Combining evolution techniques to estimate features' weights and the support sets system for ALVOT

Carrasco-Ochoa Jesús. A. & Martínez-Trinidad José Fco.
Departamento de Ciencias Computacionales
Instituto Nacional de Astrofísica, Optica y Electrónica.
Luis Enrique Erro No. 1 Sta. Ma. Tonanzintla, Puebla, México.
CP: 72840.
email: {ariel, fmartine}@inaoep.mx

Abstract

In this paper we propose an alternative method, based on evolution strategies and genetic algorithms, to estimate features' weights and the support sets system for the supervised classification model ALVOT (voting algorithms). Usually, this system is taken as the set of all typical testors, and features' weights are calculated using these testors. However, algorithms for calculating all typical testor have very high complexity, exponential in the worst case. Also, in some practical cases, the number of typical testors can be too large. This can become typical testors inapplicable for the classification stage. This problem, of high cardinality of the set of all typical testors, can cause that features' weights become too little for features which appear only in a few typical testors, including features which appear only in one typical testor. The proposed method allows generating features' weights, which maximizes ALVOT classification quality, with a support sets systems of limited size, but with a high efficiency for classification. Some performance examples, of the new method, are exposed. Quality for classification, of the results, is compared against typical testors option.

1 Introduction

ALVOT [1, 2] is a supervised classification model. It was developed in the framework of the logical combinatorial approach to pattern recognition [1]. This model is based on partial precedence. For ALVOT application it is necessary to define some parameters including features' weights and the support sets system, that indicates which parts of the objects are relevant to compare. It is very difficult to obtain values for features' weights from the specialist at the modeling process, so these values must be estimated from data. There are several

methods to estimate features' weights. In the framework of the logical combinatorial pattern recognition these weights are calculated from the set of all typical testors. Also, usually the set of all typical testors is taken as the support sets system. The use of all typical testors as support sets system gives good results at the classification stage, but there are two problems for using them: on the one hand, algorithms for calculating all typical testors have exponential complexity with regard to the number of features, so, for problems with many features the calculation of all typical testor is very expensive; on the other hand, the number of typical testors has an exponential bound, for this cause, typical testors can be too many to be useful at the classification stage (we have an example with 42 features where there are more than 300,000 typical testors). Then, use them to calculate features' weights becomes inadequate when they are too many, due to the number of typical testors is part of the denominator, in almost all expressions developed until now.

In this work we propose a new method based on evolution strategies, which allows estimating values for features' weights. This new method has as objective classification quality maximization. To evaluate classification efficiency it is necessary to have a support sets system, but it will be incongruent to use the set of all typical testors. Because of this, a method based on genetic algorithms is used to estimate the support sets system for each step. The last process uses as features' weights, the values that are been evaluated. Since the search strategy tries to maximize classification efficiency, the final values can have high classification efficiency, many times better than the efficiency of using the set of all typical testors. Additionally, the size of the support sets system is bounded, about its cardinality, by the size of the population maintained by the genetic algorithm.

In order to test the proposed method, we applied it over different data sets. Size and efficiency of the obtained features' weights and support set systems were compared against the typical testors option.

In the following sections the ALVOT model is described, concepts of testor and typical testor are given, testor theory approach to features' weights estimation is explained, a new method for estimating features' weights and the support sets system is exposed and experimentations are showed.

2 ALVOT

First, we define some concepts, which are useful to describe the voting algorithm model (ALVOT).

Definition 1 *A support sets system denoted by $\{\Omega\}$, is a set of subsets of features. This system indicates which parts of the objects will be compared. Each $\Omega \in \{\Omega\}$ is called support set.*

Definition 2 *Let Ω be a subset of the features. The Ω-part of an object O corresponding to Ω, denoted as ΩO, is the subdescription of O using only features of Ω.*

The voting algorithms model is based on the following basic ideas:

Analogy. There is a similarity function, which reflects how the analogy is made in the real problem.

Partialprecedence. The comparisons are not made between complete object descriptions but between subdescriptions previously selected.

Frequency. In voting algorithms an object belongs to a class if it is similar to enough objects of this class.

Each voting algorithm is determined by 6 parameters. The possibility of changing any of these parameters makes a family of this kind of algorithms.

The parameters that define a voting algorithm are:

$\{\Omega\}$ *SupportSetsSystem*

The support sets system determines which parts of the objects will be compared when the algorithm is applied. Any subset of the power set of the features can be used as support sets system, for example, all the subsets with a fixed cardinal, the set of all typical testors, etc.

β *SimilarityFunction*

The similarity function determines how subdescriptions will be compared. This function should reflect how the analogy between objects is handled in the real problem.

f *ObjectEvaluationFunctionforaFixSupportSet*

This function determines how much information is given by the similarity of a new object with each one of the sample objects, for a fix support set. The result of this function is called the vote given by each sample object to a new object with regard to a fix support set.

This function can take into consideration the weight of the evaluated sample object, and also, the weights of the features of the considered support set. Examples of this function are (1) and (2), where $\gamma(O_i)$ is the weight of the object O_i and $P(\Omega) = \sum_{x_i \in \Omega} P(x_i)$ being $P(x_i)$ the weight of the feature x_i.

$$f(O_i,O,\Omega) = \beta(\Omega O_i, \Omega O) \qquad (1)$$

$$f(O_i,O,\Omega) = g(O_i)P(\Omega)\beta(\Omega O_i, \Omega O) \qquad (2)$$

φ *ClassEvaluationFunctionforaFixSupportSet*

This function summarizes all object evaluations for a new object within each class, for a fix support set. The result of this function is called the vote given by each class to a new object with regard to a fix support set. Examples of this function are (3) and (4).

$$\varphi(j,O,\Omega) = \frac{1}{|K_j|} \sum_{O_i \in K_j} f(O_i,O,\Omega) \qquad (3)$$

$$\varphi(j,O,\Omega) = \begin{cases} 1 & if \ \frac{1}{|K_j|} \sum f(O_i,O,\Omega) \geq \lambda; \lambda > 0 \\ 0 & otherwise \end{cases} \qquad (4)$$

ψ *ClassEvaluationFunctionfortheSupportSetsSystem*

This function summarizes all class evaluations for a new object for the whole support sets system. The result of this function is called the vote given by each

class to a new object with regard to the whole support sets system. Examples of this function are (5) and (6).

$$\psi(j,O) = \frac{1}{|\{\Omega\}|} \sum_{\Omega \in \{\Omega\}} \varphi(j,O,\Omega) \tag{5}$$

$$\psi(j,O) = \begin{cases} 1 & if \ \frac{1}{|\{\Omega\}|} \sum_{\Omega \in \{\Omega\}} \varphi(j,O,\Omega) \geq \lambda; \lambda > 0 \\ 0 & otherwise \end{cases} \tag{6}$$

rSolutionRule

This function takes all global evaluations for each class (given by y) and decides which class or classes the new object belongs to. The solution rule takes the form $r(\psi(1,O),...,\psi(l,O)) = (\alpha_1(O),...,\alpha_n(O))$, where $\alpha_i(O)$ is 1 if the new object is assigned to the ith class, and 0 in otherwise. Examples of this function are (7) and (8).

$$\alpha_i(O) = \begin{cases} 1 & if \ \psi(i,O) > \psi(j,O) \ \forall j \neq i \\ 0 & otherwise \end{cases} \tag{7}$$

$$\alpha_i(O) = \begin{cases} 1 & if \ \psi(i,O) - r_1 > \psi(j,O) \ \forall j \neq i \\ 0 & otherwise \end{cases} \tag{8}$$

With these parameters voting algorithms work in the following 5 stages:

1. Model parameters determination.
2. Apply Object evaluation function for the new object with each support set.
3. Apply Class evaluation function for the new object with each support set.
4. Apply Class evaluation function for the new object with the whole support sets system.
5. Apply solution rule to determine which class or classes the new object belongs to.

3 Typical Testors

Into the framework of the Logical Combinatorial Pattern Recognition, feature selection is done using typical testors[1, 3]. For the ALVOT model, typical testors can be used as support sets system. A testor is defined as follows:

Definition 3 *A subset of features T is a testor if and only if when all features are eliminated, except those from T, there is not any pair of equal subdescriptions in different classes.*

The definition 3 says us that a testor is a subset of features, which allows complete differentiation of objects from different classes. Within the set of all testors, there are some testors, which are irreducible. This kind of testors is called typical testors. Typical testors are defined as follows:

Definition 4 *A subset of features T is a typical testor if and only if T is a testor and there is not any other testor T' such that $T' \subset T$.*

The definition 4 says us that a typical testor is a testor where every feature essential, this is, if any of them is eliminated the resultant set is not a testor.

The calculation of all typical testors is important because they are all subsets features that can completely differentiate objects from different classes and they are not redundant. But, if the definition 4 is used to calculate all typical testors, all 2^n different subsets of features must be verified. For each one, you have to decide if it is a testor, then you must decide if it is typical.

Since this process has exponential complexity with regard to the number features, many different algorithms to calculate all typical testors have been developed, but all of them have exponential complexity for the worst case.

Feature Weights Estimation

The approach based on Testor Theory was first proposed by Dimitriev et al. and the basic idea is the following: A testor is a feature subset, which does not confuse any pair of subdescriptions from different classes. Moving from a testor to a typical testor (eliminating features, when it is possible) we get an irreducible combination of features, where each feature is essential in order to keep differences between classes. This is the property that distinguishes each typical testor. It is natural to suppose that if a feature appears many times in the typical testors, it is more difficult to disregard it. That is, we could say it is more useful to differentiate between classes. Based on this idea Dimitriev et al. give a definition of feature's weight as the relative frequency of the occurrence each feature in the set of all typical testors. Let τ be the number of typical testors for a certain problem. Let $\tau(i)$ be the number of typical testors, which contain feature x_i. Feature's weight of x_i is given by (9).

$$P(x_i) = \frac{\tau(i)}{\tau} \qquad (9)$$

where $i = 1, ..., n$, $x_i \in R$.

This expression is based on the following idea: a feature is more important when it appears more times into the set of all typical testors.

Our proposal

this section, we present an evolution strategy to compute features' weights and a genetic algorithm to compute support sets systems associated to each combination of features' weights. First, we will explain the genetic algorithm, which is used as fitness function by the evolution strategy.

5.1 Genetic Algorithm

The individuals handled by the genetic algorithm [5, 6] will be the support sets. These sets are represented as n-uples formed by the values 0 and 1 (genes), these values represent the absence or presence of a feature, and n represents the total number of features.

We will denote the individuals as Ω_i $i = 1, ..., p$, where p is the number of individuals in the population.

The fitness function used by the algorithm is (10).

$$Fitness(\Omega_i) = \frac{Number\ of\ well\ classified\ objects}{Total\ number\ of\ classified\ objects} \qquad (10)$$

This function evaluates how many objects were well classified by ALVOT regarding the total of classified objects. The support sets system in this case is $\{\Omega_i\}$. If an individual (a support set) classifies more objects, then it will be more capable.

The crossover operator used in the algorithm is as follows: First, the population is ordered in descending form regarding individuals' fitness. The first individual (that has the highest fitness) and the last one (that has the smallest fitness) are taken and crossed using the one-point crossover using $n/2$ (the half of the individual) as crossing point. After, the second individual is crossed with the penultimate one; this process is repeated until finishing with the population.

In the genetic algorithm the generative mutation is used, this operator randomly takes an individual's gene and changes its value, if the gene has value 0, the new value will be 1 and vice versa.

In general way, the proposed algorithm woks as follows:

1. First, randomly the initial population is generated. The population size and iterations number are entrance parameters of the algorithm.

2. Then, the population's individuals are evaluated to determine their fitness, and then they are crossed using the crossover operator. The fitness is also evaluated for the new obtained population.

3. After, for each population's individual the mutation operator is applied. The fitness is also evaluated for the new obtained population.

4. Finally, the population's individuals together with the individuals obtained by crossing and mutation operators are ordered according to their fitness and the best are chosen (taking account the population size) to form a new population that will be used in the next iteration of the algorithm.

When the algorithm finishes then the efficiency of ALVOT is evaluated using both the computed support sets system (final population) and the features' weights W. This efficiency value will be used by the evolution strategy.

```
Genetic Algorithm (SSS-GA)
Input
 W: features' weights.
 size: size of population
 Num_iter: Number of iterations
1.Generate_initial_population(population,size)
2.Evaluate_population(population)
3.Repeat for i=1 until i=Num_iter
4.Crossover(population,population2)
5.Evaluate_population(population2)
6.Mutation(population,population3)
7.Evaluate_population(population3)
8.Pick_out_next_population(population,population2,population3)
9.Write(population)
10.Return ALVOT(population,W) END.
```

In the algorithm, population2 has a new population after applying crossover operator to individuals of the original population; population3 has new individuals after applying the mutation operator to individuals of the original population.

The function Pick_out_next_population chooses from population, population2, and population3 to the most capable individuals (those with highest fitness) that will form a new population for the next iteration.

5.2 Evolution Strategy

The individuals handled by the evolution strategy [7] will be the features' weights. These weights are represented as n-uples formed by values in the interval [0,1], these values represent the weight of each feature, and n represents total number of features.

We will denote the individuals as W_i $i = 1, ..., p$, where p is the number of individuals in the population.

The fitness function used by the strategy was the genetic algorithm above described, where population size and number of iterations were 15 and 10 respectively. That is to say, $Fitness(W_i) = SSS\text{-}GA(W_i)$. This function evaluates how many objects were well classified by ALVOT regarding the total of classified objects. In this case, the genetic algorithm ($SSS\text{-}GA$) is called passing as parameter the individual W_i. In this way, a support sets system is computed for and the efficiency of ALVOT using W_i and $\{\Omega\}_i$ as parameters is evaluated. an individual W_i (using their $\{\Omega\}_i$) correctly classifies more objects, then it will be more capable.

The crossover operator used in the strategy is one-point crossover. But in this case the individuals are n-uples of real numbers. The uniform mutation is used; this operator randomly takes an individual's gene and changes its value a random number in the interval [0,1].

```
Evolution Strategy (ES)
Input:
 size:  size of population
 Num_iter:  number of iterations
1.Generate_initial_population(population,size)
2.Evaluate_population(population)
3.Repeat for i=1 until i=Num_iter
4.Crossover(population,population2)
5.Evaluate_population(population2)
6.Mutation(population,population3)
7.Evaluate_population(population3)
8.Pick_out_next_population(population,population2,population3)
9.Write(population) END.
```

The evolution strategy has the same structure than the genetic algorithm except that the function Evaluate_population calls to *SSS-GA* for each individual W_i in order to compute a support sets system $\{\Omega\}_i$. In this way, the fitness of W_i is computed as the efficiency of ALVOT using W_i and $\{\Omega\}_i$ as parameters. This is shown in the figure 1. At end of the process we can select the individual with highest fitness (and its support sets system) as the features' weights and the support sets system for ALVOT in the classification stage.

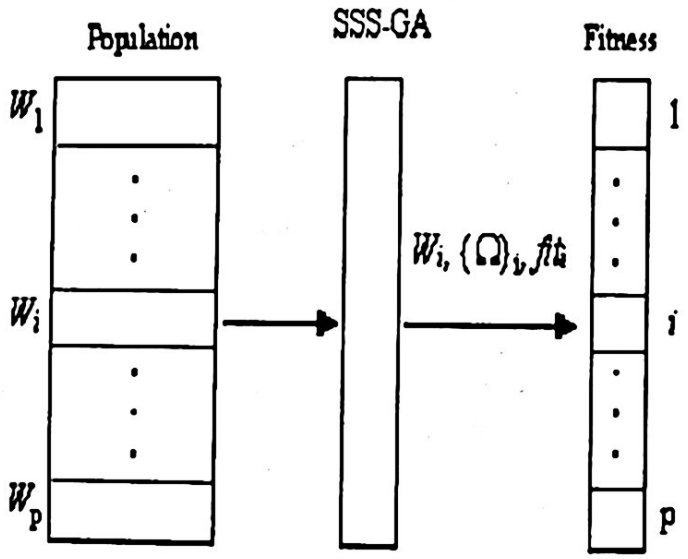

Figure 1: Diagram of the method proposed to estimate features weights and support sets system.

6 Experimental results

In this section, some experiments to estimate the features' weights using an evolution strategy and the estimation of the support sets system with the genetic algorithm are presented. In all the experimentations, a comparison of the classification efficiency using the features' weights and the support sets system obtained with the proposed method, against the typical testor option, is made.

The databases for experimentations were taken from [8].

The first experiment was carried out with the zoo database. This database contains 101 descriptions of animals grouped in 7 classes, each description is given in terms of 16 features.

In this case there are 33 typical testors. The efficiency of classification of ALVOT using these 33 testors like support sets system is of 1.0. The measure of efficiency was obtained classifying the training sample and taking the number of successes divided by the number of objects in the sample.

In the figure 2, the classification efficiency of ALVOT when W_f, the most capable individual in the last iteration of the evolution strategy, and $\{\Omega\}_f$ (the support set system associated to W_f) were used as parameters is shown. In this case, the horizontal axis represents the variation of population size in the genetic algorithm to compute the support sets system. The vertical axis is the reached efficiency using the typical testors and the features' weights computed on basis of them. For the evolution strategy we handle a fixed population size.

In the figure 2a it is seen that, for zoo database the efficiency cannot be improved but with a population of 10 support sets, three times less than the total number of typical testors, the same efficiency can be achieved as with the 33 typical testors.

The second experiment was carried out with the import85 database. This database has 205 car specifications and they are distributed in 7 classes, each specification is made using 25 features. The number of computed typical testors is 6. The classification efficiency of ALVOT using the 6 testores is 0.7512. In the figure 2b, the obtained results are shown.

The third experimentation was made with the crx database. This database has 690 credit applications, there are 2 classes, and each application is given in terms of 15 features. This example only has one typical testor. The classification efficiency of ALVOT taking account the typical testor is 0.9623. In the figure 2c, the results obtained are shown.

The last experimentation was made using the hepatitis database. This database has 155 patients with hepatitis, there are 2 classes; each patient description has 19 features. Here, the number of typical testors is 404. The classification efficiency of ALVOT using the typical testors is 0.9870. In the figure 2d it is seen the classification efficiency of ALVOT using the features' weights and the support sets system computed with our method.

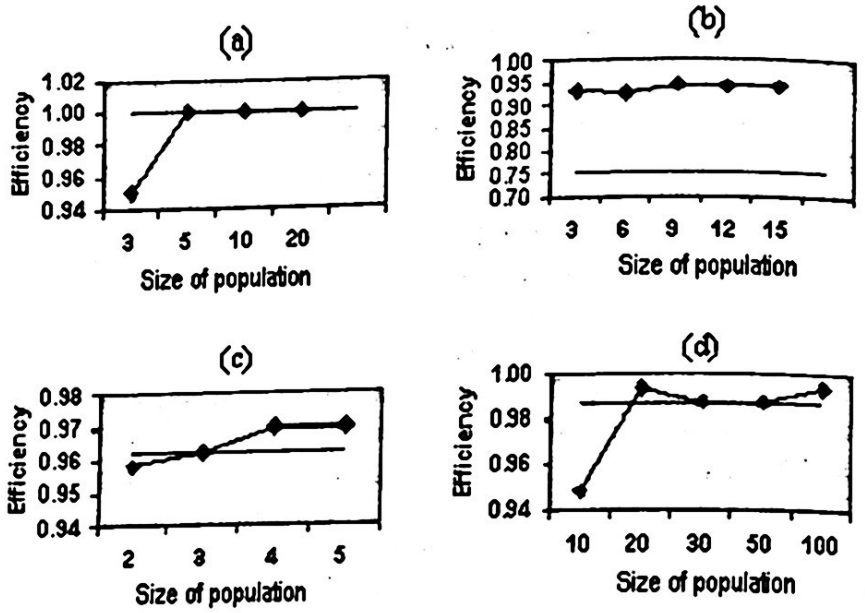

Figure 2: Efficiency of ALVOT for a)zoo, b)import85, c)crx, and d)hepatitis, using the features' weights and the support sets systems computed by our method.

7 Conclusions

In this paper, a new method, based on evolution strategy and genetic algorithms, to estimate the features' weights and the support sets system for the model of supervised classification ALVOT, is presented.

From experimentation, we can conclude that a better classification efficiency than using typical testors and features' weights based on them may be reached; but applying our method we can get a support sets system with lesser cardinality. It results in a lesser time for classification stage.

Additionally, since the problem of computing all the typical testors has exponential complexity and the proposed method can be proved that has polinomial complexity. For problems with many features may be impossible in the practice to calculate the typical testors, however the proposed method can be applied.

Another point to highlight is the fact that the number of typical testors is bounded exponentially, so that, in some cases they can be too many to be useful in the classification stage. Contrarily, in the proposed method the size of the support sets system is one of the parameters, which allows fixing the size of the system according to the practical requirements.

The main application of the here presented method is for big problems where using typical testors is not feasible. Since they cannot be computed or they turn out to be too many.

Acknowledgement.- This work was financially supported by CONACyT (Mexico) through projects J38707-A and I38436-A.

References

Martínez-Trinidad J. F.& Guzmán-Arenas A. (2001). The logical combinatorial approach to pattern recognition an overview through selected works, *Pattern Recognition* 34(4) 741-751.

Ruiz-Shulcloper J. & Lazo-Cortés M. (1999). Mathematical Algorithms for the Supervised Classification Based on Fuzzy Partial Precedence, *Mathematical and Computer Modelling*, 29(4), 111-119.

Lazo-Cortes M., Ruiz-Shulcloper J. & Alba-Cabrera E. (2001). An overview of the evolution of the concept of testor, *Pattern Recognition*, 34(4), 753-762.

Dmitriev A.N., Zhuravlev Y.I., & Krendeliev F.P. (1966). About mathematical principles of objects and phenomena classification, *Diskretni Analiz* 7, 3-15. (In Russian).

Mitchel M. (1996). *An introduction to genetic algorithms*, MIT Press.

Goldberg D. (1989). *Genetic algorithms in search, optimization and machine learning*, Addison Wesley.

Beyer H. G. (2001). *Theory of Evolution Strategies*, Springer Verlag.

http://www-old.ics.uci.edu/pub/machine-learning-databases/