International Journal of Pattern Recognition and Artificial Intelligence
© World Scientific Publishing Company

CO-EVOLUTION OF NEAREST NEIGHBOR CLASSIFIERS

CHRISTIAN GAGNÉ* and MARC PARIZEAU

Laboratoire de Vision et Systèmes Numériques (LVSN),
Département de Génie Électrique et de Génie Informatique,
Université Laval, Québec (Quebec), G1K 7P4, Canada.
cgagne@gmail.com, marc.parizeau@gel.ulaval.ca

This paper presents experiments of Nearest Neighbor (NN) classifier design using different evolutionary computation methods. Through multi-objective and co-evolution techniques, it combines genetic algorithms and genetic programming to both select NN prototypes and design a neighborhood proximity measure, in order to produce a more efficient and robust classifier. The proposed approach is compared with the standard NN classifier, with and without the use of classic prototype selection methods, and classic data normalization. Results on both synthetic and real data sets show that the proposed methodology performs as well or better than other methods on all tested data sets.

Keywords: Pattern Recognition; Nearest Neighbor Classification; Prototype Selection; Proximity Measure; Evolutionary Computation; Genetic Algorithms; Genetic Programming; Co-evolution; Multi-objective Optimization.

1. Introduction

Nearest Neighbor (NN) classification[7,11,16], sometimes called instance-based classification, is a widely used technique for pattern recognition and machine learning. It is recognized as a simple yet efficient technique for supervised learning problems with continuous features (attributes). After more than 35 years, NN classifiers are still widely studied and used for solving pattern recognition problems. This type of classifier is also often used as a baseline system relative to which new classifier systems are compared.

On the other hand, Evolutionary Computation (EC)[1] is a promising machine intelligence discipline involving the simulation of natural evolution on computers. It is a generic problem solving paradigm applicable whenever solutions can be represented by some data structure and evaluated by an objective function; the so-called "fitness" function. Populations of solutions – initially random solutions – evolve over time through a sequence of processes that include (natural) selection and different genetic operations. In the end, the fittest individual is chosen as "the" solution to the problem and, although EC systems do not guarantee convergence

[*] Christian Gagné is now with Informatique WGZ Inc., 819 ave. Monk, Québec (Quebec), G1S 3M9, Canada. E-mail: christian.gagne@wgz.ca

to an optimal solution, they have been shown in practice to sometimes outperform other techniques as well as human experts for hard problems[3,28].

In this paper, different EC techniques will be presented to design efficient and robust NN classifiers. Four approaches will be studied: 1) prototype selection with multi-objective Genetic Algorithms (GA)[24,32]; 2) neighborhood proximity measure design with Genetic Programming (GP)[2,27]; 3) cooperative co-evolution of prototype selection and neighborhood proximity design; and 4) competitive co-evolution of fitness evaluation set selection, on the one hand, and cooperative co-evolution of prototype selection and neighborhood proximity design, on the other hand. For each approach, results will be presented and analyzed using both synthetic and real data sets.

The paper is structured as follows. Next, Section 2 proceeds with a summary of the well-known NN classifier. This section emphasizes the three main elements of this classifier, two of which we aim to optimize (prototype selection and neighborhood proximity). The four synthetic data sets used in this work are described in Section 3. These data sets have been designed to highlight different aspects of the classification problem. Then, the four considered approaches are presented in Sections 4, 5, and 6. Section 7 proceeds with results obtained on five problems taken from the UCI machine learning repository[4]. Finally, the approaches and results obtained are discussed in Section 8, followed by some considerations on how EC can contribute to the design of efficient pattern recognition systems.

2. Nearest Neighbor Classification

The design of NN classifiers requires the specification of three distinct elements[16]: a set of prototype vectors, a classification rule, and a neighborhood proximity measure. The prototypes are representative data used by the classifier to assign class labels. The selection of prototypes is a critical operation which involves using the training data set to partition the input vector space based on the neighborhood proximity measure. If the selected prototypes are noisy or not truly representative of the problem at hand, the partition may be inappropriate and lead to low recognition rates. The number of selected prototypes must also be minimized because it has a direct impact on classification time.

The NN classification rule consists essentially in choosing the label associated with the nearest neighbor of an unknown input vector $\mathbf{x}$. More formally, let $P = \{\mathbf{p}_1, ..., \mathbf{p}_m\}$ denote the set of m labeled prototypes and let $\mathbf{p} \in P$ be the prototype nearest to $\mathbf{x}$. According to the basic NN rule, $\mathbf{x}$ is assigned to the class label assigned to $\mathbf{p}$. This is a very straightforward rule which performs well on noise-free data. But for real-life problems, where noise may be omnipresent in the training data, a simple refinement that leads to the so-called k-NN rule, is to consider the k nearest neighbors and to assign to $\mathbf{x}$ the label of the most frequently occurring class label, breaking ties arbitrarily. Increasing the value of k then attenuates the effect of noise but may also soften the class boundaries.

Different proximity measures can be used for k-NN classification. A very common and general one is based on the L_a norm which can be used to measure a distance between two vectors $\mathbf{x} = [x_1 \cdots x_n]^T$ and $\mathbf{y} = [y_1 \cdots y_n]^T$

$$L_a(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^a \right)^{\frac{1}{a}} \tag{1}$$

The most common instantiation of this norm is the Euclidean distance, when $a = 2$:

$$L_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

Two other common cases are the Manhattan distance ($a = 1$):

$$L_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|, \tag{3}$$

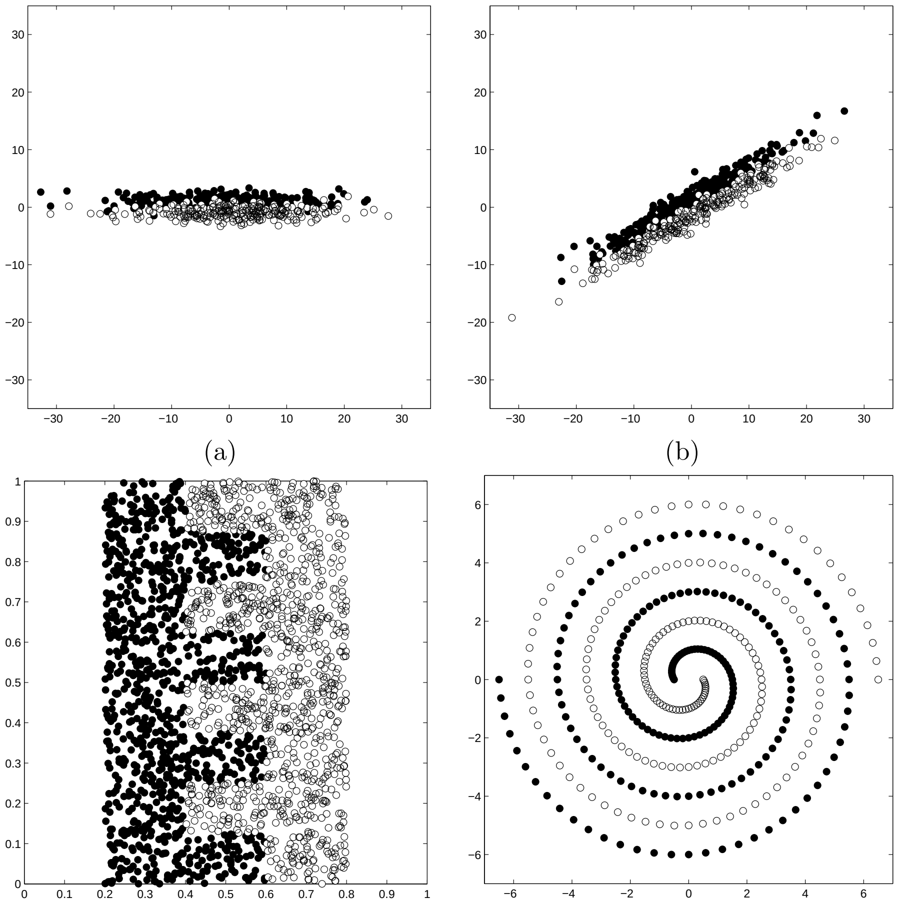
and the L_∞ norm:

$$L_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1}^{n}(|x_i - y_i|) \tag{4}$$

The problem with L_a norms is that the different dimensions of the feature space must somehow be normalized to comparable scales. The choice of a is arbitrary but puts a different emphasis on the larger components of $\mathbf{x} - \mathbf{y}$ relative to the smaller ones. For example, in L_1 the emphasis is the same for all components. Input features should thus have the same scale (units) in order to measure coherent distances. The greater the value of a, the greater the emphasis on the larger components. At the limit, when $a \to \infty$, the measured distance depends only on the single largest component. It is important to realize that the L_a family of norms is just one out of an infinite number of possible neighborhood proximity measures. One of the objectives of this paper is to develop a general approach for automatically discovering the best measure for a given classification problem.

An alternative to changing the proximity measure is to normalize the feature space. For example, a simple scaling transformation of every feature in the interval $[0, 1]$ can help in some circumstances, although it may also distort the data distribution and make discrimination more difficult. A more sophisticated solution consists in applying a whitening transform[16] that first removes any linear dependency between the features by translation and rotation (rigid transform), before applying the scaling transform. Under the hypothesis of multinormal distributions, this whitening transform can make sense, but there is no guarantee that it will enhance the discrimination power of the chosen proximity measure for other class distributions.

Finally, it should be noted that the k-NN classifier has a computational complexity of $O(mn \log k)$ for processing a single unknown input vector $\mathbf{x}$. Thus, there is a definite interest in reducing both the number of neighbors (k) and the size of the prototype set (m) in order to produce more efficient classifiers.



(c) (d)

Fig. 1. Synthetic data sets: (a) overlapped; (b) slanted; (c) jagged; and (d) spirals.

3. Synthetic Data Sets

The synthetic data sets that were designed for the first series of experiments are illustrated in Figure 1. They have a 2D feature space and two classes so that they can easily be visualized on paper. The data sets are:

(1) *Overlapped* — two horizontally elongated multinormal distributions of 250 points each, with an important overlap between classes along the smaller of their two principal axes (optimal Bayesian recognition rate is 76.8%);
(2) *Slanted* — distributions similar to the previous one but slanted by 30-degrees (optimal Bayesian recognition rate is 90.4% in this case);
(3) *Jagged* — two non-overlapping uniform distributions of respectively 971 and 1029 points, but with very jagged boundaries (100% separable);

Table 1. Best 1-NN classifier found when varying the L_a norm ($a = 1, 2, \infty$), and the type of normalization (none, scaling or whitening); using all instances of the training set as prototypes, the best classifier is selected on the evaluation and validation sets, and recognition rates are reported for the test set.

	Baseline	Best Classic Configuration		
Data Set	(L_2)	Distance	Normalization	Rec. Shift
Overlapped	69.6%	L_∞	Scaling	+2.4%
Slanted	80.0%	L_2	Whitening	+8.8%
Jagged	94.2%	L_2	None	0.0%
Spirals	81.4%	L_1	Scaling	−2.0%

(4) and *Spirals* — the classical intertwined spirals with 193 points each.

Each data set has been randomly partitioned into four equal size subsets: a *training* data set, a *fitness evaluation* data set, a *validation* data set and a *test* data set; these subsets are used throughout all experiments with synthetic data. The training set is the one from which the GA will select class prototypes. The fitness evaluation set will be used to evaluate the fitness of individual solutions, for both GA and GP. The validation set is used to select the best-of-generation individual, that is the one with the best generalization capability. Finally, the test set is used strictly at the end of the evolution process to evaluate the performance of the best-of-run solutions (experiments are repeated 10 times).

Table 1 summarizes the baseline performances that can be obtained on these synthetic data sets using a classic 1-NN classifier with an L_2 norm and no normalization of the input feature space. The table also gives the best performance that can be achieved when varying the basic parameters of the 1-NN: use of norm L_1, L_2, or L_∞; either without normalization, or with simple scaling or whitening transforms. Only one neighbor is used throughout all experiments presented in this paper.

Column "Rec. Shift" in Table 1 specifies for each data set the increase in performance that was observed for the best combination of parameters. For instance, for the *Overlapped* data set, a classifier with L_∞ norm and scaling transform performs best with an increase of +2.4% relative to the baseline. In this case, a simple scaling transform seems optimal since the data are elongated and the two features are linearly independent. For the case of the *Slanted* data set, where the two features are linearly dependent, a whitening transform seems optimal since it enables the de-correlation of the features. For the *Jagged* and *Spirals* sets the baseline result is pretty much the best that can be obtained. The negative "Rec. Shift" value for the "Spirals" set can be explained by the fact that the best classifier configuration selected using the evaluation and validation sets (which was performing a little better than the baseline configuration) was in fact performing a little worse than the

```
(1) Let $X = \{\mathbf{x}_1 \cdots \mathbf{x}_m\}$ designate the full training set, shuffled in random order;
(2) Initialize $P = \{\mathbf{x}_1\}$, the initial prototype set, and $X = X - \{\mathbf{x}_1\}$;
(3) For each element $\mathbf{x}_i \in X$ (in random order):
    (a) Let $\theta$ be the class label of the nearest neighbor of $\mathbf{x}_i$ in $P$;
    (b) If $\theta$ disagrees with the label of $\mathbf{x}_i$, then $P = P + \{\mathbf{x}_i\}$ and $X = X - \{\mathbf{x}_i\}$.
(4) If $|X| > 0$ and the condition 3(b) was true at least once, then go back to step 3;
(5) Return $P$ as the condensed prototype set.
```

Fig. 2. *Hart's condensing algorithm for prototype selection.*

```
(1) Initialize $P$, the prototype set, using the complete training set;
(2) For each $\mathbf{p}_i \in P$ (in random order):
    (a) Find in $P - \{\mathbf{p}_i\}$, the $k$ nearest neighbors of $\mathbf{p}_i$;
    (b) Find the majority class label $\theta$ within these neighbors, breaking ties arbitrary;
    (c) If $\theta$ disagrees with the label of $\mathbf{p}_i$, remove $\mathbf{p}_i$ from $P$.
```

Fig. 3. *Wilson's editing algorithm for prototype selection.*

baseline configuration when evaluated on the test set.

In the next three sections, we will study four approaches to enhance the performance of the NN classifier. We will see that it is possible to both reduce the number of prototypes and increase recognition rates.

4. Prototype Selection with a Multi-objective Genetic Algorithm

Prototype selection for NN classifiers is an important topic that has been investigated in the fields of pattern recognition and machine learning. The pioneering work of Hart[21], summarized in Figure 2, consists of a condensing rule for constructing a prototype set. Another classical approach is the editing algorithm of Wilson[44], which attempts to eliminate noisy instances from the prototype set, as described in Figure 3. It is common to apply Wilson's editing followed by Hart's condensing algorithm for maximum prototype set reduction. Table 2 presents recognition and selection rates obtained on the test sets after Wilson's editing, Hart's condensing, and Wilson's editing followed by Hart's condensing, respectively. Results show that Wilson's editing improves recognition rates slightly for the overlapped set, has no effect for the slanted set, and degrades the recognition rate for jagged and spirals

Table 2. Recognition and selection rates obtained on the test sets using Wilson's editing, Hart's condensing and Wilson's editing followed by Hart's condensing; $k = 1$ with L_2 metric and no normalization. Column "Baseline" is the recognition rate achieved with the complete training set used as a prototype set, column "Rec. Shift" is the differential in recognition rate after prototype selection, compared to the baseline, while column "Sel. Rate" is the ratio of the number of selected prototypes over the size of the training set.

		Wilson's editing		Hart's cond.		Wilson + Hart	
	Base-	Rec.	Sel.	Rec.	Sel.	Rec.	Sel.
Data Set	line	Shift	Rate	Shift	Rate	Shift	Rate
Overlapped	69.6%	+0.8%	68.8%	−8.0%	51.2%	0.0%	16.8%
Slanted	80.0%	0.0%	87.2%	+4.0%	28.0%	+3.2%	15.2%
Jagged	94.2%	−0.4%	95.2%	−1.8%	14.8%	−0.4%	9.0%
Spirals	81.4%	−17.5%	79.2%	−5.2%	53.1%	−23.7%	28.1%

data sets. Overall, the reduction in number of prototypes is modest. Hart's condensing is much more aggressive in reducing the number of prototypes but also tends to degrade recognition rates, except for the slanted set. Wilson's editing combined with Hart's condensing reduces the number of prototypes even further, while keeping recognition rates at a level comparable to Wilson's editing alone. In conclusion, these classic prototype selection methods are very efficient at reducing the number of prototypes but are not able to simultaneously enhance recognition rate. The worst performance is for the spirals that are essentially noise free and somewhat under-sampled in the training set (1/4 of the spirals).

Evolutionary Algorithms (EA) have been used previously for prototype selection. Cano et al.[8] present a detailed analysis on the use of different EA for prototype selection, while Ho et al.[23] and Kuncheva and Jain[30] demonstrated the efficiency of GA for simultaneous prototype and feature selection, all of them using a single weighted sum fitness measure. In this work, we purposely chose to conduct only prototype selection, as was done by Cano et al.[8], since feature selection will be indirectly carried out in the following sections when we evolve problem-specific neighborhood proximity measures. Other works on feature selection with GA that can be directly adapted to prototype selection include those of Yang and Honavar[46], and Oliveira et al.[33]. Both used a multi-objective GA for feature selection. Recently, Chen et al.[9] proposed a multi-objective GA for simultaneous prototype and feature selection, in which they use specialized crossover and multi-objective selection operations. They demonstrated that their multi-objective approach was able to generate good results in comparison with a single objective version of a previously proposed system[23].

Fernández and Isasi[18] tackled the prototype selection problem by taking a different route. They evolve a set of prototypes that compete to increase their quality in a local sense. The evolutionary algorithm includes some specific custom genetic operators like Mutate, Reproduce, Fight, Move, and Die, that change both the number of prototypes and their position over time.

A natural GA representation for prototype selection is to use bit strings, where each bit position is associated with a distinct sample of the training set. A value of 1 indicates that the corresponding sample is selected as a prototype, while a 0 signifies that the sample is rejected. There are two optimization objectives: to minimize the number of prototypes and to maximize the recognition rate. These objectives are contradictory as selecting fewer prototypes often leads to lower recognition rates, as observed in Table 2. In population-based multi-objective optimization, there is a set of non-dominated solutions for which no other in the population possess better values for all objectives. Such trade-off solutions are called Pareto optimal[10]. In the objective space, the curve that passes through every Pareto optimal solution is called the Pareto front. For this work, the non-dominated solutions consist of the optimal trade-off points on the Pareto front where there is no solution with both a better recognition rate and a smaller prototype set size.

The single objective fitness measure used by Ho et al.[23] and by Kuncheva and Jain[30] is a weighted sum of the recognition rate and the number of selected prototypes (and features). The use of a multi-objective fitness measure and a natural selection algorithm based on Pareto optimality has the advantage of removing any bias induced by a human-engineered weighted sum, and thus of postponing decision on the best recognition rate / prototypes reduction ratio to the end of the optimization process. Furthermore, it adds a uniform pressure amongst both objectives during the optimization process to remove any early bias, induced by arbitrary weight values, toward a specific region of the Pareto front. Depending on the context, the practitioners can then select a final solution with a different recognition rate / prototype set reduction ratio trade-off.

For each generation, the best-of-generation individual is the one with the highest recognition rate on the fitness evaluation set. At the end of a run, the best-of-run individual is selected as the one with the highest recognition rate on the validation data set among the best-of-generation individuals. This allows the individual that performs the best overall to be selected, without overfitting the data sets. This methodology is used for all subsequent experiments.

Even though this methodology for selecting the best-of-run individual may discard some useful information contained in the multi-objective optimization process, three arguments can be made in its favor. First, as indicated by Occam's razor[15], at equal recognition rates, the simplest solution should be selected. This principle is respected here by the use of the non-dominated criterion such that the best-of-generation individual is the solution with the best selection rate given the achieved recognition rate. Second, caution should be exercised in order to limit the use of the validation set, as the more individuals tested on this set, the higher the probability of overfitting it, as argued by Jensen and Cohen[26]. Finally, solutions in regions of the Pareto front with a strong selection/low recognition trade-off are likely to include the selection of some essential prototypes for classification. It can thus be

Table 3. Prototype selection with multi-objective GA for a 1-NN classifier with L_2 metric and no input feature space normalization.

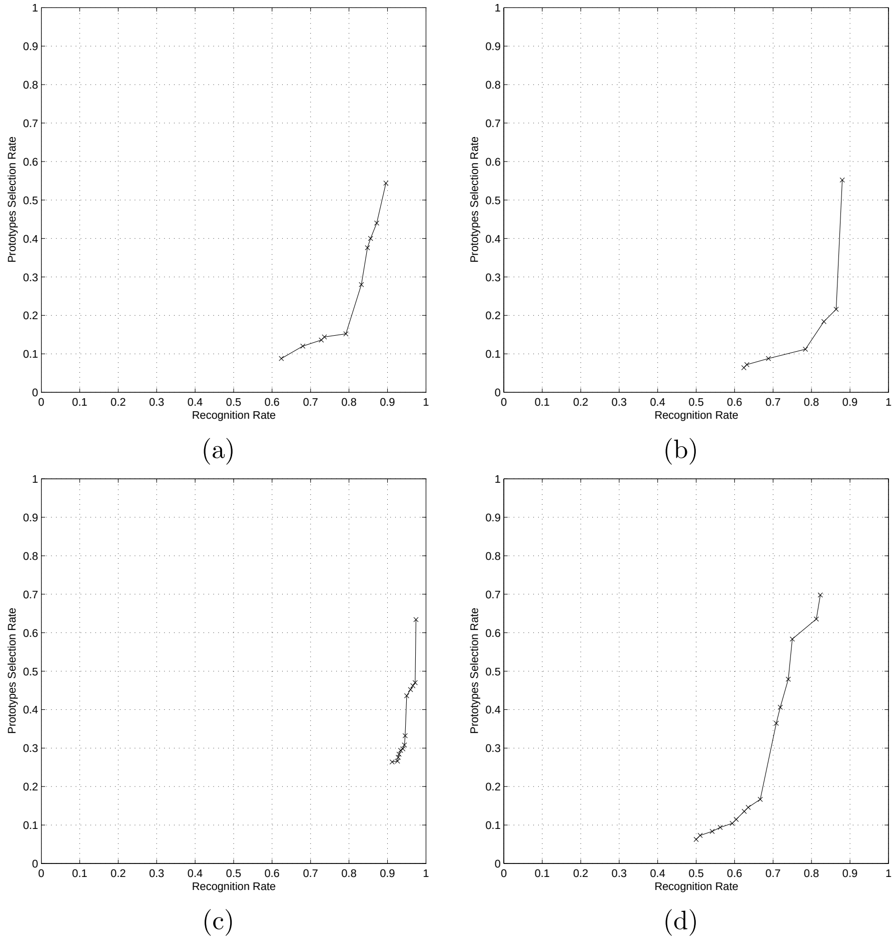
		Best of 10		Mean of 10	
		Rec.	Sel.	Rec.	Sel.
Data Set	Baseline	Shift	Rate	Shift	Rate
Overlapped	69.6%	+2.4%	60.8%	+6.4%	60.0%
Slanted	80.0%	−1.6%	46.4%	−0.8%	67.6%
Jagged	94.2%	+0.6%	66.0%	+0.7%	67.1%
Spirals	81.4%	−4.1%	65.6%	−5.1%	75.4%

speculated that mixing them by crossover with other solutions representing different trade-offs may induce a valuable evolution dynamic, by improving the selection rate of all individuals of the population while maintaining the recognition rate.

In this paper, all software implementations involving EC are based on the Open BEAGLE C++ framework[19,20], freely available on the Internet. Table 3 summarizes the results obtained on our four synthetic data sets when using a multi-objective GA for prototype selection with an L_2 metric and no input feature space normalization. Column "Best of 10" is the result for the best of the best-of-run individuals obtained in 10 independent runs of the GA, while column "Mean of 10" is the average for the same 10 best-of-run individuals. It is important to note that "best of" is determined from the validation set recognition rate, not on the selection rate, nor on the test set recognition rate. It is thus possible to observe lower recognition rates in the "Best of 10" column compared to the "Mean of 10" column, given that the best solution on the validation set is not necessary the best on the test set.

The algorithm used for the multi-objective selection is the Niched Pareto Genetic Algorithm 2 (NPGA2)[17]. The GA parameters are: population size of 1000, 100 generations, NPGA2 tournament selection with 2 individuals and a niche radius (σ_{share}) of 1.0, a priori probability of 0.75 for bits to be initialized with 1, uniform crossover probability of 0.3, and per bit mutation probability of 0.01. Results show significant recognition rate improvements for the overlapped data set. Since this data set is composed of overlapping clouds of points, the prototype selection tends to eliminate samples that are inducing misclassifications. Strangely, the recognition rate is slightly worse than the baseline for the slanted data set. Recognition rate improvements for the jagged set are very small. This is not surprising since the classes do not overlap for this data set. The selection rate found is not as good as for Hart's condensing. For spirals, recognition goes down slightly but the selection rate for this data set is good. Figure 4 presents the Pareto fronts associated with the "Best of 10" results of Table 3. These Pareto fronts illustrate the fact that non-dominated solutions obtained at the end of an evolution cover an important range of recognition/selection rate trade-offs.

In general, results presented here demonstrate that prototype selection can lead



(a)

(b)

(c)

(d)

Fig. 4. Pareto fronts for our "Best of 10" evolutions for (recognition rates are for the fitness evaluation data sets): (a) overlapped data; (b) slanted data; (c) jagged data; and (d) spirals data.

to recognition rate improvements, while attaining sustainable selection rates. Compared to the results obtained with Wilson's editing and Hart's condensing algorithms, recognition rates are preserved and sometimes improved, mainly due to Pareto optimal-based natural selection and the best-of-run selection strategy. On the other hand, prototype selection rates are not comparable to the ones achieved with Hart's condensing algorithm, for the same best-of-run selection strategy. If a practitioner wants to achieve a given recognition/selection ratio, it is possible to trace the Pareto front of solutions and choose the one nearest to the desired trade-off. Results presented in Figure 4 illustrate this point by showing that trade-offs different from those obtained with Wilson's editing and Hart's condensing can be achieved. Moreover, it is clear that the Darwinian search process of GA allows a richer exploration of different prototype selections. As usual with EC, the disadvantages of the approach are its stochastic nature which does not guarantee convergence nor replicability of the results, and the high computational burden of the fitness evaluation process. On an off-the-shelf PC (AMD Athlon 1.2 GHz), it takes about half an hour to evolve such prototype selection with GA, compared to a few seconds for deterministic heuristics such as Wilson's editing and Hart's condensing.

Also, this approach for prototypes selection may not scale very well for large data sets as the number of bits in the bit strings is proportional to the data set size. But more appropriate approaches for large data sets could be devised. For example, variable-length integer-valued vectors could be evolved, where the integers represent indices of the selected prototypes. This representation, combined with a multi-objective minimization of the number of prototypes, would allow better scalability by having integer-valued vectors of size equal to the reduced prototype set size.

5. Genetic Programming of Neighborhood Proximity

Genetic programming enables the evolution of programs using a problem-specific instruction set[2]. In the context of pattern recognition, it can be used to automatically design a problem-specific model of the data. For the NN classifier, this modeling can be done by evolving neighborhood proximity measures in order to maximize recognition rates. This allows great flexibility by going beyond the usual parametric linear modeling of the data.

Relevant literature includes the work of Demiröz and Güvenir[14] who used GA to evolve the weights of each feature in a Euclidean metric space. In Raymer et al.[36], a linear transformation matrix of dimensions $n \times m$ is evolved, where n is the original input space size, m is the transformed input space size, and $m < n$, in order to reduce dimensionality of the feature space used in k-NN classification. This is not an explicit neighborhood proximity evolution, but it is pertinent to this work as it targets the development of a problem-specific model of the data, by applying a linear transformation on the input space. On the other hand, for the application of GP to the engineering of features in the context of pattern recognition, the works of Sherrah et al.[38,39], Bot[6], and Krawiec[29] use GP for the construction of high-level features from raw features, in order to improve the classification rate on a given data set. The function and terminal sets used in Sherrah et al. (1996)[38] and by Bot[6] only allow a linear construction of high-level features from raw ones, while Sherrah et al. (1997)[39] and Krawiec[29] takes advantage of non-linear functions, which can be of great value. Recently, Smith and Bull[40] used a similar approach to the previous ones[6,29,38,39] for high-level features construction with GP, over which a selection of the GP-generated features is done by a bit string GA in order to generate the final features set.

In order to evolve genetic programs with GP[2], the data type and the programming primitives (instructions) must first be defined. It is proposed here that data be either n-dimensional vectors, where n corresponds to the feature space size, or scalars as a special case. To ensure the closure property of our evolved programs, however, all GP primitives are designed to handle combinations of both scalars and vectors. Scalars are simply converted to vectors by copying their value into each of the n vectorial dimensions.

Table 4 enumerates the complete primitive set that was used for all GP evolutions. It includes arithmetic primitives such as $+$, $-$, $\times$ and $\div$; maximum, minimum and average functions applied on two arguments; length metrics such as Manhattan, Euclidean and L_∞; n-to-1 functions that extract the sum, the maximum, or the minimum of a vector's components, and ephemeral random rotation function. The terminals used are ephemeral random binary vectors; ephemeral random scalars in the range $[-1, 1]$; maximal, minimal, and average vector of the prototype set; specific prototype input vectors; and unknown input vectors. When the result received from the root of a program is a vector of size n, this vector is transformed into a scalar using the L_1 norm.

A multi-objective fitness measure is used for the neighborhood proximity evolution to simultaneously maximize recognition rate and minimize tree size. This approach has the advantage of favoring small trees to promote the emergence of simple models of the data. Common intuition is that simple models are usually more general. Iba et al.[25] proposed to use the number of nodes in the GP trees as an estimator of their complexity based on the minimum description length principle[37]. Another important advantage is to contain the bloat phenomenon[2,42], which tends to make the trees larger and larger, from generation to generation. The approach taken here in effect restricts code growth by the use of multi-objective selection, much like it is done in Langdon[31], Bleuler et al.[5], and de Jong et al.[13], which all use the GP program size as an objective to minimize. Just as for prototype selection with GA, the best-of-run individual is selected as the one with the highest recognition rate on the validation set using the best-of-generation individuals found for the fitness evaluation set.

Results for GP engineered neighborhood proximity measures are given in Table 5 for our four synthetic data sets. Parameters used for the experiments are: population size of 1000, 100 generations, NPGA2 selection tournaments with 2 participants and a niche radius (σ_{share}) of 1.0, crossover probability of 0.9, shrink mutation probability of 0.05, node swap mutation probability of 0.05, sub-tree mutation probability of 0.05 and maximum tree depth of 17. Results show good improvements in recognition rates for overlapped, slanted and spiral data, compared to baseline results. Compared with results obtained with the best classic configurations presented in Table 1, improvements are significantly better using the evolved proximity measures for the spiral data set, slightly better for the overlapped set, but a little worse for the slanted data set. The latter result stems from the fact that, in this case, the whitening transform used by the best standard 1-NN is already optimal for this data set.

Table 4. Primitives used for the GP evolutions of neighborhood proximity measures ($\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ represent n dimensional vectors; scalars are automatically converted to vectors).

Name	Args	Description
ADD	2	Vector addition: $\text{ADD}(\mathbf{x},\mathbf{y}) = \mathbf{x}+\mathbf{y}$.
SUB	2	Vector subtraction: $\text{SUB}(\mathbf{x},\mathbf{y}) = \mathbf{x}-\mathbf{y}$.
MUL	2	Vector dot-multiplication: $\text{MUL}(\mathbf{x},\mathbf{y}) = \mathbf{z}$, with $z_i = x_i \times y_i, \forall i \in \{1,\ldots,n\}$.
DIV	2	Protected vector dot-division[27]: $\text{DIV}(\mathbf{x},\mathbf{y}) = \mathbf{z}$, with $z_i = x_i/y_i, \forall i \in \{1,\ldots,n\}$ if $y_i \notin [-0.001, 0.001]$, and $z_i = 1.0$ otherwise.
MAX	2	Vector dot-maximum: $\text{MAX}(\mathbf{x},\mathbf{y}) = \mathbf{z}$, with $z_i = \max(x_i, y_i), \forall i \in \{1,\ldots,n\}$.
MIN	2	Vector dot-minimum: $\text{MIN}(\mathbf{x},\mathbf{y}) = \mathbf{z}$, with $z_i = \min(x_i, y_i), \forall i \in \{1,\ldots,n\}$.
MEA	2	Vector average: $\text{MEA}(\mathbf{x},\mathbf{y}) = (\mathbf{x}+\mathbf{y})/2$.
MAN	1	L_1 norm of a vector: $\text{MAN}(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert$.
EUC	1	L_2 norm of a vector: $\text{EUC}(\mathbf{x}) = (\sum_{i=1}^{n} (x_i)^2)^{0.5}$.
LIF	1	L_∞ norm of a vector: $\text{LIF}(\mathbf{x}) = \max(\lvert x_1 \rvert, \ldots, \lvert x_n \rvert)$.
SUM	1	Sum of vector components: $\text{SUM}(\mathbf{x}) = \sum_{i=1}^{n} x_i$.
MXV	1	Maximum vector component: $\text{MXV}(\mathbf{x}) = \max(x_1, \ldots, x_n)$.
MIV	1	Minimum vector component $\text{MIV}(\mathbf{x}) = \min(x_1, \ldots, x_n)$.
ROT	1	Perform random rotation of a vector. If the argument is a scalar, return it unchanged. Otherwise generate a random rotation matrix using $n-1$ angles randomly generated in interval $[0, \pi]$, and return the corresponding rotated vector. Rotation matrices are generated either during initialization or through mutations, in a similar way as the ephemeral random constant[27].
EBV	0	Vector of binary ephemeral random constants[27]: $\text{EBV} = \mathbf{z}$, with $z_i \in \{0,1\}, \forall i \in \{1,\ldots,n\}$.
ESC	0	Vector of ephemeral random constants[27]: $\text{ESC} = \mathbf{z}$, with $z_i \in [-1,1], \forall i \in \{1,\ldots,n\}$.
PMX	0	Maximal vector of the prototype set; in 2D, this is the upper right corner of the data set bounding box.
PMI	0	Minimal vector of the prototype set; in 2D, this is the lower left corner of the data set bounding box.
PME	0	Average vector of the prototype set.
P	0	Instance of the prototype set.
I	0	Unknown input vector value.

Table 5. Performance of GP engineered neighborhood proximity measures for the 1-NN classifier on synthetic data sets (input feature space not normalized).

		Best	Best of 10	Mean of 10
Data Set	Baseline	Classic	Rec. Shift	Rec. Shift
Overlapped	69.6%	+2.4%	+8.0%	+6.8%
Slanted	80.0%	+8.8%	+4.8%	+5.3%
Jagged	94.2%	0.0%	+0.6%	+0.6%
Spirals	81.4%	−2.0%	+8.3%	+6.1%

The following s-expression is an example of a distance measure obtained by GP:

```
(MAN (MEA P (DIV I PME)))
```

It corresponds to the overall best solution found for the overlapped set. The value of the `PME` terminal for that training set being $[-0.382\ \ -0.00781]^T$, the expression can be re-interpreted as follows:

$$\mathrm{d}_{\text{overlapped}}(\mathbf{x}, \mathbf{p}) = \left|\frac{p_1 - 2.62x_1}{2}\right| + \left|\frac{p_2 - 128x_2}{2}\right|,$$

where the relative weight of the second data component is almost 50 times larger than that of the first component, effectively discarding the latter in favor of the former for classification decision.

6. Co-evolution of Nearest Neighbor Classifiers

Section 4 presents a method for the design of 1-NN classifiers by a selection of prototypes using a multi-objective GA. Follows in Section 5 a distinct method for the design of neighborhood proximity measure with GP. These two approaches are totally independent; it is possible to select prototypes for any kind of neighborhood proximity, and vice versa. But what about simultaneous prototype selection and neighborhood proximity evolution? This is possible through *co-evolution*.

Co-evolution in the context of EC can be defined as several populations, sometimes called species, evolving simultaneously with interaction between them such that the fitness evaluation operation of one species depends on the evolution state of the other. Co-evolution is generally classified into two main categories: 1) competitive[22], where the co-evolving species are guided by opposite goals, thus the good performance of one species hurting the performance of the other species; and 2) cooperative[35], where the species are evolving together in order to solve different parts of the same problem.

For this work, we examine both strategies. First, there is cooperative co-evolution of prototype selection using a multi-objective GA, as in Section 4, in conjunction with a GP engineered neighborhood proximity measure, as explained in Section 5. The fitness evaluation for the prototype selection species is done by

Table 6. Co-evolution of prototype selection and neighborhood proximity for 1-NN classifiers. Results marked by a † (data set Spirals) are not statistically different from those of the baseline according to a 95% confidence Student's *t*-test.

			Best of 10		Mean of 10	
		Best	Rec.	Sel.	Rec.	Sel.
Data Set	Baseline	Classic	Shift	Rate	Shift	Rate.
Overlapped	69.6%	+2.4%	+8.8%	55.2%	+6.6%	54.6%
Slanted	80.0%	+8.8%	+8.0%	65.6%	+2.2%	65.5%
Jagged	94.2%	0.0%	+0.6%	64.8%	+0.9%	64.0%
Spirals†	81.4%	−2.0%	+8.3%	72.9%	+1.7%	70.0%

computing the recognition rate for each individual using the best proximity measure of the previous generation. For the proximity measure species, this fitness is computed from the recognition rate for each individual using the best prototype set of the previous generation. For the first generation, reference individuals are selected randomly in the respective species. This approach can be qualified as optimistic collaboration credit assignment[43].

Results of the co-evolution of prototype selection and neighborhood proximity are presented in Table 6. Parameters are the same as those presented in previous sections for their respective species. Results show that it is possible to achieve recognition rates comparable to those of Table 5, while removing up to 45% of the initial prototype set.

Second, we introduce a third species that is competitively evolving with the two previous ones. This third species is made of bit strings of the size of the fitness evaluation data set. Each bit in this string indicates whether the corresponding sample in the data set is used or not for fitness evaluation. The number of true bits in each bit string is kept constant. The bit string initialization, uniform crossover, and bit flip mutation operators are modified in such a way that the number of true bits in each string is always preserved. These are small modifications to the classical operations coded by working on bit indices instead of bit values. For instance, the initialization proceeds by drawing randomly, without replacement, as much indices of true bits as prescribed by the problem. The mutation is done by drawing without replacement n indices of true bits to flip and n indices of false bits to flip, where n is equal to the number of true bits of each string times the bit flip mutation probability. Finally, the uniform crossover proceeds by collecting indices of disagreement between bits of the two parent's bit strings and dividing them into two categories, the *false-true* category (first parent bit value is false, second parent bit value true) and the opposite *true-false* category. For each of these two categories, the indices of the bits are distributed equally to generate children such that the number of true bits is preserved.

The fitness measure has a single objective which is to minimize the recognition rate. Individuals of this third species are mated with individuals of the two other

Table 7. Three species co-evolution of cooperative prototype selection and neighborhood proximity against evaluation data subset selection in the context of 1-NN classifiers. Results marked by a † (data set Spirals) are not statistically different from those of the baseline according to a 95% confidence Student's *t*-test.

			Best of 10		Mean of 10	
		Best	Rec.	Sel.	Rec.	Sel.
Data Set	Baseline	Classic	Shift	Rate	Shift	Rate
Overlapped	69.6%	+2.4%	+7.2%	20.8%	+5.7%	30.3%
Slanted	80.0%	+8.8%	+6.4%	55.2%	+4.2%	56.8%
Jagged	94.2%	0.0%	+1.2%	57.0%	+0.8%	60.3%
Spirals†	81.4%	−2.0%	+11.3%	66.7%	+2.2%	65.9%

species following a maximum credit assignment strategy: each individual of this third species, a selection of hard samples in the fitness evaluation set, is evaluated against the best prototype selection and neighborhood proximity found in the previous generation, while the hardest evaluation subset of the previous generation is used to evaluate the fitness of individuals in the two other species. The best-of-run pair of the prototype set and neighborhood proximity is determined by evaluating each best-of-generation pair with the complete validation set and by keeping the best performing pair. Figure 5 presents this algorithm in greater detail for the co-evolution of three species.

The use of a competitive co-evolution for fitness cases selection has first been experimented by Hillis[22]. Much later, Panait and Luke[34] explored the idea in the context of evolving robust programs, with conclusive results for classical GP problems. Recent developments on competitive co-evolution propose the use of some form of Pareto dominance relation between the members of the different species[12]. This allows an "ideal" evaluation of the co-evolving individuals, at the cost of increased computations by comparing all combinations of individuals from different species.

Table 7 presents the results of the cooperative co-evolution of prototype selection and neighborhood proximity against a competitive evaluation data subset. In all cases, the competitive species evolved with each individual selecting 50% of the data in the evaluation set as fitness cases. The competitive fitness evolved with a population size of 1000, a uniform crossover probability of 0.3, and a bit flip mutation probability of 0.05. Parameters of the other two evolving species are the same as those used previously.

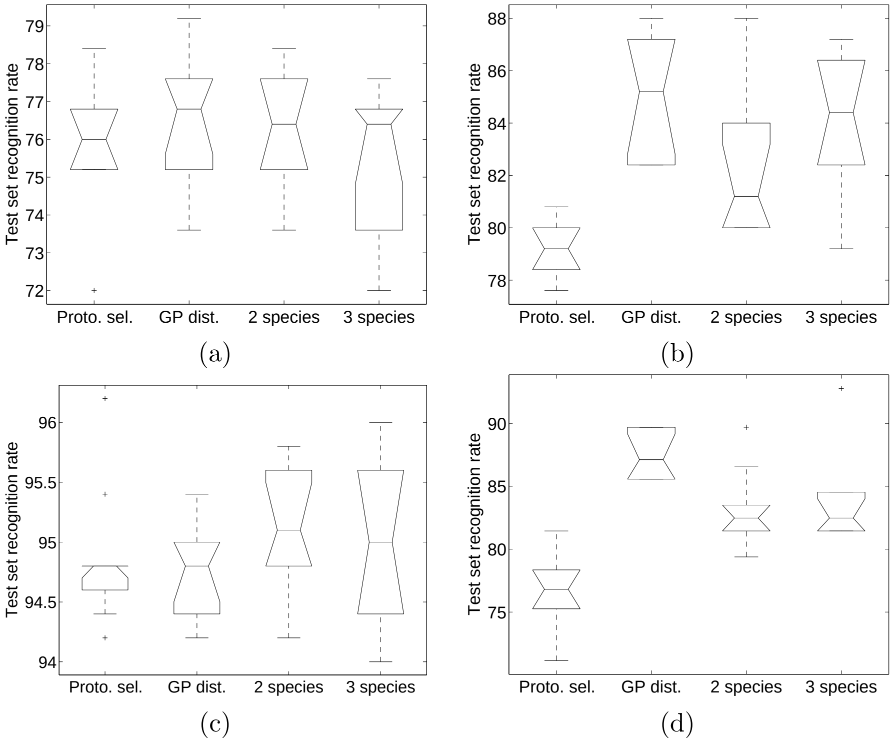
Results clearly show that using a third co-evolving species for selecting an evaluation set significantly reduces the numbers of prototypes selected, especially for the overlapped data set. Compared with the two species evolution, recognition rates are slightly reduced for overlapped and slanted data sets, but are increased for jagged and spirals. Overall, it can be concluded that the introduction of the

```
(1) Initialize generation counter, $g = 1$;
(2) Generate random initial populations for distance measures ($\mathcal{H}$), prototype
    selection ($\mathcal{P}$), and fitness case selection ($\mathcal{Q}$), respectively:
        $\mathcal{H}_1 = \{H_1^i\},\ i = 1, \ldots, m_H$
        $\mathcal{P}_1 = \{P_1^i\},\ i = 1, \ldots, m_P$
        $\mathcal{Q}_1 = \{Q_1^i\},\ i = 1, \ldots, m_Q$
(3) Select a random individual:
        $H_0^* = H_1^j,\ j = \mathrm{rand}(1, m_H)$
        $P_0^* = P_1^j,\ j = \mathrm{rand}(1, m_P)$
        $Q_0^* = Q_1^j,\ j = \mathrm{rand}(1, m_Q)$
(4) Evaluate each species of this generation using the best individual of the
    other two species for the previous generation:
        $f_H(H_g^i) = \mathrm{F}_{\mathrm{rec}}(H_g^i, P_{g-1}^*, Q_{g-1}^*),\ i = 1, \ldots, m_H$
        $f_P(P_g^i) = \mathrm{F}_{\mathrm{rec}}(H_{g-1}^*, P_g^i, Q_{g-1}^*),\ i = 1, \ldots, m_P$
        $f_Q(Q_g^i) = (1 - \mathrm{F}_{\mathrm{rec}}(H_{g-1}^*, P_{g-1}^*, Q_g^i)),\ i = 1, \ldots, m_Q$
(5) If $g = g_{\max}$, then stop;
(6) Find best individual for all species:
        $H_g^* = H_g^j,\ \ j = \underset{i=1 \ldots m_H}{\mathrm{argmax}}\,(f_H(H_g^i))$
        $P_g^* = P_g^j,\ \ j = \underset{i=1 \ldots m_P}{\mathrm{argmax}}\,(f_P(P_g^i))$
        $Q_g^* = Q_g^j,\ \ j = \underset{i=1 \ldots m_Q}{\mathrm{argmax}}\,(f_Q(Q_g^i))$
(7) Apply operations of natural selection, crossover, and mutation to all species;
(8) $g = g + 1$;
(9) Loop to step 4.
```

Fig. 5. Algorithm of the three species co-evolution. Function $\mathrm{F}_{\mathrm{rec}}(H, P, Q)$ evaluates the recognition rate of distance measure H over fitness case set Q using prototype set P.

co-evolving third species enhances the generalization capacity of the recognition system. Moreover, the third species reduces the number of evaluation test cases toward a significantly smaller number of prototypes, which contributes to reducing computational requirements, from 50% to 90% depending on the data set and specific evolution.

Figure 6 presents the box plots that stem from a one-way analysis of variance (ANOVA) of the test set recognition rates obtained with the synthetic data sets. It can be seen from these plots that the results for prototypes selection with GA



(a) (b) (c) (d)

Fig. 6. One-way analysis of variance (ANOVA) box plots of the test set recognition rates obtained with the synthetic data sets: (a) overlapped set, (b) slanted set, (c) jagged set, and (d) spirals set. The center box is bounded by the first and third quartiles of the data distribution, with the median as the central line in the box. The notches surrounding the median show the 95% confidence interval of this median. The whiskers above and below the boxes represent the spread of the data value within 1.5 times the interquartile range, with the + symbol showing outliers.

are worse than the three other approaches. Genetic programming of neighborhood distance, with and without co-evolution of prototypes selection, are the best performing approaches, followed by the three species co-evolution approach. For the three approaches involving GP, the baseline results are within the lowest quartile, which strengthen the statistical significance of the results.

7. Results on Real Data

In this section, the previous co-evolutionary approaches are re-investigated with five data sets taken from the UCI Machine Learning Repository[4]. These data sets are summarized in Table 8. Again, each data set is randomly partitioned into four equal size sub-sets (i.e. train, fitness evaluation, validation and test), just as for the synthetic data. Table 9 presents the baseline and best results obtained from classical 1-NN classifiers on these data sets, using the full training sets as prototypes, selecting the best 1-NN configuration from the fitness evaluation and validation sets

Table 8. Description of machine learning repository's data sets (ML data sets) chosen for performance evaluation of 1-NN classifier co-evolution.

Data Set	Size	Attr.	Classes	Application Domain
abalone	4177	8	29	Abalone age prediction from physical measurements.
cmc	1473	9	3	Contraceptive method choice prediction from demographic and socio-economic characteristics.
ionosphere	351	34	2	Structure detection in ionosphere from radar return signals.
pid	768	8	2	Diabetes diagnostic on female Pima Indians from general medical measurements.
spambase	4601	57	2	Non-junk/junk e-mail classification from specific word or character counts.

Table 9. ML data baseline and best classic configuration 1-NN classification rates on test sets.

	Baseline	Best Classic Configuration		
Data Set	(L_2)	Distance	Normalization	Rec. Shift
abalone	49.2%	L_1	Scaling	−0.7%
cmc	44.2%	L_∞	Scaling	+0.3%
ionosphere	84.1%	L_∞	No	+2.3%
pid	66.7%	L_2	Scaling	−1.0%
spambase	77.8%	L_1	Scaling	+10.4%

and giving results for the test sets. Likewise, Table 10 gives results on the same data sets with Wilson's editing and Hart's condensing.

First, simultaneous co-evolution of prototype selection with multi-objective GA and neighborhood proximity with GP has been conducted on the ML data sets as shown in Table 11. Results show significant recognition rate improvements for cmc and ionosphere data sets compared to the baseline as well as the best classic configurations of Table 9. For spambase, +13.5% is gained against the baseline, even though it is only about 3% better than the best classic configuration. The pid data set appears harder with a recognition gain of only +1%. The best classic configuration confirms this with a −1% over the baseline[a]. For the abalone data

[a] Readers should keep in mind that the best 1-NN configuration is selected from the validation set, with results provided on the test set.

Table 10. Recognition and selection rates obtained on ML data sets using Wilson's editing, Hart's condensing and Wilson's editing followed by Hart's condensing; $k = 1$ with L_2 metric and no normalization.

		Wilson's editing		Hart's condensing		Wilson + Hart	
		Rec.	Sel.	Rec.	Sel.	Rec.	Sel.
Data Set	Baseline	Shift	Rate	Shift	Rate	Shift	Rate
abalone	49.2%	+3.6%	48.1%	−0.9%	68.7%	+3.0%	17.2%
cmc	44.2%	+1.6%	39.1%	−0.5%	73.6%	+3.0%	15.0%
ionosphere	84.1%	−4.5%	86.2%	+1.1%	29.9%	0.0%	11.5%
pid	66.7%	+4.2%	70.8%	−4.7%	46.4%	+1.6%	14.1%
spambase	77.8%	−0.6%	74.0%	−3.8%	44.5%	−2.6%	15.7%

Table 11. Two species co-evolution of prototypes selection with multi-objective GA and distance measure with GP for 1-NN classification on ML data sets. Results with the data sets marked by a † are not statistically different from those of the baseline according to a 95% confidence Student's *t*-test.

			Best of 10		Mean of 10	
		Best	Rec.	Sel.	Rec.	Sel.
Data Set	Baseline	Classic	Shift	Rate	Shift	Rate
abalone†	49.2%	−0.7%	+4.5%	55.9%	−3.7%	56.7%
cmc	44.2%	+0.3%	+10.0%	59.8%	+5.6%	60.2%
ionosphere	84.1%	+2.3%	+8.0%	60.9%	+6.0%	54.6%
pid†	66.7%	−1.0%	+1.0%	38.0%	+1.1%	57.1%
spambase	77.8%	+10.4%	+13.5%	59.9%	+13.2%	57.5%

set, there is an important variation in recognition rates, going from an average of −3.7% to a maximum of +4.5%. This may be explained by the high number of classes of this data set, where each class is the age of an abalone. It might have been better to design a fitness function that gives some credits to classifications that are close (±1 year) to the real class labels. Finally, the selection rates obtained for all data sets is between 55% and 60%, with an exception for the best pid result (38%).

Second, the ML data sets are also tested using a three species co-evolution. For this experiment, the size of the fitness evaluation data set was restricted to either 50% of the original set, or to 200 for larger data sets (abalone and spambase). Presented in Table 12, results for the abalone, ionosphere and spambase data sets show a small decrease of performance in recognition rates, from −1% to −3%, compared with the two species co-evolution. For the cmc data set, the drop in recognition rate is more significant at −6.7% for the best result and −4.5% on average. Conversely, there is an improvement of +4.7% for the best pid data set,

Table 12. Three species co-evolution of prototypes selection with multi-objective GA, evaluation set with GA, and distance measures with GP for 1-NN classification on ML data sets. Results with the data sets marked by the † symbol are not statistically different from those of the baseline according to a 95% confidence Student's t-test.

			Best of 10		Mean of 10	
		Best	Rec.	Sel.	Rec.	Sel.
Data Set	Baseline	Classic	Shift	Rate	Shift	Rate
abalone†	49.2%	−0.7%	+2.6%	37.9%	−2.6%	36.4%
cmc†	44.2%	+0.3%	+3.3%	27.7%	+1.2%	27.3%
ionosphere	84.1%	+2.3%	+5.7%	50.6%	+4.4%	49.7%
pid†	66.7%	−1.0%	+5.7%	20.8%	+0.4%	21.8%
spambase	77.8%	+10.4%	+12.7%	45.2%	+12.1%	42.0%

even though the average performance results in a small drop of −0.7%. These recognition rate fluctuations are explained mainly by improvements to the selection rates compared to the two species co-evolution. The drop in the selection rate, which goes from 50% for ionosphere to 20% for pid, stem from solutions that use less information for classification and thus overfit less. This tends to confirm the robustness effects of competitive co-evolution that were also observed elsewhere[22,34].

Figure 7 presents the box plots for the ANOVA of the ML data sets. As expected, the plots for ionospere and spambase data sets show that the results are statistically better than baseline and best classical configuration. For the cmc and pid data sets, the baseline and best classical configuration results are within the lowest quartile for the two species co-evolution. And finally, there is no apparent statistical difference between the results for all the other comparisons between the classical configurations and the evolved approaches.

8. Discussion

For this work, the NN rule was used as the classification framework to be tuned by EC. But from a more general point of view, the paper's aim is to illustrate the idea that EC can be useful for fine tuning pattern recognition systems. This is not to say that EC can be used directly as the basis of any efficient recognition system: well established classification methods based on strong theoretical models perform very well in most situations. As the no free lunch theorems[45] indicate, a pure EC approach will probably not be able to surpass a domain-specific approach. However, EC are very generic and can be useful to hybridize classical classification approaches in order to circumvent their inherent limitations. Designing a good pattern recognition system usually goes beyond the task of training and testing a classifier. The working hypothesis of the classification methods are sometimes very strong and hard to obtain in practice. Domain knowledge is also often needed to extract the most discriminant features from raw data. EC can be very useful

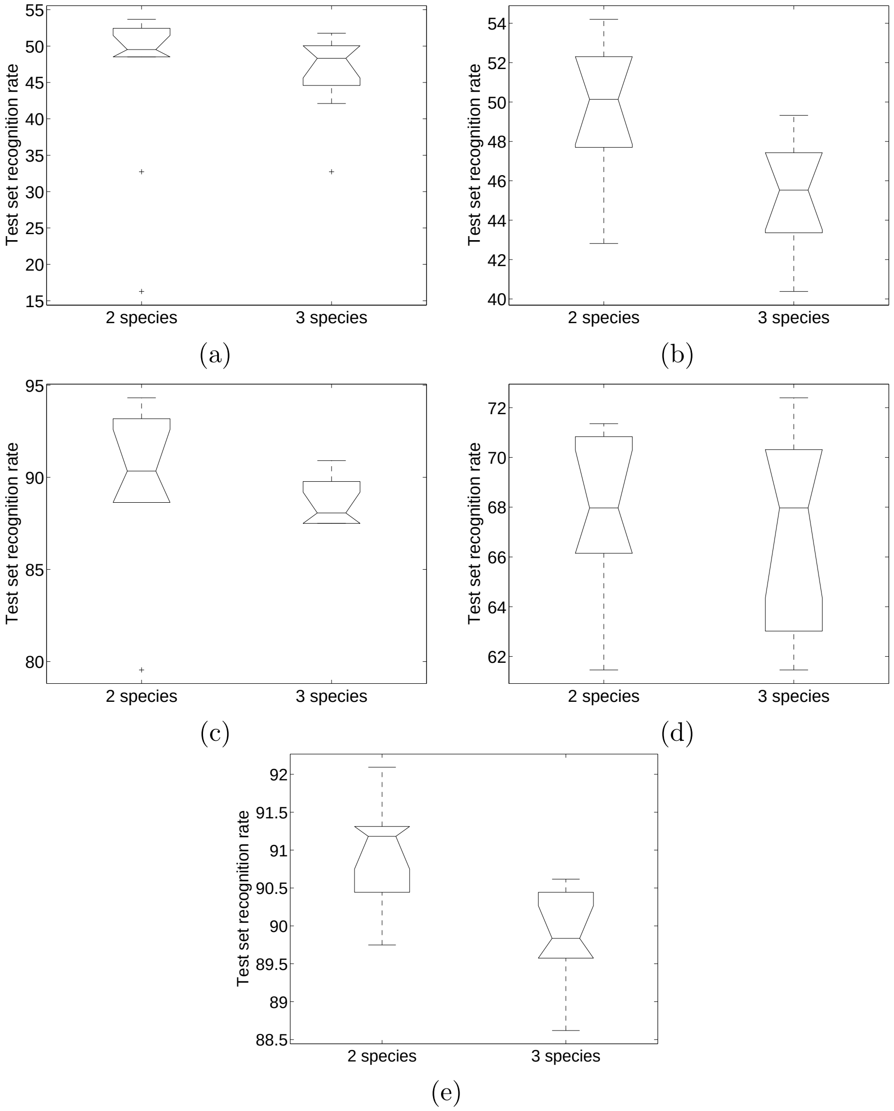


Fig. 7. One-way analysis of variance (ANOVA) box plots of the test set recognition rates obtained with the ML data sets: (a) abalone set, (b) cmc set, (c) ionosphere set, (d) pid set, and (e) spambase set.

in these cases to transform the original problem into a form that will allow the best performance of the classifier. The hybridization of EC with classifiers can be achieved is several ways to finely tune the system: bit string GA for element selection, real-valued GA for function optimization, GP for evolving the system's model, and so on. Moreover, co-evolution can be used to simultaneously optimize these different sub-systems.

Computationally speaking, as for most problems tackled with EC, the bottleneck lies in the fitness evaluations. This is especially true for the actual case given that NN classification complexity grows quadratically with the data set size. For example, 250000 measures of distance are computed for each fitness evaluation on the jagged set when using the complete training subset (500 instances) and the complete evaluation subset (500 instances). On a standard PC (AMD Athlon 1.2 GHz), the run time required for a single evolution ranges from an hour up to several days, depending on the data set used. To preserve tractability for large data sets, sampling of the data set must be conducted for fitness evaluation as it was done here with the co-evolution of prototype and fitness case selection. But the method presented here has been designed for improving generalization rather than reducing computational complexity. For large data sets, more aggressive approaches can be devised to preserved tractability, for an example of this see the paper of Song et al.[41].

9. Conclusion

This paper has introduced different ways of optimizing a Nearest Neighbor (NN) classifier using Evolutionary Computations (EC). First, a bit-string Genetic Algorithm (GA) was used to select prototypes in a training set. The optimization process is guided by two objectives: maximize recognition rate and minimize the size of the prototype set. A Pareto-based selection operation is proposed to simultaneously optimize all objectives and postpone to the end of the optimization process the trade-off between these two objectives. Results show that the multi-objective GA was able to select subsets of the training sets that achieve recognition rates equal or better to those obtained with the whole training data. Classical approaches for prototype selection achieve better selection rates, but at the expense of the recognition rates. Regarding the Pareto fronts, as shown in Figure 4, solutions with different recognition rate / prototypes set size ratio can be selected according to the needs of the practitioners.

Second, a Genetic Programming (GP) based approach was proposed for evolving data-specific neighborhood proximity measures, thus capturing the underlying model of the data. This is an important paradigm shift where the model itself is evolved, rather than the model's parameters, as in most previous approaches. Results are either better or comparable to those obtained from the best NN classifier when using a classic feature space normalization method.

Finally, cooperative and competitive co-evolution was also used to integrate the two previous approaches. A two species cooperative strategy delivered recognition rates comparable to those of the GP approach alone, while reducing the number of prototypes to levels comparable to the GA approach. This illustrates the interesting property of co-evolution, that allows different elements of a problem to evolve simultaneously with results as good as those obtained from isolated evolutions. A third competitive species was introduced, to induce an *arms race* between an evaluation data selection species and the two cooperative prototype selection and proximity measure species. In this way, the risk of overfitting the fitness evaluation data is much reduced. Moreover, even though recognition results were sometimes a little lower with this strategy, the evolved prototype sets were also much smaller.

Acknowledgments

This work was supported by NSERC-Canada and FQRNT-Québec. The authors are also grateful to Annette Schwerdtfeger for proofreading this manuscript and Jiachuan Wang for her early participation to the project.

References

1. T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators.* Institute of Physics Publishing, Bristol, UK, 2000.
2. W. Banzhaf, P. Nordin, R.E. Keller, and F.D. Francone. *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications.* Morgan Kaufmann, dpunkt.verlag, 1998.
3. Julie Beaulieu, Christian Gagné, and Marc Parizeau. Lens system design and reengineering with evolutionary algorithms. In *Proc. of the Genetic and Evolutionary Computations Conf. (GECCO 2002)*, pages 155–162, New York (NY), USA, 2002.
4. C.L. Blake and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html, 1998.
5. Stefan Bleuler, Martin Brack, Lothar Thiele, and Eckart Zitzler. Multiobjective genetic programming: Reducing bloat using SPEA2. In *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2001)*, pages 536–543. IEEE Press, 2001.
6. Martijn C. J. Bot. Feature extraction for the k-nearest neighbour classifier with genetic programming. In *Proc. of the European Conf. on Genetic Programming (EuroGP 2001)*, volume 2038 of *LNCS*, pages 256–267. Springer-Verlag, 2001.
7. Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172, 2002.
8. J. R. Cano, F. Herrera, and M. Lozano. Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study. *IEEE Trans. on Evolutionary Computation*, 7(6):561–575, December 2003.
9. Jian-Hung Chen, Hung-Ming Chen, and Shinn-Ying Ho. Design of nearest neighbor classifiers: Multi-objective approach. *Intl. Journal of Approximate Reasoning*, 40:3–22, 2005.
10. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publishers, 2002.
11. T.M. Cover. Estimation by the nearest neighbor rule. *IEEE Trans. on Information Theory*, 14(1):50–55, 1968.
12. Edwin D. de Jong and Jordan B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2):159–192, 2004.
13. Edwin D. de Jong, Richard A. Watson, and Jordan B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2001)*, pages 11–18, San Francisco (CA), USA, 2001. Morgan Kaufmann.
14. Gülsen Demiröz and H. Altay Güvenir. Genetic algorithms to learn feature weights for the nearest neighbor algorithm. In *Proc. of the Belgian-Dutch Conf. on Machine Learning (BENELEARN 1996)*, pages 117–126, 1996.

15. Pedro Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.
16. Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, second edition, 2001.
17. Mark Erickson, Alex Mayer, and Jeffrey Horn. The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems. In *Proc. of the Intl. Conf. on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 681–695, 2001.
18. Fernando Fernández and Pedro Isasi. Evolutionary design of nearest prototype classifiers. *Journal of Heuristics*, 10(4):431–454, 2004.
19. Christian Gagné and Marc Parizeau. Genericity in evolutionary computation software tools: Principles and case-study. *Intl. Journal on Artificial Intelligence Tools*, 15(2):173–194, April 2006.
20. Christian Gagné and Marc Parizeau. Open BEAGLE W3 page. http://beagle.gel.ulaval.ca, 2006.
21. Peter E. Hart. The condensed nearest neighbor rule. *IEEE Trans. on Information Theory*, 14:515–516, 1968.
22. W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.
23. Shinn-Ying Ho, Chia-Cheng Liu, and Soundy Liu. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters*, 23(13):1495–1503, November 2002.
24. J. M. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
25. Hitoshi Iba, Hugo de Garis, and Taisuke Sato. Genetic programming using a minimum description length principle. In Kenneth E. Kinnear, editor, *Advances in Genetic Programming*, Complex Adaptive Systems, pages 265–284. MIT Press, 1994.
26. David D. Jensen and Paul R. Cohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.
27. John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
28. John R. Koza, David Andre, Forrest H Bennett III, and Martin Keane. *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufman, 1999.
29. Krzysztof Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.
30. Ludmila I. Kuncheva and Lakhmi C. Jain. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters*, 20(11-13):1149–1156, November 1999.
31. William B. Langdon. Data structures and genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 20, pages 395–414. MIT Press, 1996.
32. Melanie Mitchell. *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. MIT Press, 1996.
33. L.S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In *Proc. of the Intl. Conf. on Pattern Recognition (ICPR 2002)*, pages 568–571, Québec (QC), Canada, 2002.
34. Liviu Panait and Sean Luke. Methods for evolving robust programs. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2003)*, volume 2724 of *LNCS*, pages 1740–1751, Chicago (IL), USA, 2003. Springer-Verlag.

35. Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
36. M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Trans. on Evolutionary Computation*, 4(2):164, July 2000.
37. J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
38. Jamie Sherrah, Robert E. Bogner, and Abdesselam Bouzerdoum. Automatic selection of features for classification using genetic programming. In *Proc. of the Australian and New Zealand Intelligent Information Systems Conf. (ANZIIS 1996)*, pages 284–287, 1996.
39. Jamie Sherrah, Robert E. Bogner, and Abdesselam Bouzerdoum. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. In *Genetic Programming 1997: Proc. of the Second Annual Conf.*, pages 304–312, Stanford University (CA), USA, 1997. Morgan Kaufmann.
40. Matthew G. Smith and Larry Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, September 2005.
41. Dong Song, Malcolm I. Heywood, and A. Nur Zincir-Heywood. Training genetic programming on half a million patterns: an example from anomaly detection. *IEEE Trans. on Evolutionary Computation*, 9(3):225–239, June 2005.
42. Terence Soule and James A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, Winter 1998.
43. R. Paul Wiegand, William C. Liles, and Kenneth A. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2001)*, pages 1235–1242, San Francisco (CA), USA, 2001. Morgan Kaufmann.
44. Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. on Systems, Man, and Cybernetics*, 2(3):408–421, July 1972.
45. David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Trans. on Evolutionary Computation*, 1(1):67–82, April 1997.
46. Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(2):44–49, 1998.

Christian Gagné received B.Ing. degree in computer engineering in 2000, and M.Sc. and Ph.D. degrees in electrical engineering in 2002 and 2005 respectively, all from Université Laval, Québec (Quebec), Canada. In 2005-2006, he was an ERCIM postdoc fellow jointly at the TAO team of the INRIA Futurs, France, located at the Université Paris-Sud in Orsay, and the Information Systems Institute of the University of Lausanne, Switzerland. Dr. Gagné is now working as a consultant for Informatique WGZ Inc., in Québec (Quebec), Canada. His research interests include evolutionary computation, machine learning, and distributed computing.

Marc Parizeau received B.Ing., M.Sc.A. and Ph.D. degrees all in electrical engineering from École Polytechnique, Montréal (Quebec), Canada in 1984, 1987, and 1992, respectively. Dr. Parizeau is a full professor of computer engineering at Université Laval, Québec (Quebec), Canada. He is a member of the Computer Vision and Systems Laboratory. His research interests are in pattern recognition, in learning, in evolutionary computation, and in computer vision. He is a member of the IEEE Computer Society, and the Canadian Information Processing and Pattern Recognition Society.