Fitness Function Comparison for GA-based Feature Construction

Leila S. Shafti and Eduardo Pérez

Escuela Plitécnica Superior,
Universidad Autónoma de Madrid, E-28049, Spain
{leila.shafti,eduardo.perez}@uam.es
http://www.eps.uam.es

Abstract. When primitive data representation yields attribute interactions, learning requires feature construction. MFE2/GA, a GA-based feature construction has been shown to learn more accurately than others when there exist several complex attribute interactions. A new fitness function, based on the principle of Minimum Description Length (MDL), is proposed and implemented as part of the MFE3/GA system. Since the individuals of the GA population are collections of new features constructed to change the representation of data, an MDL-based fitness considers not only the part of data left unexplained by the constructed features (errors), but also the complexity of the constructed features as a new representation (theory). An empirical study shows the advantage of the new fitness over other fitness not based on MDL, and both are compared to the performance baselines provided by relevant systems.

Key words: Machine learning, attribute interaction, feature construction, feature selection, genetic algorithms, MDL principle, Entropy

1 Introduction

When data is represented by primitive attributes, Feature Construction (FC) has an outstanding impact on Data Mining results [1]. Many feature construction techniques face serious difficulties to succeed when confronted with complex attribute interactions. *Interaction* exists among attributes when the relation between one attribute and the target concept is not constant for all values of the other attributes [2–4]. Interactions become *complex* when changing the value of one attribute does not only change the relation between another attribute and the target concept, but it yields an opposite relation.

Most FC methods perform a local search to find interacting attributes one by one. So, they face difficulties when confronted with complex high-order interaction [2]. Due to complex interaction, it is necessary to search the space of subsets of attributes. Since the search space of attribute subsets grows exponentially with the number of attributes and has high variation, a global search such as Genetic Algorithm (GA) [5] is preferred for a FC method. Recent works [6–10] show that a genetic-based FC is more likely to be successful in searching through intractable and complicated search space of interacting attributes.

There are several factors that are important in guiding a genetic-based search to converge to the optimal solution. Among them the fitness function has a major role. The fitness function intends to guide the GA toward its goal and accelerate its convergence by providing a good estimate of the quality of each individual in the population. When a GA is applied to perform FC, the goal is to generate new features that facilitate more accurate learning when they are used to change the representation of training data. Thus, the fitness function should estimate the quality of the constructed features.

Constructed features may be evaluated in different ways. Three common forms of evaluating features are MDL-based measure, Entropy-based measure, and classifier error rate measure. MDL fitness function measures the inconsistency and complexity of constructed features based on MDL (Minimum Description Length) principle [11, 12]. Entropy-based fitness measures amount of uncertainty produced using new features. The third fitness first redescribes data using constructed features and then applies a learner to classify data and measure its error rate. In this paper we concentrate on the first two forms of fitness measure. The third one is not appropriate for genetic-based search since it is computationally expensive. The fitness is evaluated for each individual in each generation; thus, a fitness function with less computational time is preferable.

Considering the importance of fitness function in GA, we modified the fitness function of MFE2/GA (a multi-feature extraction using GA) [10] to conform to MDL principle and called the new system MFE3/GA. The new fitness function is empirically compared to an Entropy-based fitness function. Also the new system is compared to the performance baselines provided by relevant systems.

2 MDL-based Fitness in MFE3/GA

MDL has been successfully integrated into several learning methods. The MDL principle was originally described in terms of optimizing a communication problem. In order to apply it to learning, the learning task has to be described as a communication problem. The learner has a table of pre-classified training data that needs to be sent to the receiver. As an alternative to sending the whole table, the learner can compress data into a "theory" (i.e., a decision tree, a set of rules or any other form of classifier) and send it to the receiver. Such theory may not be perfect, and hence make "errors" when classifying some of the training data. So, to make the communication correct, the errors should also be sent to the receiver along with the theory. This introduces a trade-off between a very simple theory that produces many errors and a more complex one that accounts for almost all data and makes only a few errors. The MDL principle establishes that the optimum solution is a theory that minimizes the sum of the code lengths corresponding to theory and errors. This criterion has been used, for instance, to control the growth of decision trees [13].

The integration of the MDL principle into the evolutionary approach is not as frequent as it is in other machine learning systems. Most GAs have focused on optimizing a fitness based on classification errors. When GA is used for FC and so

individuals represent new constructed features, MDL may become necessary. The proposed features correspond to a theory that can grow too large and complex to produce no errors in the training data, and that we may prefer to keep simpler as long as it does not produce too many errors. In spite of this, none of the genetic-based FC systems integrates MDL into their fitness function.

A partial exception is MFE2/GA. This method is a preprocessing method that receives original attributes and data, and uses GA to search the space of different sets of attribute subsets and functions defined over them. Its fitness function measures both the complexity of constructed features and their inconsistency with training data; however, it was not explicitly designed as approximation to the MDL principle. This section briefly describes MFE2/GA and introduces a modification to its fitness function to conform to MDL principle.

Each individual in MFE2/GA is designed to represent a set of attribute subsets. Each subset is represented by a bit-string of length N, where N is the number of original attributes; each bit showing the presence or absence of the attribute in the subset. Thus, each individual of k subsets is a bit-string of length $k.N$ ($k > 0$). Since each individual has different number of subsets, the length of individuals is variable. To avoid unnecessary growth of individuals, the number of subsets in each individual is limited to the up bound $K = 5$ by default.

Each attribute subset in individual is associated with a function defined over it and extracted from the data. Functions are represented by non-algebraic form [10]. For any given subset the corresponding function is defined by assigning Boolean class labels extracted from data, to all the tuples in the Cartesian product of attributes in the subset. Changing subsets in an individual implies changing the corresponding functions. GA aims to converge the population members toward the set of attribute subsets and their corresponding functions that best represent attribute interactions. When GA is terminated the constructed functions are added to the original attribute set and the new representation of data is given to a standard learner such as C4.5 [14] to proceed learning.

Before describing how the new fitness in MFE3/GA is computed, we shall introduce the notion of function length. Each function F_i, defined over subset S_i, is represented by Binary labels of tuples in Cartesian product of attributes in S_i. Thus, each F_i can be represented by $\prod_{j=1}^{m} |X_{i_j}|$ bits, which we refer to as *the length* of function, $len(F_i)$, where m is the number of attributes in S_i, and $|X_{i_j}|$ is the number of values that attribute X_{i_j} can take. Since all constructed functions are defined over proper subsets of S, the longest function F_l is one defined over $S_l = S - \{X_s\}$ where X_s is the attribute that can take fewest values. The length of F_l is $\prod_{i=1, i\neq s}^{N} |X_i|$. To reduce the complexity of constructing functions, the length of each function is limited by a parameter of the system, B. By default the limit is set to 2^B, $B = 16$, that is, 64 Kbits. In case of Binary attributes this is equivalent to a function defined over 16 attributes. So the longest function is of length $MAXLEN = min(\prod_{i=1, i\neq s}^{N} |X_i|, 2^B)$.

The fitness of each individual $Ind = \langle S_1, \ldots, S_k \rangle$ is determined by evaluating the set of corresponding functions $\{F_1, \ldots, F_k\}$ and measuring two factors: the inconsistency of the set with the training data and its complexity.

The inconsistency measure drives GA to generate more accurate functions. For measuring the inconsistency of the set of functions with training data, training data are projected onto the set of constructed features $\{F_1, \ldots, F_k\}$. Then, each tuple in the projection that matches with both positive and negative samples in data is considered as an inconsistent tuple. The inconsistency of the set of functions, $\|E\|$, is measured by the total number of samples that match with inconsistent tuples in the projection. To normalize this value we divide it by the maximum inconsistency, that is, the total number of samples in the training data, M.

The consistency of the individual is not the only factor to drive GA towards its goal. Recall the goal is to ease the complex relation among interacting attributes by constructing several functions each representing one complex interaction in the concept. To achieve this goal, the fitness function prefers a consistent individual with several small functions to a consistent individual with few large functions by measuring their complexities. The complexity of each individual is determined by the sum of length of functions defined over subsets in the individual. We normalize the complexity factor by dividing it by its maximum value that is $K \times MAXLEN$.

Then, the fitness of the individual is evaluated by the following formula and GA aims to minimize this value:

$$Fitness(Ind) = \frac{\|E\|}{M} + \frac{\sum_{i=1}^{k} len(F_i)}{K \times MAXLEN} \ . \tag{1}$$

Therefore, given two individuals equally consistent with the training data, the fitness function prefers the one with several functions defined over smaller subsets of attributes, rather than the one with few function defined over larger subsets. Note that the complexity evaluation corresponds to measure the length of functions and not length of individuals.

To compare this fitness function with other fitness functions, we also modified MFE2/GA to apply an Entropy-based fitness function and called it MFE2/GA$_E$. For each individual, the fitness is measured by calculating the Entropy of the concept given the values of new features [14, 15]. More precisely it is calculated as follows:

$$Fitness(Ind) = \sum_{i=1}^{2^k} \frac{|T_i|}{|T|} Entropy(T_i), \tag{2}$$

where T_i is set of training samples whose values for new attributes F_1 to F_k are equal to the i^{th} tuple in the Cartesian product $F_1 \times \ldots \times F_k$. To reduce overfitting, part of training data are used for constructing functions and all training data are used for Entropy-based fitness evaluation. Keeping part of data for fitness evaluation helps GA to construct individuals with smaller functions.

3 Experimental Results

This section empirically compares results obtained by two systems that use two different fitness functions: MFE3/GA with MDL-based fitness, and MFE2/GA$_E$

with Entropy-based fitness. We also compare them with two learners: the standard learner C4.5 (trees and rules), and HINT [16], a greedy-based feature construction method that similarly to MFE3/GA uses non-algebraic representation for constructed features. Part of these experiments uses synthetic concepts designed to focus the empirical study on situations were multiple complex attribute interactions make feature construction necessary for learning and difficult to achieve. We also report on similar experiments using real-world data from the Braille code domain.

3.1 Experiments with Synthetic Concepts

The synthetic concepts used as a benchmark for these experiments are composed by several complex interactions. For all concepts, attributes are Boolean except in the last 3 concepts, where there are 3-valued attributes. Table 1 gives a summary of these concepts. Columns 2 and 3 show the number of relevant and irrelevant attributes for each concept. The majority class percentage of each concept is shown in column 4. Note that for some concepts there are attributes participating in more than one underlying interaction (shared attributes). For example, in $\wedge(\mathtt{P}_{1,4}, \mathtt{P}_{3,6})$, x_3 and x_4 are shared by $\mathtt{P}_{1,4}$ and $\mathtt{P}_{3,6}$. See Appendix for a detailed definition of concepts, including a description of the complex interactions underlying these concepts.

All experiments were run 20 times independently, each using 5% of all possible instances as training data and the rest as test data. For MFE2/GA$_E$, we used only part of the 5% training data for constructing features and all training data for fitness evaluation using Entropy. Our previous experimental evaluation showed that on average, MFE2/GA$_E$ achieves higher accuracy when 30% of training data are used for feature construction. So we used 30% of training data for feature construction and all 5% training data for feature evaluation. Note that by doing this we tried to benefit MFE2/GA$_E$ and yet we believed the MDL-based MFE3/GA could out perform it.

Table 1 illustrates a summary of the empirical study. The higher of the two average accuracies obtained by C4.5 and C4.5-Rules is reported in column 5. This result is marked by c if obtained by C4.5, or by r if obtained by C4.5-Rules. The average accuracies of HINT, MFE2/GA$_E$, and MFE3/GA are reported in columns 6 to 8 respectively. Columns 9 and 10 show the average number of GA's generations for each genetic-based method. Numbers between parentheses indicate standard deviation. The highest average accuracy is marked by $\triangleleft$, but if it is not lower than the majority class percentage. The accuracy of MFE2/GA$_E$ is marked by † when it is significantly better than the accuracy of HINT. MFE3/GA's result is significantly better than those in bold and significantly worse than those in italic (t-distribution test with $\alpha = 0.02$).

As it can be seen from Table 1, the MDL-based fitness function of MFE3/GA guides this method towards better solutions as expected; and therefore, it significantly outperforms MFE2/GA$_E$ for most concepts. MFE2/GA$_E$ in most cases overfits data. It constructs set of features with very small Entropy (most of the time with zero Entropy) which means the set of features classify 5% training

Table 1. Average accuracy and number of generations for synthetic concepts

			M	Average accuracies				Avrg. No generations	
Concept	R	I	%	C4.5/R	HINT	MFE2/GA$_E$	MFE3/GA	MFE2/GA$_E$	MFE3/GA
$\wedge(P_{1,4}, P_{3,6})$	6	6	75	c**72.5(3.2)**	100(0.0)◁	98.3(2.1)	99.8(0.5)	137(35.7)	125(18.4)
$\wedge(P_{1,6}, P_{3,8})$	8	4	75	c**73.4(2.7)**	*98.6(6.3)*◁	91.8(6.5)	94.1(2.8)	**219(47.0)**	131(18.1)
$\wedge(P_{1,6}, P_{7,12})$	12	0	75	c**72.6(3.9)**	82.5(16.5)	**77.1(6.0)**	89.8(6.8)◁	**230(82.4)**	144(16.0)
$\wedge(P_{1,3}, P_{3,5}, P_{4,6})$	6	6	88	c**87.6(1.2)**	**94.1(9.6)**	**96.7(4.9)**	99.8(0.7)◁	130(24.0)	141(27.6)
$\wedge(P_{1,4}, P_{2,5}, P_{3,6})$	6	6	88	c**87.5(0.3)**	97.1(7.2)	**96.5(4.6)**	99.6(0.7)◁	153(46.9)	130(29.6)
$\wedge(P_{1,4}, P_{3,6}, P_{5,8})$	8	4	88	c**87.5(0.1)**	**90.3(11.0)**	**91.7(5.4)**	98.6(1.7)◁	207(54.0)	173(43.7)
$\wedge(P_{1,4}, P_{5,8}, P_{9,12})$	12	0	88	c**87.5(0.1)**	**78.4(4.1)**	**86.4(4.4)**†	92.4(7.2)◁	212(63.0)	199(53.9)
$\wedge(P_{1,6}, P_{2,7}, P_{3,8})$	8	4	88	c**86.6(1.8)**	92.3(10.0)	**86.7(4.1)**	93.8(2.4)◁	174(43.4)	169(40.9)
$\wedge(WL3_{1,5}, WL3_{3,7})$	7	5	64	r90.1(3.0)	91.2(11.6)	90.9(3.8)	93.1(5.9)◁	**201(65.7)**	132(28.8)
$\wedge(WL3_{1,5}, WL3_{4,8})$	8	4	68	r86.7(2.0)	88.8(8.9)	89.2(6.1)	89.9(9.6)◁	**230(72.9)**	156(51.8)
$\wedge(WL3_{1,5}, WL3_{5,9})$	9	3	72	r**84.9(2.6)**	87.9(10.1)	88.5(6.2)	93.5(7.0)◁	**213(55.4)**	154(37.6)
$\wedge(WL3_{1,5}, WL3_{6,10})$	10	2	75	r**82.2(2.1)**	**78.6(5.2)**	83.3(3.5)†	88.1(8.4)◁	**233(80.5)**	167(43.4)
$\wedge(WL3_{1,4}, WL3_{3,6}, WL3_{5,8})$	8	4	58	r**89.2(4.1)**	**89.3(12.0)**	**92.9(5.7)**	97.5(2.2)◁	**208(60.2)**	162(50.2)
$\wedge(WL3_{1,4}, WL3_{5,8}, WL3_{9,12})$	12	0	68	r**79.5(3.2)**	**71.8(4.5)**	**81.1(6.5)**†	92.3(10.5)◁	**239(60.9)**	177(49.1)
$\wedge(W23_{1,6}, W23_{7,12})$	12	0	71	r**68.2(2.3)**	**65.9(3.3)**	**72.8(3.1)**†	83.4(9.3)◁	**215(65.5)**	159(40.1)
$\wedge(W23_{1,4}, W23_{5,8}, W23_{9,12})$	12	0	76	r**74.7(1.9)**	**69.7(3.0)**	**80.4(4.3)**†	94.1(9.4)◁	250(75.0)	207(67.3)
$\wedge(W23_{1,5}, W23_{6,10}, W23_{11,15})$	15	0	76	r**88.5(3.1)**	98.9(2.8)	**98.5(2.5)**	100(0.0)◁	**228(66.6)**	187(24.2)
$\wedge(W23_{1,6}, W23_{7,12}, W23_{13,18})$	18	0	84	r**98.1(0.9)**	100(0.0)◁	**99.5(0.5)**	100(0.0)◁	215(57.0)	200(24.9)
$\wedge(A_{1,4}, A_{5,8}, A_{9,12})$	12	0	82	r**89.8(5.0)**	**79.7(3.1)**	**89.1(4.0)**†	97.8(4.3)◁	243(77.0)	225(69.1)
$\wedge(B_{1,4}, B_{5,8}, B_{9,12})$	12	0	88	c**86.9(1.3)**	**81.1(2.1)**	**88.0(1.3)**†	89.6(4.0)◁	231(68.9)	190(70.3)
$\wedge(C_{1,4}, C_{5,8}, C_{9,12})$	12	0	58	r**66.2(3.8)**	**64.6(7.8)**	**84.6(16.0)**†	98.5(6.9)◁	**254(75.4)**	170(24.0)
$\wedge(D_{1,4}, D_{5,8}, D_{9,12})$	12	0	88	r90.6(2.8)	**83.7(1.9)**	**89.7(1.4)**†	92.3(3.4)◁	217(77.6)	194(45.2)
$\wedge(E_{1,4}, E_{5,8}, E_{9,12})$	12	0	76	r**77.0(3.0)**	**72.2(4.8)**	**81.4(6.8)**†	93.0(10.5)◁	232(62.1)	200(65.7)
$\wedge(A_{1,4}, C_{5,8}, E_{9,12})$	12	0	74	r**82.2(3.4)**	**73.7(5.6)**	**84.2(7.4)**†	97.5(6.1)◁	232(77.6)	197(50.8)
$\wedge(A_{1,4}, B_{5,8}, D_{9,12})$	12	0	86	r**87.6(3.6)**	**81.5(3.4)**	**88.7(2.7)**†	92.0(4.7)◁	209(65.2)	206(54.3)
$\wedge(A_{1,4}, B_{5,8}, C_{9,12})$	12	0	79	r**86.3(3.5)**	**75.8(4.3)**	**87.2(4.2)**†	94.6(7.2)◁	248(57.1)	209(71.8)
$\wedge(B_{1,4}, C_{3,6}, A_{7,10}, D_{9,12})$	12	0	87	r88.5(2.0)	**83.2(2.8)**	**88.6(1.3)**†	90.8(3.6)◁	195(41.7)	199(48.5)
$\wedge(A_{1,4}, B_{5,8}, C_{9,12}, E_{13,16})$	16	0	87	r**94.8(2.1)**	99.8(1.0)	**99.2(1.2)**	100(0.0)◁	214(64.7)	235(35.1)
$\wedge(C_{1,4}, WL3_{5,8}, W23_{9,12})$	12	0	68	r**74.2(3.1)**	**70.6(7.2)**	**80.7(7.3)**†	93.7(11.1)◁	**231(68.2)**	178(47.4)
$\wedge(W23_{1,5}, C_{5,8}, WL3_{8,12})$	12	0	77	c**76.4(1.2)**	**71.2(2.4)**	**78.1(2.6)**†	84.0(8.7)◁	**219(71.8)**	169(43.1)
$\wedge(W23_{1,5}, C_{4,7}, WL3_{6,10})$	10	2	77	r**77.4(2.8)**	**75.9(6.6)**	**80.5(3.4)**†	88.7(8.9)◁	232(53.1)	193(55.0)
$\vee(pal_{1,4}, pal_{3,6}, pal_{5,8})$	8	0	70	r71.2(2.6)	**63.8(4.1)**	70.0(3.6)†	71.4(1.7)◁	**213(55.2)**	138(28.1)
$\vee(pal_{1,4}, pal_{4,7}, pal_{7,10})$	10	0	70	r**97.5(2.3)**	100(0.0)◁	**95.7(5.4)**	100(0.0)◁	**228(70.1)**	149(13.0)
$palindrome_6 + 2$	6	2	96	c**96.3(0.1)**	**93.2(2.0)**	**97.6(1.8)**†	99.6(0.7)◁	162(60.7)	133(19.4)
AVERAGE				r82.7	83.7	87.8	93.6◁	213.3	173.3

data perfectly. But when they are evaluated on test data, they produce errors. This is because Entropy does not consider the complexity of the theory proposed by the constructed features. It constructs large functions that perfectly match training data and produce overfitting.

Also comparing the average number of generations of both GA methods illustrates that MDL-based fitness function helps GA to converge to optimal solution faster than the Entropy-based method.

Comparing the results of MFE2/GA$_E$ and HINT indicates that, although Entropy-based FC achieves lower accuracy than MDL-based FC, its overall average accuracy is still better than HINT. This shows the advantage of using GA for FC when concepts are composed by several complex interactions and few training data are available. Even a genetic-based FC method with not the best fitness function outperforms the greedy-based FC. Note that the overall average

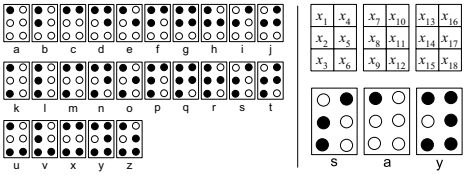

Fig. 1. Braille code representation and a sample of valid code

accuracy of HINT is only slightly higher than the standard learner C4.5/Rules for this type of concepts.

3.2 Experiments with Real-world Data

This section reports on a similar empirical comparison, but this time based on a task defined over a real-world domain. A Braille code is a 3×2 matrix of raised/unraised dots. The target concept is to distinguish Braille-coded text from randomly generated codes, using a windowing of 3 codes. Each sample consists of 3 codes, and each code is represented by 6 binary attributes, giving a total of 18 attributes. If all 3 codes are Braille, the sample is classified as true, and otherwise, it is classified as false. Figure 1 shows the Braille code as it was originally invented for French alphabet (which did not include the w), where raised and unraised dots are shown by black and white circles respectively.

A total of 20 data sets of 31250 samples were generated with majority class of 50%. Experimental results showed that the tree generated by C4.5 using features constructed by MFE3/GA has these features near the root, but still uses many primitive attributes at deeper levels. This indicates that the features generated were not enough for abstracting all interactions. So we increased the parameter K (see Section 2) from 5 to 9, allowing MFE3/GA to generate more features. This requires more CPU time, but a single learning trial still takes only a few minutes (for 5% data about 2 min. on a Pentium 4).

Experiments were performed increasing training data from 1% to 20% to see how data size affects methods. Table 2 shows accuracies of C4.5, C4.5Rules, HINT, MFE2/GA$_E$, and MFE3/GA. MFE3/GA's accuracy is significantly better than those in bold and worse than those in italic (t test, $\alpha = 0.02$).

Consider the results corresponding to 1% data in Table 2. For this size of training data, all FC methods achieve lower accuracies than C4.5 and C4.5Rules.

Table 2. Average accuracy over 20 runs for Braille-validation problem

Data Size	C4.5	C4.5 Rules	HINT	MFE2/ GA$_\mathbf{E}$	MFE3/ GA
1%	*90.7(1.9)*	*94.8(2.2)*◁	**75.9(4.1)**	**63.6(7.4)**	85.3(6.4)
5%	**97.6(0.4)**	**99.6(0.3)**	**90.2(3.4)**	**96.5(5.0)**	99.8(0.3)◁
10%	**98.6(0.3)**	**99.9(0.2)**	**95.9(3.4)**	**98.2(3.0)**	100.0(0.1)◁
15%	**99.0(0.1)**	**99.9(0.1)**	**99.0(0.8)**	**97.1(4.9)**	100.0(0.0)◁
20%	**99.4(0.1)**	100.0(0.0)	**99.4(0.6)**	**99.4(1.4)**	100.0(0.0)◁

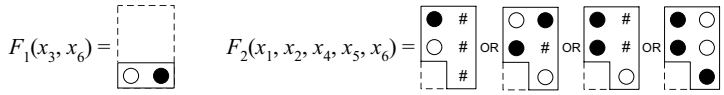

Fig. 2. Features constructed by MFE3/GA for Braille code concept

MFE2/GA$_E$ gets the lowest accuracy comparing to other FC methods because this method uses only 30% of 1% training data for function generation and overfits data. MFE3/GA overfits data less than other FC methods due to its MDL-based fitness function.

Table 2 shows that when the number of training data increases all FC methods take the advantage of training data size and improve their accuracies. However, MFE3/GA is the only FC method in the table that gets higher accuracy than C4.5 and C4.5Rules. It significantly outperforms all other methods except for 20% data when both MFE3/GA and C4.5Rules get 100 percent accuracy. The results of MFE2/GA$_E$ with 15% and 20% training data size show that when more data are provided, this method overfits data and achieves lower accuracy.

Note that C4.5-Rules generates a large number of rules (often more than 35 for 5% data) that are difficult to interpret. Features generated by MFE3/GA can be easily interpreted. For all experiments, MFE3/GA successfully discovers that there are three relations of 6 attributes each in the training data, and constructs functions to highlight these three relations. Each relation corresponds to one position in the 3-code window. MFE3/GA usually constructs two functions for each relation of 6 attributes, representing the definition of a Braille code, in total six functions for a sequence of three Braille codes. Figure 2 shows the two functions that are usually constructed to define the valid codes represented by the first 6 attributes. Similar functions are found for the other groups of six attributes. The solid line in the figure shows the domain of each function. A black circle indicates the attribute value is '1' (raised dot), a white circle means the attribute value is '0' (unraised dot), and a '#' means "don't care" (i.e., it can be either '0' or '1'). The first function, F_1, highlights all codes with unraised dot 3 and raised dot 6, as invalid codes, which need to be excluded from the target. The second function, F_2, is a disjunction of four rules to define all Braille letters ignoring dot 3. The conjunction, $\overline{F}_1 \wedge F_2$, classifies all Braille codes. When more data is available, MFE3/GA encapsulates the relation among 6 attributes and represents it by a single function. Thus, it constructs a total of just three functions, one function for each subset of 6 attributes, to represent a sequence of three Braille codes.

Also note that, in spite of using non-algebraic representation similarly to MFE3/GA, HINT needs more data to uncover the underlying concept structure and improve accuracy. This is probably due to MFE3/GA's use of GA-based search and evaluation of multiple candidate features simultaneously. Several interactions exist among 18 attributes in this concept. HINT needs to construct a complex hierarchy of functions representing interactions, which is a difficult task for its greedy procedure.

4 Conclusion

The accuracy advantage of the MFE2/GA approach was related to the structure of the individuals in the GA population. Each individual provides a collection of new features intended to change the representation of data, in a way that highlights underlying complex attribute interactions and, hence, simplifies learning. Due to this meaning of the genetically evolved individuals, we proposed the use of the MDL principle for evaluating the fitness of each individual. The new MDL-based fitness implemented in the MFE3/GA method includes two terms: one that approximates the complexity of the collection of new features (theory), and a second one that accounts for the misclassifications produced by those features (errors). To assess the advantage introduced by this new fitness, we performed an empirical study using a benchmark of synthetic concepts designed to involve several combinations of complex attribute interactions.

The study shows that the proposed MDL-based fitness yields significantly better predictive learning accuracy than other fitness solely based on Entropy. In addition, our empirical results show that even without the improvement of an MDL-based fitness, the MFE2/GA$_E$ approach with an Entropy-based fitness measure retains most of its accuracy advantage over two relevant learners: a standard learner as C4.5 (trees and rules), and HINT, a non-GA feature construction methods that, like MFE3/GA, uses non-algebraic representation for constructed features. Finally, similar empirical results were found using real-world data from the Braille Code domain.

Acknowledgment. Work has been partially supported by the Spanish Ministry of Science and Technology, under Grant number TSI2005-08225-C07-06.

References

1. Liu, H., Motoda, H.: Feature Extraction, Construction and Selection: A Data Mining Perspective. Volume 453 of The International Series in Engineering and Computer Science. Kluwer Academic Publishers, Norwell, MA, USA (1998)
2. Freitas, A.A.: Understanding the crucial role of attribute interaction in data mining. AI Review **16**(3) (Nov. 2001) 177–199
3. Jakulin, A., Bratko, I.: Testing the significance of attribute interactions. In Brodley, C.E., ed.: Proc. of the Twenty-first International Conference on Machine Learning, New York, USA, ACM Press (2004) 409–416
4. Pérez, E., Rendell, L.A.: Using multidimensional projection to find relations. In: Proc. of the Twelfth International Conference on Machine Learning, Tahoe City, California, Morgan Kaufmann (July 1995) 447–455
5. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag New York, Inc. (1999)
6. Larsen, O., Freitas, A.A., Nievola, J.C.: Constructing X-of-N attributes with a genetic algorithm. In: Proc. of the GECCO, San Francisco, Morgan Kaufmann (July 2002) 1268
7. Muharram, M., Smith, G.D.: Evolutionary constructive induction. IEEE Transactions on Knowledge and Data Engineering **17**(11) (2005) 1518–1528

8. Otero, F., Silva, M., Freitas, A., Nievola, J.: Genetic programming for attribute construction in data mining. In: Proc. of the Sixth European Conference in Genetic Programming. Volume 2610., Springer-Verlag (Apr. 2003) 384–393
9. Ritthoff, O., Klinkenberg, R., Fischer, S., Mierswa, I.: A hybrid approach to feature selection and generation using an evolutionary algorithm. In: UK Workshop on Computational Intelligence. (Sep. 2002)
10. Shafti, L.S., Pérez, E.: Reducing complex attribute interaction through non-algebraic feature construction. In: Proc. of the IASTED International Conference on AIA, Innsbruck, Austria, Acta Press (Feb. 2007) 359–365
11. Grunwald, P.D.: The Minimum Description Length Principle. Mit Press (2007)
12. Rissanen, J.: A universal prior for integers and estimation by minimum description length. The Annals of Statistics **11**(2) (Jun. 1983) 416–431
13. Quinlan, J.R., Rivest, R.L.: Inferring decision trees using the minimum description length principle. Inf. Comput. **80**(3) (1989) 227–248
14. Quinlan, R.J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, California (1993)
15. Shannon, C.E.: A mathematical theory of communication. Bell System Tech. Journal **27** (July 1948) 379–423 and 623–656
16. Zupan, B., Bohanec, M., Bratko, I., Demsar, J.: Learning by discovering concept hierarchies. Artificial Intelligence **109**(1-2) (1999) 211–242

Appendix: Concept Definitions

All concepts in Section 3.1 are defined over Boolean attributes except the last 3 concepts in Table 1, where attributes are 3-valued. The concept $palindrome_6$+2 is palindrome of 6 attributes with 2 additional irrelevant attributes. The other concepts are defined as conjunctions $\wedge(f_1, \ldots, f_n)$ or disjunctions $\vee(f_1, \ldots, f_n)$. Let $w(x_{i..j}) \stackrel{\text{def}}{=} \textit{weight of attributes } x_i \textit{ to } x_j$. Then f_m is one of the followings:

- $\mathrm{P}_{i,j} \stackrel{\text{def}}{=} \textit{parity}(x_i, \ldots, x_j)$
- $\mathrm{WL3}_{i,j} \stackrel{\text{def}}{=} w(x_{i..j}) < 3$
- $\mathrm{W23}_{i,j} \stackrel{\text{def}}{=} w(x_{i..j}) \in \{2, 3\}$
- $\mathrm{pal}_{i,j} \stackrel{\text{def}}{=}$ palindrome of x_i to x_j
- Any of functions $A_{i,j}$ $B_{i,j}$, $C_{i,j}$, $D_{i,j}$, and $E_{i,j}$, defined over 4 Boolean attributes x_i to x_j as explained below

Functions A, B and E consider their 4 attributes as a 2-by-2 bitmap and are true if and only if the bitmap contains the following patterns: function A detects if any two (vertically or horizontally) adjacent bits are set to 1; function B is as A but excluding the case of all bits set to 1; and function E is as A but including the case of all bits set to 0. Functions C and D consider their 4 attributes as a 4-by-1 bitmap (or just a sequence) and are true if and only if the bitmap contains the following patterns: function C detects if any two adjacent bits are set to identical values but not all bits have the same value; and function D detects if there are any two adjacent bits set to 1.

To illustrate the complexity of concepts used, note for instance that the DNF of function $A_{1,4}$ is $x_1x_2 + x_2x_3 + x_3x_4 + x_4x_1$, and some concepts of Table 1 are conjunction of $A_{1,4}$, $A_{5,8}$ and, $A_{9,12}$, or other three such concepts from the above functions.