# Constructing X-of-N Attributes with a Genetic Algorithm

**Otavio Larsen** [1] **Alex Freitas** [2] **Julio C. Nievola** [1]

[1] Postgraduate Program in Applied Computer Science
Pontificia Universidade Catolica do Parana
Rua Imaculada Conceicao, 1155
Curitiba - PR 80215-901 Brazil
{larsen, nievola}@ppgia.pucpr.br

[2] Computing Laboratory
University of Kent at Canterbury
Canterbury, CT2 7NF. UK
A.A.Freitas@ukc.ac.uk
http://www.cs.ukc.ac.uk/people/staff/aaf

**Abstract***: We propose a new Genetic Algorithm (GA) for constructive induction. The goal of the GA is to construct new X-of-N attributes out of the original attributes of a given data set, in order to improve the effectiveness of a data mining algorithm to be applied to that data set. The GA follows the preprocessing approach for constructive induction, so that the new attributes constructed by the GA can be given (together with the original attributes) to any data mining algorithm. In this paper we use as a data mining algorithm C4.5, which is a well-known decision-tree induction algorithm. In order to evaluate the effectiveness of the new attributes constructed by the GA, we compare the performance of C4.5 with and without the new attributes constructed by the GA across several data sets.*

**Keywords***: Constructive Induction, X-of-N Attributes, Genetic Algorithm, Data Mining.*

## 1. Introduction

In essence, data mining consists of extracting knowledge from data. One of the most popular tasks of data mining is classification [6]. In this task each record (also called an example, or a case) belongs to a class, among a predefined set of classes. The goal is to predict the class of a record based on the values of the record's attributes.

The predictive accuracy obtained by a classification algorithm is strongly dependent on the quality of the attributes of the data being mined. When the attributes have a low quality, in the sense of being little relevant for predicting the class of a record, the predictive accuracy will tend to be low. The majority of classification algorithms use only the original attributes of the data being mined. Hence, these algorithms are quite sensitive to the quality of the original attributes.

In order to address this problem, this paper proposes a new Genetic Algorithm (GA) for constructing X-of-N attributes. The X-of-N representation was chosen due to its great expressiveness power, as will be discussed in section 2; whereas the paradigm of GA was chosen due to its ability in coping with attribute interactions, as will be discussed in section 3.

The remainder of this paper is organized as follows. Section 2 presents a brief overview of attribute construction. Section 3 proposes a new GA for constructing X-of-N attributes. Section 4 reports computational results evaluating the quality of the attributes constructed by the proposed GA. Finally, section 5 concludes the paper.

## 2. An Overview of Attribute Construction

Attribute construction, or constructive induction, consists of constructing new attributes by applying some operations/functions to the original attributes, so that the new attributes make the classification problem easier for a data mining algorithm. The motivation for attribute construction tends to be particularly strong in real-world database systems, since it is often the case that the data stored in these systems was not collected for data mining purposes [3]. In these cases predictive accuracy can often be significantly improved by constructing new attributes which are more relevant (for predicting the class of an example) than the original attributes.

Many constructive induction algorithms work by simply constructing conjunctions and/or disjunctions of attribute-value pairs that occur in the data being mined. This kind of representation has a limited expressiveness power to represent attribute interactions. A much more expressive representation is X-of-N [14], [15]. An X-of-N condition consists of a set of N attribute-value pairs. The value of an X-of-N condition for a given example (record) is the number of attribute-value pairs of the example that match with the N attribute-value pairs of the condition. For instance, consider the following X-of-N condition: X-of-{"Sex = male", "Age < 21", "Salary = high"}. Suppose that a given example has the following attribute-value pairs: {"Sex = male", "Age = 51", "Salary = high"}. This example has 2 out of the 3 attribute-value pairs of the X-of-N condition, so that the value of the X-of-N condition for this example is 2. Note that an X-of-N condition can take on any integer value in the range 0...N, whereas a conjunction or disjunction of attribute-value pairs can take on only boolean values.

Attribute construction methods can be roughly categorized into the preprocessing and interleaving approaches [7]. In the preprocessing approach the attribute construction method is run separated from the classification algorithm, and the constructed attributes can be given to any classification algorithm. This has the advantage of generality and relatively short processing times, by comparison with the interleaving approach. By contrast, in the interleaving approach the attribute construction method and the data mining algorithm are intertwined into a single algorithm. This has the advantage that the constructed attributes are "optimized" for the data mining algorithm in question, but it has the disadvantage of losing generality and increasing processing time in general. In this paper we follow the preprocessing approach, for the sake of generality and computational efficiency.

In the next section we describe in detail a new genetic algorithm for constructing X-of-N attributes. It should be noted that in general other evolutionary algorithms for attribute construction found in the literature construct new attributes using an attribute representation different from (and often with less expressiveness power than) the X-of-N representation. For instance, [1] constructs new attributes by using the logic XOR (eXclusive OR) operator; [8] uses the logical AND and NOT operators; and [9] uses arithmetic operators. To the best of our knowledge this paper is the first work to propose a genetic algorithm to construct X-of-N attributes.

# 3. A New Genetic Algorithm for Constructing X-of-N Attributes

Before we describe our GA in detail, we first briefly discuss the motivation to use a GA for this task. X-of-N attributes involve a lot of interaction between attributes – actually, this is the major source of their power. Therefore, intuitively it makes sense to use an attribute construction method that copes well with attribute interactions. It can be argued that GAs cope well with attribute interactions, as follows [3], [4].

First, GAs work with a population of candidate solutions (individuals) at a time, rather than with a single candidate solution at a time. The population individuals concurrently explore different parts of the search space. Second, crossover (the main genetic operator in GAs) modifies individuals on a several-attributes(genes)-at-a-time basis. Third, the fitness function evaluates an individual as a whole, rather than evaluating only a part of an individual. Overall, these characteristics lead a GA to perform a global search in the space of candidate solutions, improving its ability to cope with attribute interactions. In addition, there is some empirical evidence that GAs tend to cope better with attribute interaction than local search-based rule induction algorithms – see e.g. [2], [11].

### 3.1 Individual Representation and Fitness Function

In our GA an individual consists of the set of N attribute-value pairs composing a X-of-N attribute. Each attribute-value pair is of the form $A_i = V_{ij}$, where $A_i$ is the i-th attribute and $V_{ij}$ is the j-th value belonging to the domain of the $A_i$. The current version of our GA can cope only with categorical attributes. Hence, continuous attributes are discretized in a preprocessing step.

In order to represent X-of-N attributes of different sizes we use a variable-length individual representation, where the value of N (i.e. the number of attribute-value pairs in the X-of-N attribute) is an integer number varying from 2 to 7. The lower limit of 2 is natural because if N = 1 the X-of-N

attribute would degenerate into one of the original attribute-value pairs of the data being mined. The upper limit of 7 was chosen in order to avoid the construction of attributes too complex for the user - the particular value of 7 can be justified based on some psychological studies [10].

In order to implement this variable-length individual representation (at the phenotype level) we have used a fixed-length genotype. More precisely, the number of genes of the genotype is fixed at *d*, the number of attributes in the data being mined. Each gene contains not only an attribute-value pair of the form $A_i = V_{ij}$, as mentioned above, but also a flag, called the active bit $B_i$, which takes on the value 1 or 0 to indicate whether or not, respectively, the i-th attribute-value pair is present in the decoded X-of-N attribute (phenotype). The general structure of the genotype is shown in Figure 1, where *d* is the number of attributes of the data set being mined.

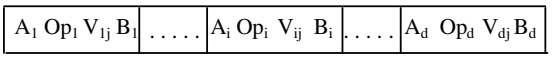


**Figure 1:** General structure of the genotype of an individual

In our GA the fitness function is defined as the information gain ratio of the constructed attribute. The information gain ratio (IGR) measure is also used to evaluate the quality of an attribute in the well-known C4.5 algorithm [12], and a detailed description of this measure can be found in that reference.

### 3.2 Selection Method, Genetic Operators and Niching

We used a version of tournament selection somewhat tailored for our data mining-oriented fitness function, as follows. In a conventional version of tournament selection, first *k* individuals are randomly picked, with replacement, from the population. Then the individual with the best fitness value, out of the *k* individuals, is selected as the winner of the tournament. This process is repeated *P* times, where *P* is the population size.

However, in our attribute construction problem, the selection of the individual with the largest value of the fitness function (i.e., IGR) could sometimes lead to a poor choice of attribute (individual). The reason is that, since the IGR is computed by dividing the information gain of an attribute *A* by the amount of information of *A* [12], an attribute *A* with low information gain could be selected just because the amount of information of *A* is low. This would be clearly undesirable, since *A* would have a low predictive power. In order to avoid this problem, we have modified our tournament selection procedure so that the winner of the tournament is the individual with the largest fitness among the set of individuals whose Information Gain is greater than or equal to the average Information Gain of all the *k* individuals playing the tournament. (We have used a tournament size (*k*) of 5.) This constraint for the selection of the best attribute is also used in C4.5 [12].

Once the tournament selection process has selected *P* individuals, these individuals undergo crossover and mutation with user-defined probabilities. We used the well-known uniform crossover operator. We chose this crossover operator because it avoids the positional bias inherent in other crossover operators [13]. In other words, genes are swapped in a way independent of the position of the gene inside the genotype of individuals. In our target problem this is desirable because, from a logical viewpoint, an individual consists of an *unordered* set of genes, where the position of a gene is irrelevant to determine its meaning.

We used a simple mutation operator, as follows. For each individual that will undergo mutation, the id of the gene (attribute-value pair) to undergo mutation is randomly generated with a uniform probability distribution. Then it is randomly decided whether the part of the gene that will undergo mutation is the value of the corresponding attribute or the corresponding active bit (see Section 3.1). In the former case, the attribute value is replaced by a new attribute value belonging to the domain of the corresponding attribute, whereas in the latter case the value of the active bit is flipped. The probabilities of crossover and mutation were set to 80% and 1%, respectively.

We also use elitism – i.e., at the end of each generation the best individual of the current generation is passed unaltered into the next generation. In our case there is, however, a caveat in the selection of the best individual of the current generation. We select the best individual of the current generation as the

individual with the largest fitness among the set of individuals whose Information Gain is greater than or equal to the average Information Gain of all individuals of the current generation, for the same reason as explained above in our modification of tournament selection.

Finally, we have also used a conventional niching method, namely fitness sharing [5], to foster population diversity and avoid premature convergence to a local maxima. The basic idea of fitness sharing is to share the fitness of an individual with other individuals that are "similar" (or "close") to it. In other words, the fitness of an individual is reduced in proportion to the number of individuals that are within its "niche". This size of a niche is determined by a parameter, called $\theta_{share}$, which can also be interpreted as a threshold of dissimilarity. Hence, if the distance between two individuals is greater than $\theta_{share}$, they do not affect each other. However, if their distance is smaller than or equal to $\theta_{share}$, their fitness will be reduced. The performance of the method depends on the value of the parameter $\theta_{share}$. We did some experiments to tune the value of this parameter, as will be discussed in section 4. In general, when using fitness sharing, the similarity between two individuals can be measured in genotypic space or phenotypic space. In our work similarity was measured on the genotypic space, since in our application this is considerably faster than measuring similarity on the phenotypic space.

### 3.3 Result Designation

Once a GA run is over, the solution (constructed attribute) returned to the user is the best individual of the last generation. However, there is a caveat in the choice of the individual to be returned to the user. The best individual of the last generation is selected by using the same method used for selecting the best individual for elitism purposes, as explained in the previous subsection.

## 4. Computational Results

In order to evaluate how good the new attributes constructed by the GA are, we have compared the performance of C4.5 using only the original attributes with the performance of C4.5 using both the original attributes and the new attributes constructed by the GA. Hereafter we refer to the former and to the latter as the original data set and the extended data set, respectively. C4.5 is a very well-known data mining/machine learning algorithm that builds a decision tree, and it is described in detail in [12].

The performance of C4.5 in both data sets – i.e. the original data set and the data set extended with X-of-N attributes – was measured with respect to the classification error rate. In most of the experiments (the exceptions are mentioned below) the error rate was measured by using a 10-fold cross-validation procedure [6]. As a result of this procedure, the results reported here are the average of results measured on a test set unseen during training. The experiments used public-domain data sets of the UCI data set repository, available from http://www.ics.uci.edu/~mlearn/MLRepository.html.

As mentioned in the previous section, we did some preliminary experiments to set the value of the parameter $\theta_{share}$ of fitness sharing. In order to set this value, we used three data sets from the UCI repository, namely Ljubljana breast cancer, Voting and Post-operative. The best results were obtained with the $\theta_{share}$ value of 0.1, on a normalized scale of 0 to 1. This value has consistently led to the best classification error rate results in all three data sets. Therefore, we fixed the value of $\theta_{share}$ at 0.1 and used this fixed value in our final experiments with seven other data sets of the UCI data repository, whose results are reported next. (Of course, we are not reporting the classification error rate of our method for the three above-mentioned data sets, since those data sets were explicitly used to set the value of a parameter of the GA via a cross-validation procedure – i.e., the results for those data sets are an unfair, optimistically-biased estimate of the true classification error rate associated with the GA.) The seven data sets used in our final experiments are described in Table 1.

The results of the experiments are shown in Table 2. The results for the first four data sets of Table 2 were produced by a 10-fold cross-validation procedure. The results for the last three data sets (the monks data sets) were obtained by using a single partition of the data into training and test sets, since these data sets already come with a pre-defined partition. The second and third columns of Table 2 show the error rate obtained by C4.5 in the original data set and the extended data set (with the new X-

of-N attribute), respectively. The numbers after the symbol "±" denote standard deviations. For each data set, the difference in the error rates of the second and third columns is deemed to be significant when the two error rate intervals (taking into account the standard deviations) do not overlap. When the error rate of the "original + X-of-N" attributes is significantly better (worse) than the error rate of the "original" attributes, there is a "(+)" ("(–)") sign in the third column, indicating that the attribute constructed by the GA significantly improved (degraded) the classification performance of C4.5.

**Table 1: Main characteristics of the data sets used in our final experiments**

| Data Set | No. of examples | No. of attributes | No. of classes |
|---|---|---|---|
| Hepatitis | 155 | 19 | 2 |
| Wisconsin breast cancer | 699 | 10 | 2 |
| tic-tac-toe | 958 | 9 | 2 |
| Promoters | 106 | 57 | 2 |
| monks-1 | 432 | 7 | 2 |
| monks-2 | 432 | 7 | 2 |
| monks-3 | 432 | 7 | 2 |

Observing the occurrences of "(+)" and "(–)" signs in the third column, one can see that the attribute constructed by the GA significantly improved the performance of C4.5 in three data sets, and it significantly degraded the performance of C4.5 in just one data set. In the other three data sets the difference in the error rates was not significant. In addition a visual inspection of the trees generated by C4.5 show that the new X-of-N attributes constructed by the GA have been massively used.

**Table 2: Error rate obtained by C4.5 in seven data sets**

| Data Set | Error Rate (%) | |
|---|---|---|
| | Original attributes | original + X-of-N attrib. |
| hepatitis | 22.84 ± 1.83 | 26.34 ± 4.99 |
| Wisconsin breast cancer | 4.57 ± 0.64 | 4.83 ± 0.78 |
| tic-tac-toe | 14.31 ± 1.14 | 5.34 ± 0.47 (+) |
| promoters | 18.88 ± 2.19 | 13.25 ± 1.98 (+) |
| monks-1 | 0.00 ± 0.00 | 0.00 ± 0.00 |
| monks-2 | 29.60 ± 0,04 | 0.00 ± 0.00 (+) |
| monks-3 | 0.00 ± 0.00 | 2.80 ± 0,01 (–) |

## 5. Conclusions and Future Research

In this paper we presented a new Genetic Algorithm (GA) for constructive induction, which constructs new X-of-N attributes out of the original attributes of a given data set. We employed the C4.5

algorithm to compare the benefits of the constructed attributes in terms of classification error rate. The approach was tested in seven public-domain data sets.

The obtained results suggest that using our proposed method the error rate of C4.5 tends to be significantly improved (i.e., reduced) more often than it is significantly worsened (i.e., increased), and the generated X-of-N attributes are massively used in the tree constructed by C4.5.

The GA proposed in this work constructs only X-of-N attributes. In future research it would be interesting to develop new GAs that can construct other kinds of attributes as well, which would increase the autonomy and the flexibility of the GA.

***Acknowledgment***

This research project is financially supported by Motorola, Jaguariuna –SP, Brazil.

## References

[1] H. Bensusan and I. Kuscu. Constructive induction using genetic programming. *Proc. ICML'96's Workshop on Evolutionary Computing and Machine Learning*. 1996.

[2] V. Dhar, D. Chou and F. Provost. Discovering interesting patterns for investment decision making with GLOWER – a genetic learner overlaid with entropy reduction. *Data Mining and Knowledge Discovery journal, 4(4)*, 251-280. Oct. 2000.

[3] A.A. Freitas. Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review 16(3)*, Nov. 2001, pp. 177-199.

[4] A.A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, 2002.

[5] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. *Proc. Int. Conf. Genetic Algorithms (ICGA-87)*, 41-49. 1987.

[6] D. Hand. *Constructing and Assessing Classification Rules.* John Wiley & Sons, 1997.

[7] Y.-J. Hu. Constructive induction: covering attribute spectrum. In: H. Liu & H. Motoda (Eds.) *Feature Extraction, Construction and Selection: a data mining perspective*, 257-272. Kluwer, 1998.

[8] Y.-J. Hu. A genetic programming approach to constructive induction. *Genetic Programming 1998: Proc. 3rd Int. Conf.*, 146-151. Morgan Kaufmann, 1998.

[9] I. Kuscu. Generalisation and domain specific functions in genetic programming. *Proc. 2000 Congress on Evolutionary Computation (CEC-2000)*, 1393-1400. IEEE, 2000.

[10] G.A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review 63*, 81-96. 1956.

[11] A. Papagelis and D. Kalles. Breeding decision trees using evolutionary techniques. *Proc. 18th Int. Conf. on Machine Learning (ICML-2001)*, 393-400. San Mateo, CA: Morgan Kaufmann, 2001.

[12] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[13] G. Syswerda. Uniform crossover in genetic algorithms. *Proc. 2nd Int. Conf. Genetic Algorithms (ICGA-89)*, 2-9. 1989.

[14] Z. Zheng. Constructing nominal X-of-N attributes. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* 1064-1070. Morgan Kaufmann, 1995.

[15] Z. Zheng. Constructing X-of-N attributes for decision tree learning. *Machine Learning 40 (2000)*, 1-43.