each of the spaces is really considered with the same care. This offers more flexibility. The other difference resides in the normalization factor. Whereas Kuncheva used the number of available features or patterns, we propose a new parameter, $f_{\max}$ for the feature space and $p_{\max}$ for the pattern space. It is used to limit the range of variation of λ: The user considers that a solution which requires more than $f_{\max}$ features is not very different from another one which would use all the features. As an example, dealing with a classification problem with a hundred features, $f_{\max}$ could be set to 15. That means that a feature space of dimension 5 is really better than one of dimension 15 or higher.

Preserving both elitism and diversity constitutes the main challenge for a GA. Most methods such as determinist crowding (DC), restricted tournament selection (RTS), and others [35, 36] are continuously looking for a balance between elitism and diversity in the current population. We propose to use two distinct populations with different evolution rules and no direct interaction. The first one is called the current population, $\mathrm{pop_c}$, its evolution is managed using classical genetic schemes (elitism, DC, RTS, etc.) The second one is called the archive population, $\mathrm{pop_a}$, it acts as an evolutionary memory. It is a repository of good chromosome solutions found during the evolution. At each generation, $\mathrm{pop_a}$ is updated and may be used to partially regenerate $\mathrm{pop_c}$ if needed. The final $\mathrm{pop_a}$ constitutes the output of the GA.

The current population needs to be reseeded when a diversity index drops below a given threshold. The breaking mechanism is then used to produce major changes in the current population by including chromosomes from the archive population or applying a high mutation rate to refresh the chromosome.

The diversity index is based on the chromosomes similarities. Two chromosomes are said to be similar if their hamming distance is less than a predefined threshold. As a chromosome is the union of two subchromosomes, the hamming distances are computed in the two different spaces. The similarity between the ith and jth chromosomes is

$$s(i,j) = \begin{cases} 1 & \text{if } d_{\mathrm{h}}^{\mathrm{f}}(i,j) < n_{\mathrm{f}} \text{ and } d_{\mathrm{h}}^{\mathrm{p}}(i,j) < n_{\mathrm{p}} \\ 0 & \text{else} \end{cases} , \tag{6.5}$$

where $d_{\mathrm{h}}^{\mathrm{f}}(i,j)$ (respectively, $d_{\mathrm{h}}^{\mathrm{p}}(i,j)$) stands for the hamming distance in the feature (respectively, pattern) space, and n_{f} (respectively, n_{p}) is a predefined threshold.

The proportion of chromosomes similar to the ith one is given by

$$P_s(i) = \frac{1}{s-1} \sum_{j=1, j\neq i}^{s} s(i,j), \tag{6.6}$$

where s is the population size.

The breaking mechanism is active when there are a lot of similar chromosomes within the population. The $P_s(i)$ are thresholded to compute the diversity index:

$$\mathrm{DI} = \frac{1}{s} \sum_{i=1}^{s} S(i) \qquad \text{where } S(i) = 1 \; if \; P_s(i) > \mathrm{th}_{\min} \text{ and 0 else.} \tag{6.7}$$

When the diversity index, DI, is too low, some of the chromosomes which have a lot of similar ones in the population, some of the i ones for which $S(i) = 1$, are either replaced by ones randomly chosen in the archive population or regenerated with a high mutation probability.

The update of the archive population takes into account both elitism and diversity. The decision to include a given chromosome in $\mathrm{pop_a}$ is based on two criteria, the first one is the fitness score. If there exists a chromosome in the archive population with a much lower score than the candidate, it is replaced by the candidate. This is the elitist side of the process. If the candidate score is slightly better than others, the candidate replaces the chromosome with the most comparable structure, the one with the closest hamming distance. Even if the candidate score is a little worse than that of the archive population, it can be used to replace one of a set of similar chromosomes, in order to increase the diversity level.

As previously stated the whole procedure is made up of two steps. For the preliminary phase, whose main objective is to promote diversity, we have selected the RTS genetic scheme for $\mathrm{pop_c}$ evolution, the diversity level being controlled by the breaking mechanism. There is no hybridization with local approaches within this preliminary phase.

This stage automatically ends when there is a large enough number of good and diverse chromosomes in the population. This condition can be formulated as follows. Let S' be the set of chromosomes whose fitness score is greater than a threshold, and F_{div} (respectively, P_{div}) a diversity measure in the feature (respectively, pattern) space. The fulfillment of the condition states that the three indexes, $s' = |S'|$, F_{div} and P_{div} have to be sufficiently high.

The diversity measure we use is

$$F_{\mathrm{div}} = \frac{1}{s'} \sum_{i=1}^{s'} \sum_{j=1, j\neq i}^{s'} d_{\mathrm{h}}^{\mathrm{f}}(i,j). \tag{6.8}$$

An analog definition stands for P_{div}. A cautious implementation also controls the end of the first phase by the number of iterations.

At the end of this step, the most irrelevant features, i.e., those which are selected with a low frequency, are discarded from the candidate set. This filter contributes

to making the GA selection easier and particularly faster. The feature selection task is not completed: the remaining features are likely to be redundant. The final selection step is carried out using a local way. In this step, the convergence phase, an elitist approach is preferred to select an accurate solution, diversity remaining controlled by the archive population, and the GA is combined by local search procedures.

6.3.2 Local Tuning

As previously stated, recent literature reports that GA are not easy to tune when dealing with large systems. The objective of a GA is twofold: space exploration and solution tuning. Reaching both of these objectives may prove difficult. The hybrid part of the algorithm is devoted to helping the GA in the tuning phase. Thus, the GA is in charge of the space exploration, it is likely to find a set of acceptable solutions, and the local procedures aim at improving these solutions by an exhaustive search in their neighborhood.

Of course, extensive search is time consuming, and local tuning has to be applied carefully, only when the expected gain is higher than the cost.

The local tuning includes two different phases: an ascending and a descending procedure.

The ascending phase aims at aggregating new elements, features or prototypes, in a given chromosome while the goal of the descending phase is, on the contrary, to remove features or prototypes from the chromosome description. Both procedures are random free. They are based on the population yielded by the GA.

Let us first consider the ascending step. It can be applied to the feature or the prototype space. Let S' be the set of chromosomes whose fitness score is higher than a given threshold, and $S'_1 \subseteq X$ (respectively, $S'_2 \subseteq Z$) be the set of features (respectively, prototypes) included in at least one chromosome (from S') description.

The ascending procedure consists, for each chromosome in S', in aggregating each of the features in S'_1 (respectively, each of the prototypes in S'_2) to the chromosome and selecting the ones that improve the classification results. The process is repeated until no improvement is possible or a maximal number of ascending iterations is reached.

It should be mentioned that the number of features and prototypes to be tested is reasonably small as some features have been discarded by the first phase of the GA, and among the others, only those which are currently part of one of the best chromosomes are used. This remark highlights the complementary roles played by the GA and the local approach. However, depending on the evolution stage, the cardinalities of S'_1 and S'_2 may be important. In this case, in order to control the ascending procedure computational cost, the number of features or prototypes tested by the procedure is limited. The selected ones are randomly chosen in S'_1 or S'_2. S' may also be of high cardinality and contain redundancy. It is particularly true when the current population is near a local convergence. Then, only a limited number of

chromosomes are submitted to the ascending procedure. They are randomly chosen in S'.

The descending phase is only applied to the S' set. For each chromosome each of the selected features (respectively, prototypes) is removed if its removal does not affect the classification results while improving the fitness function.

In order to save time, ascending and descending procedures are carried out periodically within the so called "convergence phase."

Different strategies are likely to yield comparable results. In our implementation, the convergence phase is organized as a sequence of the following operations:

1. A tuning phase including an ascending procedure followed by a descending one: The preferred options aggregate new prototypes and remove features as the lower the feature dimension space the better the interpretability. This complete mode is quite expensive, it is run with a large period.
2. A tuning phase with a descending procedure in only one space: The feature and prototype spaces are alternatively managed.
3. A pure GA.

The prototype selection is not only based on classification results, it also takes into account the prototype status within the training set in order to avoid selecting either noisy or superfluous prototypes and favor the selection of critical ones.

Although there is general agreement about the basic idea, the implementation of each of these concepts remains an open topic. Many research teams have proposed their own definition, and some of them include reasoning rules. We opt for a simple and unambiguous definition.

A prototype is said to be noisy if none of its k nearest neighbors is of its class. That means this prototype is not able to correctly classify any of its neighbors. The value of k, $k > 1$, is set according to the class cardinality, the higher k the lower the number of prototypes likely to be removed. On the other hand, a prototype is said to be superfluous if all of its neighbors are of its class. That means its neighbors remain well classified after its removal. The amount of filtering depends only on the number of neighbors, k, no additional heuristic is needed.

These concepts, noisy or superfluous prototypes, are highly dependent on the feature space and the distance in use. Thus, their implementation requires a specific management: A list of prototypes to be discarded is attached to a given feature space. The identification of noisy or superfluous prototypes is carried out at the beginning of the descending procedure. The prototypes part of the list are no longer available, neither for the ascending procedure nor GA selection or chromosome generation.

Note that GA may select superfluous prototypes as they improve classification results. In this case the solution found can be considered as good but remains a suboptimal one. The local approach aims, in this case, at increasing the convergence speed toward an optimal solution.

6.4 Results and Discussion

The proposed hybrid GA (HGA) as a nearest prototype classifier is now applied to various synthetic and real world data sets. The results are compared with other approaches.

6.4.1 The Data Used

To test the proposed method, seven data sets are used: five are available in repositories [37] and some results are already known, the two others are from the chemometric area.

The database characteristics are summarized in Table 6.1.

It is needless to introduce the Iris and the Wisconsin breast cancer data, they have been widely used by many machine learning algorithms.

The Gaussian8D is a synthetic data set. It has been designed as a benchmark to study classifier behavior when the classes overlap. The first class is represented by a multivariate normal distribution with zero mean and standard deviation equal to 1 in each dimension, for the second class the mean is also 0, but the standard deviation is 2 for each input variable.

The Satimage data, from ELENA a European Esprit project database, are generated from Landsat Multispectral scanner images. This set has previously been used to design a nearest neighbor classifier [32].

The texture database is also from ELENA. It has been created in order to study the texture discrimination capabilities of algorithms. The attributes are texture features estimated in four directions (0°, 45°, 90°, and 135°) and taking into account the spatial dependency of a given pixel with its two nearest neighbors.

As shown in Table 6.1, the chemometric databases are usually more difficult to manage. The goal is to select a reduced but representative set of relevant compounds from chemical databases in order to investigate how they can be used to design new products.

The first chemometric database describes how product toxicity affects a kind of fish (the fathead minnow). The product descriptors are parameters related to the 2-D molecular structure. They are of different kinds: topological, electrotopological, constitutional, physical, chemical, etc. The number of toxicity levels is 8.

The second one was extracted from the CIPSLINE database. The latter is composed of more than 4,500 compounds coming from various medicinal and drug data

Table 6.1. Database characteristics

	Iris	Breast	Gaussian8D	Satimage	Texture	Chemo1	Chemo2
# items	150	699	1,000	1,028	989	566	1,294
# features	4	10	8	36	40	166	167
# classes	3	2	2	6	11	8	8

reports, published between 1988 and 1999. The records include the molecular and chemical structures, the pattern's name and information about the sources. We used the subset of this database relevant for eight anticancerous properties.

6.4.2 Comparison with Known Algorithms

Three basic genetic strategies are implemented: determinist crowding, elitism, and RTS. They are combined with either an aggregative fitness function or a Pareto-based evaluation [33].

The elitism strategy implementation is the one described in Kuncheva [31]: The subset of children created after genetic operations compete with their parents. The best of the whole set, parents and children, survive in the next generation.

Within the deterministic crowding [38] and RTS [39] frameworks the update of the population also aims at maintaining diversity. The new chromosome has to be better than the old one but additional mechanisms ensure that it is also similar to the one to be replaced.

We also compare two fitness sharing strategies [40] combined with the RTS approach. This technique penalizes the fitness function when the diversity among the population becomes poor. To assess diversity, two measures are used leading to two types of similarity. Genotype similarity uses the hamming distance while the phenotype one is based on the Euclidean distance between the normalized space dimension, whatever the selected features or prototypes. It only takes into account $|S_1|$ and $|S_2|$. For both strategies, a common triangular sharing function has been considered. The width was $0.1 * (f + p)$ for the genotype similarity and 0.05 for the phenotype one.

The multihill climbing algorithm [30] is also tested.

Finally, the ten selected algorithms are:

1. DA: determinist crowding – aggregation
2. DP: determinist crowding – Pareto
3. EA: elitism strategy – aggregation
4. EP: elitism strategy – Pareto
5. RA: restricted tournament selection (RTS) – aggregation
6. RP: RTS – Pareto
7. RGA: RTS – fitness sharing genotype – aggregation
8. RPA: RTS – fitness sharing phenotype – aggregation
9. MHA: multihill climbing - aggregation
10. HGA: hybrid GA

From each data set, ten training and test samples are randomly generated. The training set is made up of about 80% of the data set, the remaining 20% being the test set. For each of the ten samples, the training data are centered and normalized, the computed coefficients are applied to the corresponding test set. Then for each of the ten algorithms, a classifier is designed using the training data and its performance is assessed over the test data.

The same very common genetic parameters have been chosen whatever the database, the main features are listed below:

- Initial population: random bit generation with $\text{prob}(0) = 0.8$ and $\text{prob}(1) = 0.2$
- Crossover probability: 0.5
- Mutation probability: 0.05
- Fitness function: $w_c = 0.4$, $w_p = w_f = 0.3$
- Constants used for λ computation: $f_{\max} = \min(30, f)$, $p_{\max} = 0.3p$
- Number of generations: 300 for the first nine algorithms.

For the HGA, these specific parameters are used:

- Initial population: $a_1 = 0.1$, $a_2 = 0.5$, and $a_3 = 50$
- Number of generations: 80
- Number of local search loops: min = 30, max = 50
- Fitness score threshold to apply local optimization: from 0.7 (two classes) to 0.5 (11 classes)
- Repartition of local procedures: 20% ascending/descending, 50% descending on only one space, 30% pure GA
- Maximum number of chromosomes selected for the ascending procedures: 25
- Maximum number of ascending/descending iterations: 10/5
- Maximum number of prototypes/features tested: 30/30
- Diversity index: $n_f = 1$, $n_p = 0.1p$, $P_s = 0.75p$
- Prototype selection: $k_i = \min(4, 0.5\sqrt{p_i})$, p_i being the number of patterns from class i.

The fitness function of Pareto-based evaluation considers both dominated and nondominated solutions as proposed in [33]. The score of a given chromomose, Y, results from the difference of two terms: The number of chromosomes which are dominated by Y and the number of chromosomes which dominate Y.

6.4.3 Main Results

For each database, each of the nine algorithms is compared with our HGA. The comparison between two algorithms is achieved through their best chromosome. When the HGA is better, the score of the experiment is 0.1, when the results are equal the score is 0 and when the alternative algorithm is better, the score is -0.1.

The overall results are given in Table 6.2. For each of the alternative algorithms the first row corresponds to the cumulative sum over the ten trials when the comparison is made in the Pareto space. In this case, the nondominated solution which dominates the highest number of chromosomes in the final population has been selected. It is quite easy to get "equal" results as in order to win, an algorithm has to be better than the other for at least one of the three objectives and as good as the other for the remaining ones. The second line compares the results using the aggregative function.

These results highlight that the HGA gives better results than any of the other algorithms as the score is always higher than 0. This is especially true when the

Table 6.2. Overall results

	Iris	Breast	Gaussian8D	Satimage	Texture	Chemo1	Chemo2
DA	0.7 1	0.8 1	0 1	0.2 1	0.9 1	0.7 1	0 1
DP	0.8 1	0.8 1	0.1 1	0.2 1	1 1	0.9 1	0 0.3
EA	0.9 0.9	1 1	0.2 1	0.5 1	0.4 0.6	1 1	0 1
EP	0.3 1	0.3 1	0.1 0.1	0 0.1	0.1 0.1	0.2 0.1	0.1 0.1
RA	0.9 0.9	1 1	0.1 1	0.3 1	0.7 1	1 1	0 1
RP	0.9 0.9	0.9 1	0 1	0.7 1	0.7 1	1 1	0 1
RGA	0 1	0 1	0 1	0.3 1	0 1	0.9 1	0 1
RPA	0.3 1	0.4 1	0.2 0.3	0.3 1	0.5 1	1 1	0 1
MA	1 1	0.5 1	0.2 1	0.1 1	0.6 1	1 1	0 1

comparison is done according to the aggregative function. In this case, most of scores are close to 1.

Let us note that the final score may include compensation operations, e.g., a 0.4 score can be the result of $0.6 - 0.2 + 0$, meaning the alternative algorithm has given best results for two of the ten experiments, the hybrid algorithm won six times while for the two remaining samples the results were comparable. This does not occur in our case, as there is no negative contribution in any of the scores.

In Table 6.3, the best chromosomes yielded by each of the ten algorithms for the same, randomly chosen, experiment are compared. The first row indicates the number of selected features for each data set, the second row shows the number of selected prototypes while the third row gives, for each algorithm, the classifier accuracy assessed over the test set.

Note that the use of a Pareto-based fitness function does not improve the results as can be seen for each of the tested algorithms (DP vs DA, etc.) The classification rates are at most comparable but the number of prototypes and features are much higher.

As expected, the simplest bases, Iris and breast cancer, are more or less equally managed by all the systems, though it should be noted that the number of prototypes selected by HGA for the breast cancer data is much smaller (7) than the one corresponding to the other algorithms (from 31 to 110).

Concerning all the other, nontrivial, data sets, the space dimensionality, either the feature or the pattern one, is significantly smaller using HGA for a comparable accuracy. The main difference resides in the prototype selection for medium databases,

Table 6.3. Detailed results for one experiment

	Iris	Breast	Gaussian8D	Satimage	Texture	Chemo1	Chemo2
DA	1 3 0.961	2 31 0.945	2 73 0.726	3 62 0.8	4 79 0.818	22 53 0.565	37 126 0.8
DP	1 6 0.947	1 31 0.9	2 79 0.718	2 123 0.726	4 101 0.816	40 65 0.501	40 151 0.786
EA	1 3 0.966	1 31 0.913	2 71 0.713	3 80 0.8	4 86 0.825	16 48 0.587	36 100 0.807
EP	1 7 0.946	2 110 0.89	1 151 0.59	9 180 0.792	11 186 0.79	39 93 0.456	29 197 0.688
RA	1 3 0.966	1 32 0.9	2 78 0.716	3 59 0.825	4 89 0.855	9 35 0.547	41 117 0.818
RP	1 3 0.966	1 37 0.9	2 80 0.712	3 91 0.793	3 91 0.786	31 51 0.538	36 120 0.782
RGA	1 7 0.964	1 38 0.901	2 80 0.712	2 96 0.754	4 88 0.83	28 56 0.56	41 122 0.793
RPA	1 5 0.965	1 37 0.914	2 171 0.736	3 70 0.776	3 100 0.832	23 60 0.54	42 117 0.79
MA	1 15 0.942	2 110 0.912	2 149 0.654	6 149 0.811	6 143 0.746	21 93 0.555	32 90 0.713
HGA	1 3 0.966	1 7 0.935	2 11 0.778	2 11 0.812	3 28 0.879	2 26 0.649	3 34 0.731

such as Gaussian8D, Satimage, or texture. Considering large and difficult, data sets, such the chemometric ones, the feature space is also notably reduced.

These differences between HGA and the other algorithms can be accounted for the hybridization process, but we however think that the other mechanisms and concepts incorporated in the GA itself play an important role.

For the Chemo2 base, the HGA accuracy is not very good, this is due to the standard weights used in the fitness function which favor space dimension as $w_f + w_p > w_c$.

When dealing with this difficult base, the user wishing to design an accurate classifier would prefer $w_f = w_p = 0.25$ and $w_c = 0.5$. Using this configuration the results, for the same experiment, are as follows:

- Accuracy rate: 0.8
- Number of selected features: 7
- Number of prototypes: 37

6.4.4 Complementary Results

Once achieved the comparison with well known algorithms, let us now try and assess the efficiency of the additional procedures included in the proposed GA. Three of them are studied: the hybrid component, the breaking mechanism, and the prototype status (superfluous or noisy) identification. This lead to the following four configurations:

1. V0: HGA without neither hybridization nor breaking process
2. V1: HGA without hybridization
3. V2: HGA without prototype status
4. V3: HGA

Two configurations do not include the hybrid component, the difference between both being the breaking mechanism. The two remaining ones differ by the presence of the prototype identification step.

This experiment is carried out with the ten samples of Chemo2, the most difficult data set. Common parameters are set for these trials. The aggregative weights are $w_c = 0.5$, $w_f = w_p = 0.25$. The number of generations is set in order to ease the comparison, instead of using the automatic transition between phases: 80 for the preliminary phase and 220 for the convergence one. The initial population is generated fusing the following parameters: $a_1 = 0.02$, $a_2 = 0.2$, and $a_3 = 30$. The hybrid specific parameters are: 10% ascending/descending, 40% descending on only one space, 50% pure GA.

For one experiment Figs. 6.1 and 6.2 show the evolution of the average values for classification rate and the number of selected prototypes of all the chromosomes in the archive population, pop_a.

Table 6.4 shows the final overall results (mean and standard deviation) over the 10 runs.

The hybridization effect is clearly visible either from the classification or the prototype selection point of view: The trend sharply changes after a small number of iterations in the convergence phase. The influence of the prototype status identification process is obvious in Fig. 6.2 when watching the difference between $V2$ and $V3$ configurations. The complexity model is then reduced, this does not improve the classifier generalization ability.

Comparing $V0$ and $V1$ configurations, we may conclude that the breaking mechanism is efficient. The improvement of the classification results (5% in average) may be due to the diversity constraint managed by this breaking mechanism. With a small size population only a part of the huge space is explored, and then this kind of mechanism seems to be relevant without any additional computation cost.

The results may vary from one experiment to another due to the random part of the process. Even so, as shown in Table 6.4, the overall trend indicates that the proposed additional mechanisms are suitable to improve the GA performance.

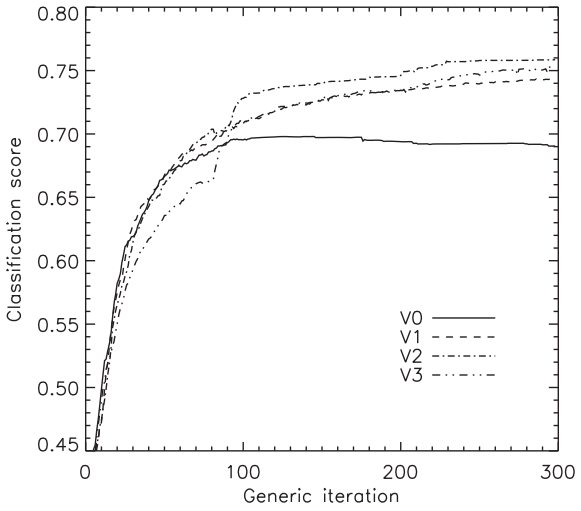

Fig. 6.1. Classification rate evolution for one Chemo2 sample

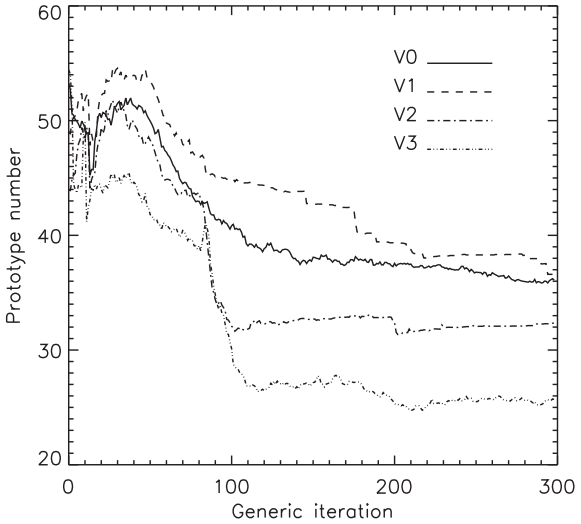

Fig. 6.2. Number of selected prototypes evolution for one Chemo2 sample

Table 6.4. Compared ten run results with Chemo2

	V0	V1	V2	V3
$C_{1\text{nn}}(Z)$	0.710 ± 0.028	0.730 ± 0.022	0.747 ± 0.029	0.742 ± 0.024
$\lvert S_1 \rvert$	6.2 ± 0.9	5.9 ± 1.2	6.0 ± 0.5	6.1 ± 0.6
$\lvert S_2 \rvert$	38.5 ± 4.2	42.3 ± 5.3	31.3 ± 4.2	27.2 ± 2.9

6.5 Conclusion

In order to build an efficient nearest neighbor classifier, three objectives are considered: Find the smallest subsets of features and prototypes that maximize the classification accuracy. As these objectives are not at all independent and are conflicting ones, the optimization problem is usually managed by heuristic approaches. The procedure has to achieve a twofold task: A space exploration to find good solutions and a fine tuning to converge to local optimal solutions.

Setting the parameters to ensure a trade-off between these two tasks within a reasonable time is difficult. Exploratory strategies (crowding, sharing, etc.) require a lot of resources to give a good solution in high dimensional problems, while elitism-based strategy may ignore interesting parts of the space.

The hybrid algorithm proposed in this paper, aims at maintaining both qualities of a genetic population, namely diversity and elitism. Diversity is partially ensured by a breaking mechanism to regenerate the chromosomes while an evolutive memory combined with a local search procedure handles the elitist part of the task.

The whole process is optimized by dividing the algorithm into two self-controlled phases with dedicated objectives. The preliminary phase is pure GA. Based on a RTS scheme, it focuses on exploration. The second phase, which includes local search, is more elitist.

As proved by the results, this algorithm is likely to give satisfactory results within a reasonable time when dealing with medium size data sets.

In order to manage larger databases, some improvements are needed to make the method more tractable. The fitness function computation is expensive, requiring a lot of distance calculations and sorting procedures. Further work should investigate how clustering or space partitioning techniques [41–43] could help in time saving.

References

1. Guyon I and Elisseeff A (2003) An introduction to variable and descriptor selection. Journal of Machine Learning Research 3: 1157–1182
2. Piramuthu S (2004) Evaluating feature selection methods for learning in data mining application. European Journal of Operational Research 156: 483–494
3. Dash M and Liu H (1997) Feature selection for classification. Intelligent Data Analysis 1: 131–156
4. Dash M, Choi K, Scheuermann P, and Lui H (2002) Feature selection for clustering-a filter solution. In Proceedings of the second International Conference on Data Mining 115–122
5. Kohavi R and John G (1997) Wrappers for feature subset selection. Artificial Intelligence 97(1–2): 273–324
6. Dasarathy B V, Sanchez J S, and Townsend S (2003) Nearest neighbor editing and condensing tools-synergy exploitation. Pattern Analysis & Applications 3: 19–30
7. Francesco J F, Jesus V, and Vidal A (1999) Considerations about sample-size sensitivity of a family of edited nearest-neighbor rules. IEEE Trans. On Syst. Man and Cyber. 29(4) Part B

8. Aha D, Kibler D, and Albert M K (1991) Instance-based learning algorithms. Machine Learning 6: 37–66
9. Brighton H and Mellish C (2002) Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery 6: 153–172
10. Hart P E (1968) The condensed nearest neighbor rule. IEEE Trans. Inf. Theory 16: 515–516
11. Gates G W (1972) The reduced nearest neighbor rule. IEEE Trans. Inf. Theory 18(3): 431–433
12. Swonger C W (1972) Sample set condensation for a condensed nearest neighbour decision rule for pattern recognition. Watanabe S (Ed.), Academic, Orlando, FA 511–519
13. Bezdek J C and Kuncheva L I (2000) Nearest prototype classifier designs: an experimental study. International Journal of Intelligent Systems 16(12): 1445–1473
14. Wilson D R and Martinez T R (2000) Reduction techniques for instance-based learning algorithms. Machine Learning 38(3): 257–286
15. Eiben A E, Hinterding R, and Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3(2): 124–141
16. Shisanu T and Prabhas C (2002) Parallel genetic algorithm with parameter adaptation Information Processing Letters, (82): 47–54
17. Wen-Yang L, Tzung-Pei H, and Shu-Min L (2004) On adaptating migration paramaters for multi-population genetic algorithms. IEEE Int. Conf. Syst. Man and Cyber. 6: 5731–5735
18. Lozano M, Herrera F, and Cano J R (2007) Replacement strategies to preserve useful diversity in steady-state genetic algorithms. Information Science
19. Martinez-Estudillo A, Hervas-Martinez C, Martinez-Estudillo F, and Garcia-Pedrajas N. (2006) Hybrid method based on clustering for evolutionary algorithms with local search. IEEE Trans. Syst. Man and Cyber. Part B 36(3): 534–545
20. Hart W E (1994) Adaptative global optimization with local search. PhD Thesis, University of California, San Diego
21. Land M W S (1998) Evolutionary algorithms with local search for combinatorial optimization. PhD Thesis, University of California, San Diego
22. Knowles J D (2002) Local Search and hybrid evolutionary algorithms for pareto optimization. PhD Thesis, University of Reading, UK
23. Carlos A. Coello Coello (1999) A Comprehensive Survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information Systems 1(3): 129–156
24. Moscato P (1999) Memetic algorithms: A short introduction. Corne D, Glover F, and Dorigo M (eds.), New Ideas in Optimization, McGraw-Hill, Maidenhead, 219–234
25. Merz P (2000) Memetic algorithms for combinatorial optimization problems: fitness landscapes and effective search strategies. PhD thesis, University of Siegen
26. Krasnogor N and Smith J (2002) A memetic algorithm with self-adaptive local search: TSP as a case study. Proceedings of the 2002 ACM symposium on Applied computing, Madrid, Spain, 178–183
27. Krasnogor N (2002) Studies on the theory and design space of memetic algorithms. Thesis University of the West of England, Bristol
28. Tamaki H, Mori M, Araki M, and Ogai H (1995) Multicriteria optimization by genetic algorithms: a case of scheduling in hot rolling process. Proceedings of the third APORS 374–381
29. Zhang H and Sun G (2002) Optimal reference subset selection for nearest neighbor classification by tabu search. Pattern Recognition 35: 14811–490

30. Shalak D.B (1994) Prototype and feature selection by sampling and random mutation hill climbing algorithms. In Proceedings of the Eleventh International Conference on Machine Learning, Morgan Kaufman, New Brunswick 293–301
31. Kuncheva L I and Jain L C (1999) Nearest neighbor classifier: simultaneous editing and descriptor selection. Pattern Recognition Letters 20 (11–13): 1149–1156
32. Ho S -H, Lui C -C, and Liu S (2002) Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. Pattern Recognition Letters 23: 1495–1503
33. Chen J H, Chen H M, and Ho S Y (2005) Design of nearest neighbor classifiers: multiobjective approach. International Journal of Approximate Reasoning 40 (1–2): 3–22
34. Zitzler E (2002) Evolutionary algorithms for multiobjective optimization. In Evolutionary methods for design, optimization and Control. Giannakoglou K, Tsahalis D, Periaux J, Papailiou K, and Forgaty T (Eds.), CIMNE, Barcelona, Spain
35. Wiese K and Goodwin S D (1998) Keep-best reproduction: a selection strategy for genetic algorithms. Proceedings of the 1998 symposium on applied computing 343–348
36. Matsui K (1999) New selection method to improve the population diversity in genetic algorithms. Syst. Man and Cyber. IEEE Int. Conf. 1: 625–630
37. Blake C, Keogh E, and Merz C J (1998) UCI repository of machine learning databases [http://www.ics.uci.edi/ mlearn/MLRepository.html]. Department of Information and Computer Science University of California, Irvine, CA
38. Mahfoud S W (1992) Crowding and preselection revisited. Second Conference on Parallel problem Solving from Nature (PPSN'92), Brussels, Belgium 2: 27–36
39. Harik G (1995) Finding multimodal solutions using restricted tournament selection. Proceedings of the sixth International Conferernce on Genetic Algorithms, Eshelman L J (Ed.) Morgan Kaufman, San Mateo, CA 24–31
40. Miller B L and Shaw M J (1996) Genetic algorithms with dynamic sharing for multimodal function optimization. Proceedings of the International Conference on Evolutionary Computation, Piscataway 786–791
41. Cano J R, Herrera F, and Lozano M (2003) Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. IEEE Trans. Evol. Comput. 7(6): 193–208
42. Kim S W and Oommen B J (2003) Enhancing prototype reduction schemes with recursion: a method applicable for large data sets. IEEE Trans. Syst. Man and Cyber. Part B 34(3): 1384–1397
43. Sanchez J S (2004) High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition 37: 1561–1564