Evolutionary Approach to Feature Selection with Associative Models

Angel Ferreira Santiago[1], Cornelio Yáñez Márquez[1],
Mario Aldape Pérez[2], and Itzamá López Yáñez[2]

[1] Centro de Investigación en Computación, CIC Building, Mexico City, Mexico
B130078@sagitario.cic.ipn.mx, cyanez@cic.ipn.mx
[2] Centro de Innovación y Desarrollo Tecnológico en Cómputo, CIDETEC Building, Mexico City, Mexico
maldape@ieee.org, ilopezy@ipn.mx

Abstract. Feature selection aims to find ways to single out the subset of features which best represents the phenomenon at hand and improves performance. This paper presents an approach based on evolutionary computation and the associative paradigm for classification. A wrapper-style search guided by a genetic algorithm uses the Hybrid Associative Classifier to evaluate candidate solutions and thus approximate the optimal feature subset for different data sets. The results suggest that this is a feasible approach for feature selection, obtaining solutions equal or similar to the optimal solution while evaluating a relatively small fraction of the search space.

Keywords: Feature Selection, Neural Networks, Genetic Algorithms

1 Introduction

Information in excess often becomes redundant or presents irrelevant portions which rather than help describe the phenomenon at hand, hinder the ability to better understand it. In addition, the time most machine learning methods require to work on a data set is proportional to the number of features it contains. Therefore, reducing the number of features or selecting those which are relevant is crucial for improving processing speed and accuracy of classification. Thus, the field of Feature Selection aims to find ways to single out the subset of features which best represents the phenomenon and as such optimizes performance.

Feature selection algorithms can be divided into two main groups: *filters* and *wrappers*. Filter methods compute a subset of features based on analysis of the intrinsic properties of the features themselves[1]. These methods are classifier-agnostic since they solely rely on information about the features to make a decision about which ones are the most informative. Thus, they can be used

as a prior step to any classification method. In contrast, wrappers analyze performance of feature subsets based on a specific classifier[2]. Since this approach depends on a classification algorithm, the optimal subset obtained by one wrapper method may be different from the result of using the method with another classification technique. Because exhaustive search is computationally expensive, wrapper methods employ a heuristic search of the solution space to find a solution. Wrapper methods can be further divided into deterministic and stochastic searches [3].

Different approaches to feature selection have been proposed over the years. For example, using semi-supervised learning together with an ensemble classifier [4], selecting features by clustering [5], filtering features through the analysis of relevance and redundancy [6] and studying mutual information between variables to remove redundant ones [7].

Determining the starting point and the direction of the search is crucial to finding or approximating the optimal solution. Working with a population of candidate solutions (as opposed to a single one) allows the search to start in different points of the solution space. If the search can be guided to follow the most promising paths, there is a fair chance of arriving at the neighborhood of the optimal solution. A genetic algorithm (GA) thus can be used to perform a broad exploration of the solution space while exploiting the best solutions to steadily approach the optimal feature subset. Feature selection lends itself to a genetic search with a binary chromosome representation. Since each feature will either be selected or not, a feature subset can be expressed as a binary vector. For each component, a value of 1 shall represent a selected feature and a value of 0 shall mean the opposite. The use of a genetic algorithm to address the problem of feature selection was introduced by Siedlecki and Sklansky[8]. Furthermore, Yang and Honavar [9] adopt this approach to perform feature selection with a neural network.

In this paper, a genetic algorithm-based wrapper method for feature selection is proposed. This method employs an associative classifier as the method for evaluating feature subsets. The use of a wrapper-style feature selection method with an associative classifier was first proposed in [10]. This proposal executes an exhaustive search aided by parallel processing to reduce the time required. However, that approach is still too time-consuming, albeit guaranteed to find the optimal solution. As such, this work aims to find a balance between classification accuracy and the time needed to find a suitable feature subset.The preliminary results of the conducted experiments show that a GA coupled with an associative classifier exhibits rapid convergence to a near-optimal feature subset on different data sets. The rest of the paper is organized as follows. Section 2 provides a brief overview of genetic algorithms. Section 3 explains the associative classifier

employed in the proposed model. The proposed model is described in Section 4 and the experimental results are presented in Section 5. To conclude, Section 6 presents a discussion on the obtained results.

2 Genetic Algorithms

Genetic algorithm research was popularized with the work of Holland [11]. It is a branch of evolutionary computation which focuses on modeling the genetic evolution of organisms as it happens in nature, via recombination of genes and random mutations. A brief overview of genetic algorithms follows, although for a more complete review the reader is referred to Holland's work or the book by Engelbrecht [12].

Research in GAs has flourished recently; having been applied in several different areas. GAs have been used in anti-missile systems [13], vehicle design [14] and signal processing [15], among others.

The crux of a genetic algorithm is in the choice of operators which control how the candidate solutions will evolve and how the population will converge to a solution. These operators consist of the *crossover operator*, the *mutation operator* and the *selection operator*.

Crossover operators exist to produce new solutions via the exchange of genetic material between two existing "parent" solutions. That is, crossover enables the GA to exploit an area of the search space. In the case of feature selection, crossover hopes to pass on the genetic material representing the most useful features on to the next generation. Among the most used crossover operators are *one-point crossover*, *two-point crossover* and *uniform crossover* [12].

One-point crossover selects a point $1 < p < n$, where n is the number of values in the chromosome, at which to split the parent vectors. After that, an offspring is created by assembling the first segment of one parent and the second segment of the remaining parent. That is, given two parents A and B, a child solution C is given by:

$$C_i = \begin{cases} A_i, & i \leq p \\ B_i, & i > p \end{cases}$$

Similarly, two-point crossover works by selecting two crossover points, p_1 and p_2, splitting the parent chromosomes at those two points, and then swapping their middle segments to create an offspring. That is:

$$C_i = \begin{cases} A_i, & i \leq p_1 \\ B_i, & p_1 < i \leq p_2 \\ A_i, & i > p_2 \end{cases}$$

Two-point crossover can be generalized to n-point crossover by generating n points and swapping segments between the parents to create an offspring.

Uniform crossover generates an offspring by randomly selecting one of the parents from which to inherit each gene.

Mutation operators were initially conceived as background processes which bore little importance to the evolutionary process [11]. However, they are a powerful tool to enable further exploration of the search space and a timely mutation can potentially save a population trapped in a local optimum. Usually, low mutation rates are employed in GAs as a way to occasionally introduce new genetic material into the population.

A selection operator is in charge of deciding which candidate solutions will progress into the next generation and which ones will perish. This is often done by assigning a solution with better fitness a greater chance of being selected (Roulette Wheel selection) or by selecting a subset of the population and selecting the one with best fitness (Tournament selection)[12].

The general structure of a genetic algorithm is as follows:

1. Initialize the population.
2. Evaluate the fitness of every individual.
3. Apply crossover and mutation operators.
4. Apply the selection operators to decide the individuals who will progress to the next generation.
5. If a stopping condition has not been met, return to step 2. Otherwise, return the best individual.

3 Associative Models

An associative memory is a system which takes a codified input or pattern and produces an output which can be either a class label or another pattern. One of the main advantages of associative memories is that, when they are correctly designed, they can accurately recover a pattern even if it has been altered. This robustness against alterations makes them attractive for applications in which the input patterns are likely to be noisy.

A comprehensive review on associative memory models and their inner workings is presented in [16]. This section will focus on presenting the specific model used in this work, the Hybrid Associative Classifier with Translation (CHAT).

As the name suggests, CHAT is a hybrid between two prior associative models: Steinbuch's Lernmatrix and Anderson & Kohonen's Linear Associator [17]; specifically, it adopts the learning phase of the latter and the recovery phase of the former. The combination of these two models allows the CHAT to overcome some of the roadblocks exhibited by its predecessors: it can work with real-valued inputs (unlike the *Lernmatrix*) and it does not need its inputs to consist of a set of orthonormal vectors as the *Linear Associator* does. Aside from this combination, the input patterns are averaged and subsequently this mean vector is substracted from each pattern to assemble a new translated training set which improves classification accuracy.

The CHAT algorithm works as outlined below:

1. For each $x^1 ... x^p$ patterns in the training set, compute the mean vector

$$\bar{x} = \frac{1}{p} \sum_{i=1}^{p} x^i$$

2. Compute the translated input patterns $x^{1'}, ..., x^{p'}$

$$x^{i'} = x^i - \bar{x}$$

3. For each $x^{i'}$ which belongs to the k-th class, create a class vector y^i which has a value of 1 only in the k-th coordinate

$$y_m^i = \begin{cases} 1, & m = k \\ 0, & otherwise \end{cases}$$

4. Apply the learning phase of the *Linear Associator*
5. Apply the recovery phase of the *Lernmatrix*

4 Proposed Method

As stated above, this paper presents a wrapper-style feature selection model based on a Genetic Algorithm and an associative classifier. The objective is to build a feature selector which is relatively fast and robust against noise. This work borrows from [16] the classification algorithm employed; however, it attempts to obtain a feature subset which yields a better classification performance by exploring a portion of the solution space.

The proposed feature selection method is described in graphical form in Figure 1.

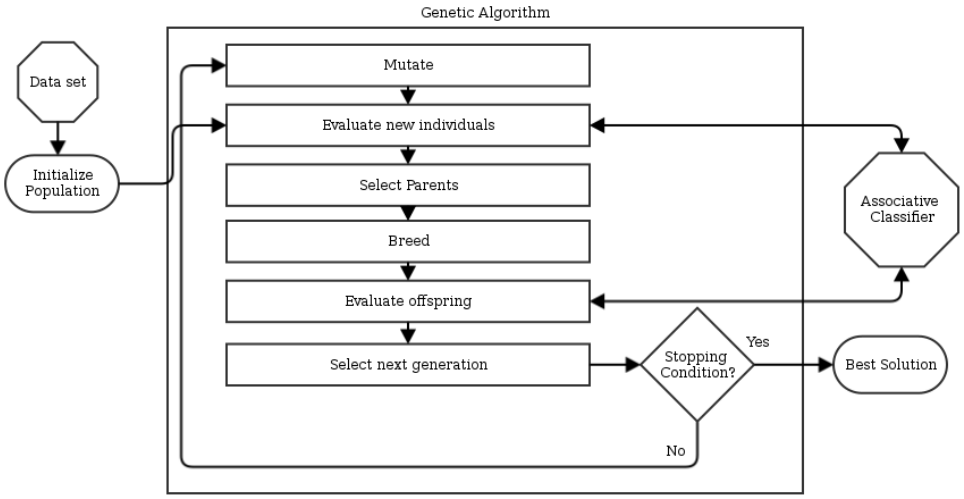

Fig. 1. Proposed method

The method follows the footprints of a GA: it begins by initializing a random population of candidate feature subsets and starting the search. The GA evolves the population by applying the mutation, crossover and selection operators specified by the user. Those candidates which appear more promising progress through the generations and gradually approximate the optimal solution. To obtain the fitness value of a candidate solution, a CHAT classifier is trained and then used to classify pattern in the data set using the subset of features selected by the solution. The estimation method used to obtain classification performance was 10-fold cross-validation.

The proposed algorithm was implemented in the R programming language [18], making use of its *parallel* package to speed up computation. The experiments were run on a PC running Arch Linux with an AMD A10 processor and 8GB of RAM.

5 Experimental Results

The aim of this work was to find a method to obtain better results than those produced by the algorithm presented in [16], while evaluating a fraction of the possible solutions than an exhaustive search would. The experimental results of applying a GA-based search to feature selection on the data suggest that this is in fact a feasible approach for data sets consisting of more than a dozen features, obtaining optimal or close to optimal fitness values with an important

feature amount reduction while evaluating much fewer solutions compared to an exhaustive search.

Data sets taken from the UCI Machine Learning Repository [19] were employed for the experimental part of this project. Specifically, the Breast Cancer, Heart Disease, Credit Approval and Hepatitis data sets were chosen.

The parameters shown in Table 1 were constant for all experiments.

A smaller population was used for the breast cancer data set to avoid reaching the target fitness in the first generation with a great likelihood (due to there being only 511 possible subsets) and thus failing to demonstrate the genetic search.

An exhaustive search was run prior to applying the GA in order to have two reference values against which to compare the performance of our method: number of candidate solutions evaluated and classification accuracy.

The search was stopped when one of the following conditions was reached:

1. The maximum number of generations was reached.
2. The maximum fitness computed by exhaustive search was reached.
3. 25 generations passed with no improvement on the best solution.

The population size was maintained at a constant number. To achieve this, after producing the offspring solutions, the n offspring replaced the n worst solutions of the original population.

Rather than time, this experiment measured the number of candidate solutions evaluated; unlike running time, this amount is independent of the machine used to run the experiments. For this purpose the program counted the number of solutions generated and evaluated.

Six different combinations of selection and crossover operations were tested in the experiments. Table 2 details each of these combinations. Mutation was kept at a constant probability of 0.15.

Tables 3 to 6 show the detailed results of the experiments conducted for this work. Specifically, they show the number of the experiment, the number of generations passed before stopping, the number of features selected and the fitness of the best solution found. Additionally, it presents the percentage of solutions evaluated relative to an exhaustive search, the percentage of approximation of the optimal solution and the dimensionality reduction of the best solution.

Table 1. Experiment parameters

Dataset	Features	Population	Offspring	Generations	Max Fitness
Breast	9	10	4	50	0.9780
Heart	13	100	40	50	0.8333
Credit	14	100	40	50	0.8564
Hepatitis	19	100	40	50	0.8507

Table 2. Operator combinations

Experiment	Crossover	Selection
1	One-point	Roulette
2	One-point	Tournament
3	Two-point	Roulette
4	Two-point	Tournament
5	Uniform	Roulette
6	Uniform	Tournament

Table 3. Experimental results - Breast Cancer dataset

Type	Gen.	Feat.	Fitness	Eval.	% Evaluated	% Optimal	% Reduction
1	18	4	0.978026	182	35.62%	100.00%	55.56%
2	28	7	0.976577	272	53.23%	99.85%	22.22%
3	34	7	0.976556	326	63.80%	99.85%	22.22%
4	12	5	0.978026	128	25.05%	100.00%	44.44%
5	6	6	0.978026	74	14.48%	100.00%	33.33%
6	28	2	0.976556	272	53.23%	99.85%	77.78%

Table 4. Experimental results - Heart Disease dataset

Type	Gen.	Feat.	Fitness	Eval.	% Evaluated	% Optimal	% Reduction
1	12	5	0.833333	1280	15.63%	100.00%	61.54%
2	10	5	0.833333	1100	13.43%	100.00%	61.54%
3	13	5	0.833333	1370	16.72%	100.00%	61.54%
4	13	5	0.833333	1370	16.72%	100.00%	61.54%
5	34	4	0.82963	3260	39.79%	99.56%	69.23%
6	6	5	0.833333	740	9.03%	100.00%	61.54%

Table 5. Experimental results - Credit Approval dataset

Type	Gen.	Feat.	Fitness	Eval.	% Evaluated	% Optimal	% Reduction
1	16	5	0.856522	1640	10.01%	100.00%	64.29%
2	16	3	0.856522	1640	10.01%	100.00%	78.57%
3	26	5	0.83875	2540	15.50%	97.93%	64.29%
4	28	1	0.855072	2720	16.60%	99.83%	92.86%
5	28	6	0.855072	2720	16.60%	99.83%	57.14%
6	29	8	0.855072	2810	17.15%	99.83%	42.86%

Table 6. Experimental results - Hepatitis dataset

Type	Gen.	Feat.	Fitness	Eval.	% Evaluated	% Optimal	% Reduction
1	29	5	0.832083	2810	0.54%	97.80%	73.68%
2	43	7	0.83875	4070	0.78%	98.58%	63.16%
3	65	6	0.839167	6050	1.15%	98.63%	68.42%
4	60	4	0.83875	5600	1.07%	98.58%	78.95%
5	64	9	0.839167	5960	1.14%	98.63%	52.63%
6	56	5	0.83875	5240	1.00%	98.58%	73.68%

It is clear from these results that the GA is capable of approximating the optimal feature subset to within 3% of classification accuracy; even outright finding the optimal solution in some test runs.

With regards to dimensionality reduction, the results show important cutdown on the number of features selected. For the bigger data sets, a reduction of half or more of the features can be observed in all cases. For the Breast Cancer data set, the dimensionality reduction is less noticeable; however, this is due to the small number of features.

Perhaps the key result observed in these experiments is the one concerning the time needed to approximate an optimal feature subset. It can be seen from the result tables that even the most inefficient experiments (the ones on the Breast Cancer data set) required less evaluations than an exhaustive search to find the optimal solution or one extremely similar to it in performance. The other data sets exhibited similar behavior: the more features, the less evaluations (relative to an exhaustive search) were needed to obtain a near-optimal solution. This is exacerbated by the fact that some runs stopped after 25 generations with no improvement, meaning the near-optimal candidate was found much earlier than these results suggest.

The results of this experiment showcase genetic algorithms as a way to relatively quickly search for a solution which reduces the number of features considerably

and improves classification accuracy. A discussion on these results will be offered in the next section.

6 Conclusions

Feature selection is usually faced with a choice between maximizing the accuracy of the classifier and minimizing the time needed for finding a solution. Past approaches have focused on studying the relationship between features to make a decision or constructing a feature subset by gradually adding or removing features in a determined region of the search space.

This study presents a genetic algorithm-based approach which is able to start seeking a solution in multiple places of the search space and then moves along a promising direction which leads to a near-optimal solution. Variants of the most important genetic operators were tested and their results compared to those produced by an exhaustive search. The results showed that this approach to feature selection obtains an approximately optimal solution in less time than an exhaustive search would need; especially so on data sets where the amount of features starts to prohibit full exploration of the solution space. This suggests that GAs are an useful tool for feature selection, especially for larger data sets: their ability to preserve blocks of relevant features along the generations helps the algorithm to converge to a near-optimal solution relatively quickly.

However, some considerations have to be taken into account which limit the power of this approach to obtain the optimal feature subset. In the first place, for extremely large data sets the best solution is unknown and several local optimums can exist in the search space and the GA can get trapped around these local optimums. It is difficult, if not impossible, to detect whether the algorithm has converged at a less-than-desirable solution. Secondly, the associative classifier used is best suited for working on two-class problems, as it generally struggles with multi-class problems. As such, the results produced by applying this approach to this kind of data sets present lesser classification accuracy.

Currently further work is being done with this feature selection model. Larger data sets with hundreds to thousands of features are being tested with this method. Additionally, different genetic operators are being used in an attempt to find the combination of these which yields better results. Other kinds of evolutionary algorithms should be tested in the future in order to determine if they can produce similar or better results than GAs; similarly, future research should also focus on coupling the GA with other classifiers.

Acknowledgment

The authors of the present paper would like to thank the following institutions for their economical support to develop this work: Science and Technology National Council of Mexico, SNI, and National Polytechnic Institute of Mexico (Project No. 20140926), SIP, CIDETEC, and CIC.

References

1. Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence **97** (1997) 245–271
2. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97**(1) (1997) 273–324
3. Jain, A., Zongker, D.: Feature selection: evaluation, application, and small sample performance. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**(2) (1997) 153–158
4. Han, Y., Park, K., Lee, Y.: Confident wrapper-type semi-supervised feature selection using an ensemble classifier. In: 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC). (2011) 4581–4586
5. Li, G., Hu, X., Shen, X., Chen, X., Li, Z.: A novel unsupervised feature selection method for bioinformatics data sets through feature clustering. In: IEEE International Conference on Granular Computing, 2008. GrC 2008. (2008) 41–47
6. Yu, L., Liu, H., Guyon, I.: Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research **5** (2004) 1205–1224
7. Fleuret, F., Guyon, I.: Fast binary feature selection with conditional mutual information. Journal of Machine Learning Research **5** (2004) 1531–1555
8. Siedlecki, W., Sklansky, J.: A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters **10**(5) (1989) 335–347
9. Yang, J., Honavar, V.: Feature subset selection using a genetic algorithm. IEEE Intelligent Systems and their Applications **13**(2) (1998) 44–49
10. Aldape-Pérez, M., Yañez-Márquez, C., Camacho-Nieto, O., Ferreira-Santiago, A.: Selección de características utilizando el paradigma de memoria asociativa y computación paralela. Computación y Sistemas **17**(1) (2013) 41–52
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, USA (1992)
12. Engelbrecht, A.P.: Computational Intelligence: An Introduction. John Wiley & Sons (2007)
13. Mao-Rui, Z., Yong, S., Jian-fei, L., Da-wei, Y.: Improved genetic algorithm application in laser Anti-Missile systems. In: 2010 International Conference on E-Product E-Service and E-Entertainment (ICEEE). (2010) 1–4
14. Ng, T.T.H., Leng, G.S.B.: Application of genetic algorithms to conceptual design of a micro-air vehicle. Engineering Applications of Artificial Intelligence **15**(5) (2002) 439–445
15. Tang, K., Man, K., Kwong, S., He, Q.: Genetic algorithms and their applications. IEEE Signal Processing Magazine **13**(6) (1996) 22–37

16. Aldape Pérez, M.: Enfoque asociativo para la selección de rasgos. Thesis, Instituto Politécnico Nacional. Centro de Investigación en Computación (2011) Doctorado en Ciencias de la Computación.
17. Santiago-Montero, R.: Clasificador Híbrido De Patrones Basado En La Lernmatrix De Steinbuch Y El Linear Associator De Anderson-Kohonen. PhD thesis (2003)
18. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2014)
19. Bache, K., Lichman, M.: UCI machine learning repository. http://archive.ics.uci.edu/ml (2013)