Feature Subset Selection Using Genetic Algorithms for Handwritten Digit Recognition

L. S. OLIVEIRA[1-3], N. BENAHMED[2], R. SABOURIN[1-3], F. BORTOLOZZI[1], C. Y. SUEN[3]

[1]PUCPR Pontifícia Universidade Católica do Paraná
PPGIA Programa de Pós-Graduação em Informática Aplicada
LARDOC Laboratório de Análise e Reconhecimento de Documentos
Rua Imaculada Conceição 1155, 80215-901 - Curitiba, PR - BRAZIL
{soares,fborto}@ppgia.pucpr.br

[2]ETS Ecole de Technologie Supérieure
LIVIA Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle
1100, rue Notre Dame Ouest, Montreal, H3C 1K3, CANADA
benahmed@livia.etsmtl.ca, sabourin@gpa.etsmtl.ca

[3]CENPARMI Centre for Pattern Recognition and Machine Intelligence
1455 de Maisonneuve Blvd. West, Suite GM 606 - Montreal, H3G 1M8, CANADA
suen@cenparmi.concordia.ca

Abstract. In this paper two approaches of genetic algorithm for feature subset selection are compared. The first approach considers a simple genetic algorithm (SGA) while the second one takes into account an iterative genetic algorithm (IGA) which is claimed to converge faster than SGA. Initially, we present an overview of the system to be optimized and the methodology applied in the experiments as well. Afterwards we discuss the advantages and drawbacks of each approach based on the experiments carried out on NIST SD19. Finally, we conclude that the IGA converges faster than the SGA, however, the SGA seems more suitable for our problem.

1 Introduction

In practical pattern recognition problems, a classification function learned through an inductive learning algorithm assigns a given input pattern to one of the n existing classes of the system. Usually, the representation of each input pattern consists of features since they can distinguish one class of patterns from another in a more concise and meaningful way than offered by the raw representation. In many applications, it is not unusual to find problems involving hundreds features. However, it has been observed that, beyond a certain point, the inclusion of additional features leads to a worse rather than better performance. Moreover, the choice of features to represent the patterns affects several aspects of the pattern recognition problem such as accuracy, required learning time and necessary number of samples.

This apparent paradox presents us with a feature subset selection problem in automated design of pattern classifiers. Such a problem refers to the task of identifying and selecting a useful subset of features to be used to represent patterns from a larger set of often mutually redundant or even irrelevant features. Therefore, the main goal of feature subset selection is to reduce the number of features used in classification while maintaining an acceptable classification accuracy.

Feature subset selection algorithms can be classified into two categories based on whether or not feature selection is performed independently of the learning algorithm used to construct the verifier. If feature selection is done independently of the learning algorithm, the technique is said to follow a filter approach. Otherwise, it is said to follow a wrapper approach [7]. The first one is computationally more efficient but its major drawback is that an optimal selection of features may not be independent of the inductive and representational biases of the learning algorithm that is used to build the classifier. On the other hand, the wrapper approach involves the computational overhead of evaluating a candidate feature subset by executing a selected learning algorithm on the database using each feature subset under consideration.

Feature subset selection in the context of practical applications such as handwritten recognition presents a multi-criterion optimization function, e.g. number of features and accuracy of classification. Genetic algorithms offer a particularly attractive approach for this kind of problems since they are generally quite effective for rapid global search of large, non-linear and poorly understood spaces. Moreover, genetic algorithms are very effective in solving large-scale problems [16, 22].

This paper focuses on the feature subset selection for handwritten digit recognition through a modified wrapper-

1530-1834/01 $10.00 © 2001 IEEE

based multi-criterion approach using genetic algorithms in conjunction with a multi-layer perceptron neural network. Two different versions of the genetic algorithm were explored: simple genetic algorithm (SGA) and iterative genetic algorithm (IGA). All experiments reported in this paper use NIST SD19.

This paper is structured as follows. Section 2 presents a brief introduction of genetic algorithms. Section 3 describes the methodology applied in this work. Sections 5 and 6 present both approaches of genetic algorithms discussed in this paper. Section 7 reports the experiments carried out and section 8 includes some discussion and comparison. Finally, section 9 presents our conclusions.

2 Genetic Algorithms

In this section we present a brief introduction about genetic algorithms. A more detailed introduction can be found in [17].

The genetic algorithm is a model of machine learning which derives its behaviour from a metaphor of some of the mechanisms of evolution in nature. This is done by the creation within a machine of a population of individuals represented by chromosomes, in essence a set of character strings that are analogous to the base-4 chromosomes that we see in our own DNA.

The individuals represent candidate solutions to the optimization problem being solved. In genetic algorithms, the individuals are typically represented by n-bit binary vectors. The resulting search space corresponds to an n-dimensional boolean space. It is assumed that the quality of each candidate solution can be evaluated using a fitness function.

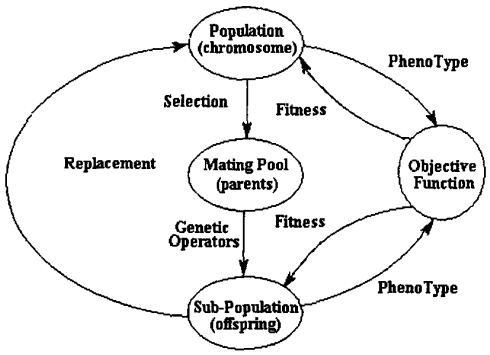

Figure 1: A SGA cycle.

Genetic algorithms use some form of fitness-dependent probabilistic selection of individuals from the current population to produce individuals for the next generation. The selected individuals are submitted to the action of genetic operators to obtain new individuals that constitute the next generation. Mutation and crossover are two of the most commonly used operators that are used with genetic algorithms that represent individuals as binary strings. Mutation operates on a single string and generally changes a bit at random while crossover operates on two parent strings to produce two offsprings. Other genetic representations require the use of appropriate genetic operators.

The process of fitness-dependent selection and application of genetic operators to generate successive generations of individuals is repeated many times until a satisfactory solution is found. In practice, the performance of genetic algorithm depends on a number of factors including: the choice of genetic representation and operators, the fitness function, the details of the fitness-dependent selection procedure, and the various user-determined parameters such as population size, probability of application of different genetic operators, etc. The specific choices made in the experiments reported in this paper are summarized in section 3.2. Figure 1 depicts a SGA cycle. The basic operation of the genetic algorithm is outlined as follows:

```
Procedure:
begin
  t <- 0
  initialize P(t)
  while (not termination condition)
    t <- t + 1
    select P(t) from p(t - 1)
    crossover P(t)
    mutate P(t)
    evaluate P(t)
  end
end
```

Since genetic algorithms were designed to efficiently search large spaces, they have been used for a number of different application areas such as camera calibration [19], signature verification [21], medical diagnosis [11], facial modeling [20] and handwritten recognition [8].

3 Methodology

3.1 Representation and Operators

In this subsection we present the choice of a representation for encoding candidate solutions to be manipulated by the genetic algorithm.

Each individual in the population represents a candidate solution to the feature subset selection problem. Let m be the total number of features available to choose from to represent the patterns to be classifier ($m = 132$ in our case). The individual (chromosome) is represented by a binary vector of dimension m. If a bit is a 1, it means that the corresponding feature is selected, otherwise the feature is not selected. This is the simplest and most straightforward

representation scheme [12]. As mentioned before, other genetic representations require the use of appropriate genetic operators.

Since we are representing a chromosome through a binary string, the operators mutation and crossover operates in the following way: Mutation operates on a single string and generally changes a bit at random. Thus, a string 11010 may, as a consequence of random mutation get changed to 11110. Crossover on two parent strings to produce two offsprings. With a randomly chosen crossover position 4, the two strings 01101 and 11000 yield the offspring 01100 and 11001 as a result of crossover.

3.2 Parameter Settings

Our experiments used the following parameter settings:

- Population size: 30
- Number of generation: 1000
- Probability of crossover: 0.8
- Probability of mutation: 0.007

The parameter settings were based on results of several preliminary runs. They are comparable to the typical values reported in the literature [1].

3.3 Selection Mechanism

The selection mechanism is responsible for selecting the parent chromosome from the population and forming the mating pool. The selection mechanism emulates the survival-of-the-fittest mechanism in nature. It is expected that a fitter chromosome receives a higher number of offsprings and thus has a higher chance of surviving on the subsequent evolution while the weaker chromosomes will eventually die.

In this work we are using the roulette wheel selection [4] which is one of the most common and easy-to-implement selection mechanism. Basically it works as follows: each chromosome in the population is associated with a sector in a virtual wheel. According to the fitness value of the chromosome, the sector will have a larger area when the corresponding chromosome has a better fitness value while a lower fitness value will lead to a smaller sector.

3.4 Objective Function and Fitness Evaluation

The fitness evaluation is a mechanism used to determine the confidence level of the optimized solutions to the problem. Usually, there is a fitness value associated with each chromosome, e.g., in a minimization problem, a lower fitness value means that the chromosome or solution is more optimized to the problem while a higher value of fitness indicates a less optimized chromosome.

Our problem consists of optimizing two objectives: minimization of the number of features and minimization of the error rate of the classifier. Therefore, we are dealing with a multi-objective optimization problem. While in single-objective optimization the optimal solution is usually clearly defined, this does not hold for multi-objective optimization problem. Instead of a single optimum, there is rather a set of alternative trade-offs, generally known as Pareto-optimal solutions.

In order to generate the Pareto-optimal set, we are using a classical approach proposed by Hajela and Lin in [18], called weighting method, which aggregates the objectives into a single and parameterized objective. Such an aggregation is performed through a linear combination of the objectives

$$f(x) = f_1(x) \times \omega_1 + f_2(x) \times \omega_2 \qquad (1)$$

where ω_i are called weights and, without loss of generality, normalized such that $\sum w_i = 1$. $f_1(x)$ is the error rate produced by the classifier for a given feature subset (represented by the chromosome x) and $f_2(x)$ is the number of features selected in the chromosome x. Therefore, the fitness of a chromosome is represented by a single and parameterized objective function $f(x)$.

Using genetic algorithms for feature subset selection involves the running of a genetic algorithm for several generations. Regarding a wrapper approach, in each generation, evaluation of a chromosome (a feature subset) requires training the corresponding neural network and computing its accuracy. This evaluation has to be performed for each of the chromosomes in the population. Since such a strategy is not feasible due to the limits imposed by the learning time of the huge training set considered in this work, we have adopted the strategy proposed by Moody and Utans in [10], which uses the sensitivity of the network to estimate the relationship of input features with network performance.

The sensitivity of the network model to variable β is defined as:

$$S_\beta = \frac{1}{N} \sum_{j=1}^{N} ASE(\bar{x}_\beta) - ASE(x_\beta) \qquad (2)$$

with

$$\bar{x}_\beta = \frac{1}{N} \sum_{j=1}^{N} x_{\beta_j} \qquad (3)$$

where x_{β_j} is the β^{th} input variable of the j^{th} exemplar. S_β measures the effect on the training ASE (average square

error) of replacing the β^{th} input x_β by its average $\bar{x}_\beta$. Replacement of a variable by its average value removes its influence on the network output.

So, in order to evaluate a given feature subset we replace the unselected features by their averages. In this way, we avoid training the neural network and hence turn the wrapper approach feasible for our problem. We call this strategy modified-wrapper. Such a kind of scheme has been employed by Emmanouilidis et al in [3] and Yuan et al in [9].

4 Feature Set and Classifier

In this section we describe both the feature set and classifier used in our experiments. The feature vector is based on a mixture of concavity and contour-based features while the classifier is a neural network trained with the backpropagation algorithm [5]. Such a recognition module has been successfully applied on handwritten digit recognition to our recent works [14, 15].

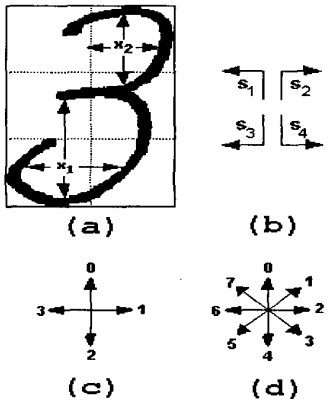

Figure 2: Feature set: (a) Concavities, (b) Auxiliary directions, (c) 4-Freeman directions and (d) 8-Freeman directions.

The basic idea of concavity measurements [13] is the following: for each white pixel in the component, we verify in each possible direction (Figure 2a), if a black pixel can be reached. The number of times as well as the directions leading to the black pixels are computed and stored in a vector. When black pixels are reached in four directions (e.g. point x_1 in Figure 2a), we branch out in four auxiliary directions (s_1 to s_4 in Figure 2b) in order to confirm if the current white pixel is really inside a closed contour. Those pixels that reach just one black pixel are discarded. Therefore, the concavity measurements are represented by 13 components.

The second part of the vector contains contour information, which is extracted from a histogram of contour directions. Taking into account 8-Freeman directions (Figure 2d), we have 8 more components in our feature vector. The last component of this vector corresponds to the character surface. Finally, the image is divided into six regions and 132 components normalized between 0 and 1 are considered.

In order to train the neural network, we have used the NIST SD19 in the following way: the training, validation and testing sets were composed of 195,000, 60,089 and 58,646 samples from hsf_{0,1,2,3}, hsf_7 and hsf_4 respectively. The recognition rates (zero-rejection level) achieved by the classifier were 99.66%, 99.13% and 97.52% on the training, validation and testing sets respectively.

Despite the fact that this feature set achieves good recognition rates on NIST database, we believe that it can be optimized since it contains a large number of components. In the next section we will compare two different strategies to carry out this task.

5 Simple genetic algorithm

In this experiment an SGA was used, i.e., an algorithm based on bit representation, one-point crossover, bit-flip mutation, roulette wheel selection (with elitism). The sole modification that we have carried out was the initialization of the population. We have inserted a chromosome with all features selected. Since we know an admissible solution of the system, it is very interesting to use such a knowledge in order to speed up the convergence time of the genetic algorithm.

6 Iterative genetic algorithm

This approach is based on the work presented by Man et al [12]. The main idea behind this approach is to speed up the convergence time of the algorithm by restricting the search space in each iteration. The algorithm is described as follows:

1. Let N_2 be the maximum allowable topology for searching (132 features in our case). The algorithm is applied and terminated when a solution x_1 with ($f_1 \approx 0$) is obtained. (Figure 3a).

2. Assuming that $f_2(x_1) = N_3$, the searching domain for the complexity of the topology is reduced from N_2 to $N_3 - 1$. The algorithm is then applied again until another solution with ($f_1 \approx 0$) is obtained (Figure 3b).

3. Repeat step 2 by reducing the searching domain of the topology complexity and eventually the optimal point x_{opt} with $f_2(x_{opt}) = N_1$ will be obtained (Figure 3c).

4. Another iteration with the complexity of the topology bounded by $N_1 - 1$ is carried out and no solution may be found. This process can be terminated by setting a maximum number of generations for the algorithm. If no solution is found after the generation exceeds this maximum number, the solution obtained in step 3 would be considered as the optimal solution for the problem (Figure 3d).

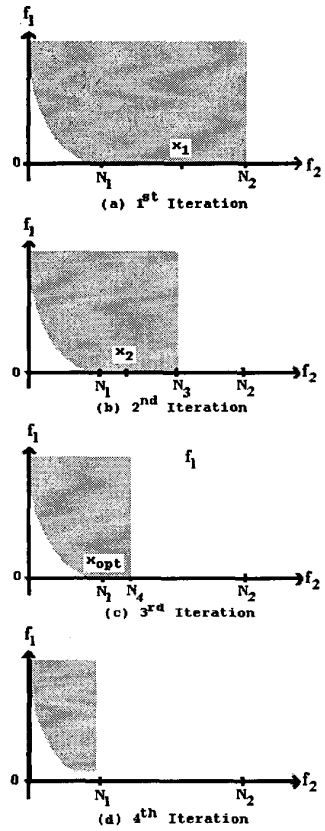

Figure 3: Iterative approach.

The same methodology and parameter settings used in the previous approach are also used here. The differences lie basically in two points: search mechanism and initialization of the population. The search mechanism is clearly illustrated in Figure 3, where the best solution found in the previous iteration is used to initialize the current one. The initialization of the population was modified in this approach in order to allow a more focused search in each iteration. In order to perform this, we have used a Hamming distance d between the injected solution and the chromosomes generated at the initialization time. Such a constraint is defined as

$$d(S_1, S_2) < \tau \quad (4)$$

where S_1 is the chromosome that represents the best solution found in the last iteration of the algorithm (if the algorithm is running the first iteration, S_1 will be represented by the chromosome with all features selected), S_2 is a chromosome generated at the initialization time and τ is the threshold that defines the maximal distance between S_1 and S_2. Figure 4 shows an example of the initialization using $\tau = 5$. In such a case, the initialization of the population produces a population entirely located in the sub-space A.

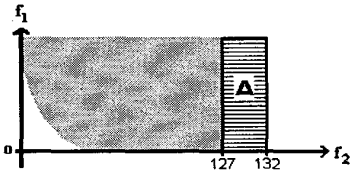

Figure 4: Initialization of the population using the Hamming distance.

7 Experiments

In this section we present some experiments that use the two different approaches outlined in the previous sections. The main goal of these experiments is to optimize the feature set presented in section 4 to reduce both the complexity and error rate of the classifier.

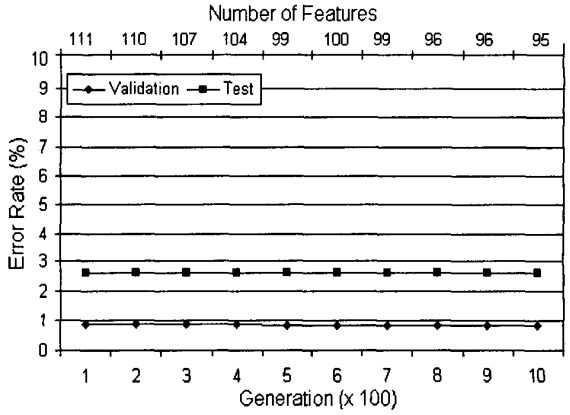

Figure 5: Performance of the SGA.

The first experiment takes into account the SGA to perform the features subset selection. After 1,000 generations, the best solution found by the SGA was 95 features and error rates of 0.83% and 2.60% on validation and testing sets

Table 1: Results found by both approach on different data sets.

Data Set	Original Feature Set		SGA Subset		IGA Subset	
	Features	Error %	Features	Error %	Features	Error %
Learning	132	0.34	95	0.34	104	0.34
Validation	132	0.87	95	0.83	104	0.84
Test	132	2.48	95	2.60	104	2.58

respectively. Figure 5 shows the trade-off the between error rates and the number of features selected.

The second experiment considers the iterative approach. We have run 10 iterations with 100 generations each. The best solution provided by this strategy was found at the seventh iteration and it has selected 104 features and produced error rates of 0.84% and 2.58% on validation and testing sets respectively. As we can observe in Figure 6, after the seventh iteration this approach faces an over-training problem, since it finds admissible error rates on the validation set, a small feature subset (about 65 features) but a very poor generalization on the testing set.

In order to validate the solutions provided by the genetic algorithms, we have re-trained the classifier with these solutions and observed the same performance achieved by the original classifier.

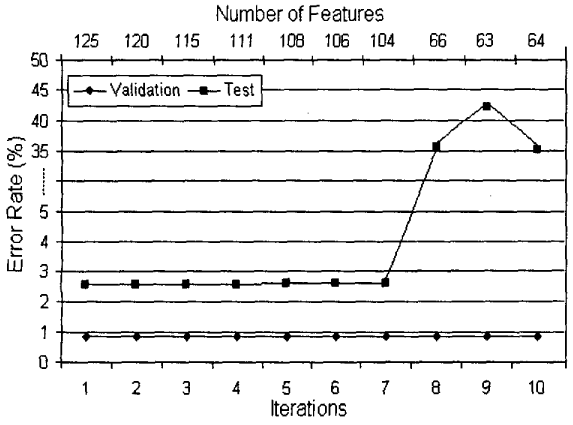

Figure 6: Performance of the IGA.

Figure 7 and Table 1 summarize the results found by both approaches on different data sets.

8 Discussion

So far, we have described two different approaches of feature subset selection using genetic algorithms. As we have seen, the main difference between these approaches lies in the search mechanism. In the previous section we have observed that both approaches achieved satisfactory results in reducing the number of features used by the classifier while maintaining the error rates in the same level produced by the original feature set.

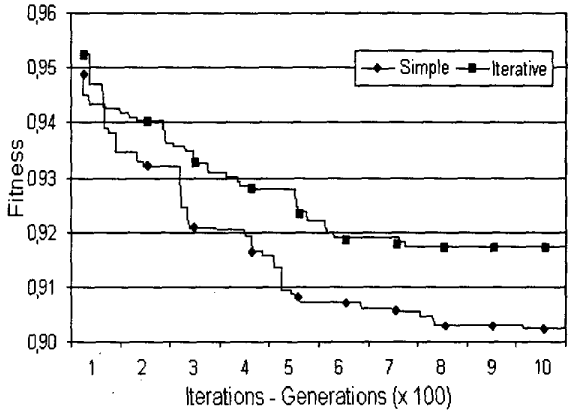

Figure 7: Evolution of the error rates.

We have seen in our experiments that the IGA converges faster than the SGA, since it found an optimal solution at the seventh iteration (700 generations). However, we can observe also that the SGA reaches a more interesting solution in terms of number of features (95 instead of 104).

Figures 8a and b clearly illustrate the evolution of the chromosomes in the objective plane for all generations for both approaches. As we can see in Figure 8a, the SGA focuses its search in a more defined sub-space. Such a concentration is due to the objective function that we have chosen and also the elitist selection method applied. On the other hand, the IGA searches in a broad space and consequently it finds a great variety of solutions. However, part of these solutions pay a high price (low accuracy) for having a reduced number of features. This behaviour can be explained by the methodology applied in creating a new population of chromosomes, which was described in section 6.

Figures 8c and d detail Figures 8a and b respectively.

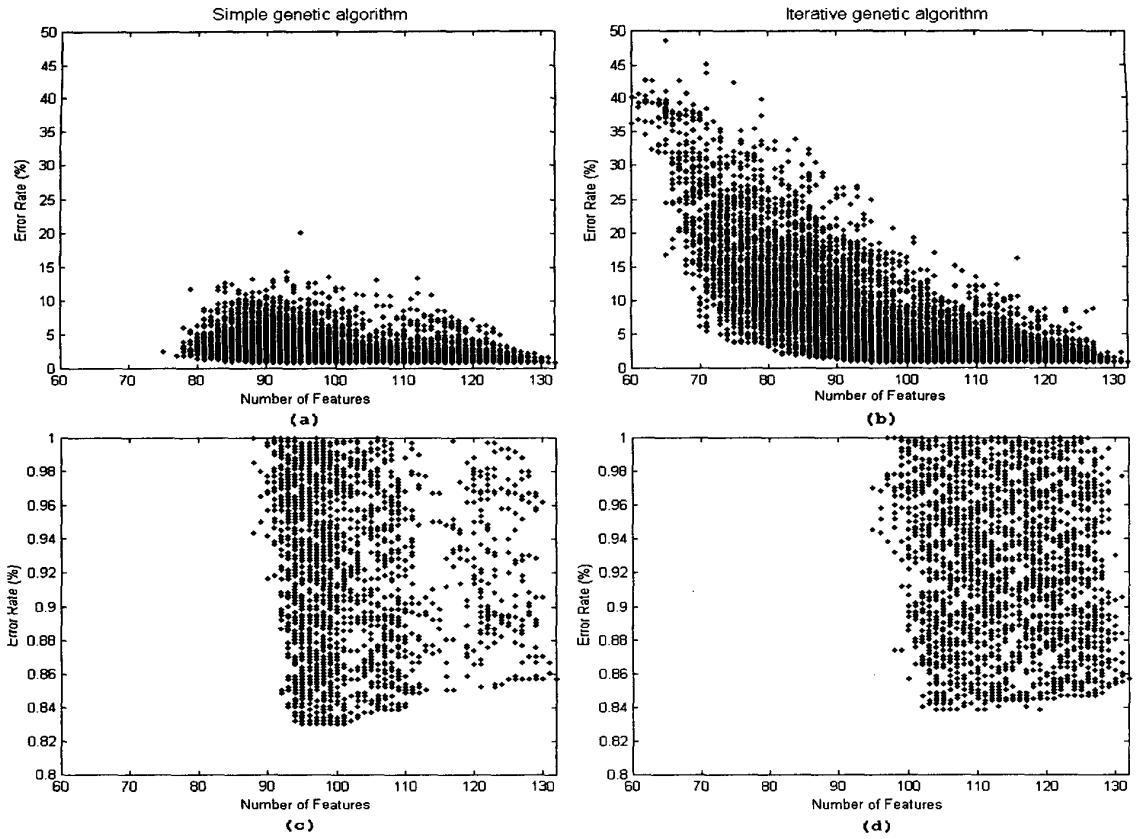

Figure 8: Evolution of the chromosomes in the objective plane for all generations: (a) Simple genetic algorithm, (b) Iterative genetic algorithm (c) Zoom of SGA and (d) Zoom of IGA.

In both Figures we can observe the distribution of the chromosomes for an error rate between 0,8 and 1%. As mentioned before, both approaches produce similar error rates, however, we can verify that the SGA yields a great number of solutions in the interval 90-100 features, which does not happen in the IGA. In this Figure we can see also a set of alternative trade-offs (error rate-number of features) generated during the search.

It is interesting to emphasize that all results reported so far are computed by using a classical multi-objective optimization function. We believe that our methodology can be enhanced by using different approaches of multi-objective optimization in order to focus the search in the Pareto-optimal subset of solutions [6].

Since we are dealing with a large scale classifier (more than 100 features and a huge database) our experiments were very time consuming. For instance, 1,000 generations of SGA in a SUN Ultra (167 Mhz CPU, with 128 Mb RAM) took about 20 days. However, low cost parallel computing based on a cluster of personal computers (PCs) makes this approach tractable on short term [2].

9 Conclusion

In this paper we have discussed two different strategies for feature subset selection using genetic algorithms. All experiments reported in this work use a wrapper-based multi-criterion approach in conjunction with a multi-layer perceptron neural network. We have shown that such a scheme became feasible by means of the sensitivity analysis.

We have seen that both approaches discussed in this paper achieved interesting results in reducing the complexity of the classifier. However, the SGA seems more suitable for our problem than the IGA by Man et al in [12]. This conclusion is based on comprehensive experiments carried out on NIST SD19 databases, where the SGA provided a reduction of about 28% of the feature vector and maintained the error rates at the same level of the original feature set.

In spite of the fact that both approaches did not suc-

ceed in reducing the error rate of the classifier, we consider they achieved their objective, since the classifier used in our experiments already have a good performance on NIST SD19 database.

For future works we plan to study different approaches of multi-objective optimization as well as to apply different operators and schemes of representation for our problem.

Acknowledgements

The authors wish to thank Pontifícia Universidade Católica do Paraná, Paraná Tecnologia and Natural Sciences and Engineering Research Council of Canada which have supported this work.

References

[1] A.E.Eiben, R.Hinterding, and Z.Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, 3(2):124–141, 1999.

[2] E. Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, 2000.

[3] C.Emmanouilidis, A.Hunter, and J.MacIntyre. A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In *Congress on Evolutionary Computation*, volume 1, pages 309–316, 2000.

[4] L. Davis. *Handbook on Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[5] D.E.Rumelhart, R.Durbin, R.Golden, and Y.Chauvin. Backpropagation: The basic theory. In Y.Chauvin and D.E.Rumelhart, editors, *Backpropagation:Theory, Architectures and Applications*, pages 1–34. Lawrence Erlbaum, Hillsdale, NJ, 1995.

[6] E.Zitzler, K.Deb, and L.Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[7] G.John, R.Kohavi, and K.Pfleger. Irrelevant features and the subset selection problems. In *11^{th} International Conference on Machine Learning*, pages 121–129, 1994.

[8] G.Kim and S.Kim. Feature selection using genetic algorithms for handwritten character recognition. In *7^{th} IWFHR*, pages 103–112, Amsterdam-Netherlands, 2000.

[9] H.Yaun, S.S.Tseng, W.Gangshan, and Z.Fuyan. A two-phase feature selection method using both filter and wrapper. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 132–136, 1999.

[10] J.Moody and J.Utans. Principled architecture selection for neural networks: Application to corporate bond rating prediction. In J.Moody, S.J.Hanson, and R.P.Lippmann, editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, 1991.

[11] J.Yang and V.Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(1):44–49, 1998.

[12] K.F.Man, K.S.Tang, and S.Kwong. *Genetic Algorithms: Concepts and Designs*. Springer-Verlag, London-UK, 1999.

[13] L.Heutte, J.Moreau, B.Plessis, J. Plagmaud, and Y.Lecourtier. Handwritten numeral recognition based on multiple feature extractors. In *2^{nd} ICDAR*, pages 167–170, 1993.

[14] L.S.Oliveira, E. Lethelier, F. Bortolozzi, and R.Sabourin. A new approach to segment handwritten digits. In *7^{th} IWFHR*, pages 577–582, Amsterdam-Netherlands, 2000.

[15] L.S.Oliveira, R.Sabourin, F.Bortolozzi, and C.Y.Suen. A modular system to recognize numerical amounts on brazilian bank cheques. To appear in 6^{th}ICDAR, Seattle-USA, September, 2001.

[16] M.Kudo and J.Sklansky. Comparision of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25–41, 2000.

[17] M.Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge - MA, 1996.

[18] P.Hajela and C.Y.Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

[19] Q.Ji and Y.Zhang. Camera calibration with genetic algorithms. *IEEE Trans. on Systems, Man, and Cybernetics, Part A:Systems and Humans*, 31(2):120–130, 2001.

[20] S.Y.Ho and H.L.Huang. Facial modeling from an uncalibrated face image using a coarse-to-fine genetic algorithm. *Pattern Recognition*, 34(5):1015–1031, 2001.

[21] V.E.Ramesh and N.Murty. Off-line signature verification using genetically optimized weighted features. *Pattern Recognition*, 32(2):217–233, 1999.

[22] W.Siedlecki and J.Sklansky. A note on genetic algorithms for large scale on feature selection. *Pattern Recognition Letters*, 10:335–347, 1989.