Machine Learning by Multi-feature Extraction Using Genetic Algorithms[1]

Leila S. Shafti and Eduardo Pérez

Universidad Autónoma de Madrid, E–28049 Madrid, Spain
leila.shafti@ii.uam.es, eduardo.perez@ii.uam.es
WWW home page: http://www.ii.uam.es/

Abstract. Constructive Induction methods aim to solve the problem of learning hard concepts despite complex interaction in data. We propose a new Constructive Induction method based on Genetic Algorithms with a non-algebraic representation of features. The advantage of our method to some other similar methods is that it constructs and evaluates a combination of features. Evaluating constructed features together, instead of considering them one by one, is essential when number of interacting attributes is high and there are more than one interaction in concept. Our experiments show the effectiveness of this method to learn such concepts.

1 Introduction

Learning algorithms based on similarity assume that cases belonging to the same class are located close to each other in the instance space defined by original attributes. So, these methods attain high accuracy on domains where the data representation is good enough to maintain the closeness of instances of the same class, such as those provided in Irvine databases [1]. But for hard concepts with complex interaction, each class is scattered through the space due to low-level representation [2, 3]. *Interaction* means the relation between one attribute and the target concept depends on another attribute. When the dependency is not constant for all values of the other attribute, the interaction is *complex* [4]. Figure 1 shows an example of interaction and complex interaction between two attributes in instance space. The complex interaction has been seen in real-world domains such as protein secondary structure [5].

Constructive induction (CI) methods have been introduced to ease the attribute interaction problem. Their goal is to automatically transform the original representation space of hard concepts into a new one where the regularity is more apparent [6, 7]. This goal is achieved by constructing *new* features from the given attribute set to abstract the interaction among several attributes into a new one.

Most CI methods apply a greedy search to find new features. But because of the attributes' interaction, the search space for constructing new features has more variation. Recent works on problems with interaction [8, 3] show that a

[1] This work has been partially supported by the Spanish Interdepartmental Commission for Science and Technology (CICYT), under Grant number TIC2002-1948.

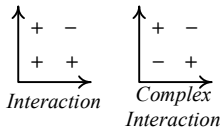

Fig. 1. Interaction and complex interaction in instance space

global search strategy such as Genetic Algorithms (GA) [9] is more likely to be successful in searching through the intractable and complicated search space [10].

Another limitation of many CI methods is the language they apply for representing constructed features. These methods use algebraic form of representing features. By *algebraic* form we mean features are shown by means of some algebraic operators such as arithmetic or Boolean operators. By contrast to this kind of representation, we have *non-algebraic* form of representation, which means no operator is used for representing the feature. For example, for a Boolean attribute set $\{X_1, X_2\}$ an algebraic feature like $(X_1 \wedge \overline{X}_2) \vee (\overline{X}_1 \wedge X_2)$ can be represented by a non-algebraic feature such as $\langle 0110 \rangle$, where the j^{th} element in $\langle 0110 \rangle$ represents the outcome of the function for j^{th} combination of attributes X_1 and X_2 acording to the truth table. This form of representation has been used in [11] and [12] for learning. We showed in [8] that a complex algebraic expression is required to capture and encapsulate the interaction into a feature, while non-algebraic representation reduces the difficulty of constructing complex features. For this reason, and in spite of their use of GA search strategy, some CI methods such as GCI [13], GPCI [14], Gabret [15] and the hybrid method of Ritthoff et al [16] fail when a high-order complex interaction exists among attributes.

There are very few methods that use non-algebraic form of representation; among them are MRP [17] and DCI [18]. These CI methods produced high accuracy on complex concepts with high interaction. However they have limitations and deficiencies. MRP represents features by sets of tuples obtained from training data using relational projection, and applies a greedy search in order to construct features. Each constructed feature is used as a condition to split data in two parts and then for each part, new features are generated. Therefore, only one feature is constructed and evaluated at a time. DCI applies GA to reduce the problem of local optima. However, it still evaluates features one at a time and is incapable of constructing more than one feature. Therefore, both methods construct and evaluate new features individually.

When the number of interacting attributes is high, the feature that encapsulates the interaction is complex and difficult to be constructed. A CI method must break down the complex interaction into several smaller features. Such set of features works as a theory of intermediate concepts that bridge the gap from the input data representation to the hard target concept. In that case each feature that partially shows the interaction, by itself, does not give enough information about concept and may be considered as an irrelevant feature. In order to see the goodness of new features, the CI method should evaluate the combi-

nation of features together as related parts of the theory. For this reason MRP, DCI, and other similar methods fail when number of interacting attributes grows and the interaction is complex (see Sect. 3).

In this paper, we will introduce MFE/GA, a Multi-feature Extraction method based on GA and non-algebraic form of representation, for constructing a set of new features to highlight the interaction that exists among attributes. Our experiments show the advantage of our method over other CI methods when number of interacting attributes grows, and the interaction is complex.

2 GA Design

MFE/GA uses GA to generate and combine features defined over different subsets of original attributes. The current version of the method only works with nominal attributes. Therefore, any continuous attribute should be converted to nominal attributes before running MFE/GA. The GA receives training data represented by original attributes set and finds subsets of interacting attributes and features representing the interaction. When the GA finishes, the new features are added to the attributes set and the new representation of data is given to a standard learner for learning. This section explains the system design.

2.1 Individuals Representation

For constructing new features we need to perform two tasks: finding the subset of interacting attributes and generating a function defined over each subset. Our individuals are sets of subsets of primitive attributes such as $Ind = \langle S_1, S_2, \ldots, S_k \rangle$ where $S_i \subset S$, $S_i \neq \emptyset$, and S is the set of original attributes.

Subsets in individuals are represented by bit-strings of length n, where n is the number of original attributes; each bit showing the presence or absence of the attribute in the subset. Therefore each individual is a bit-string of length $k.n$ $(k > 0)$ such as $Ind = \langle b_1, \ldots, b_n : b'_1, \ldots, b'_n : b''_1, \ldots, b''_n : \ldots \rangle$.

Since each individual has different number of subsets, the length of individuals is variable. To avoid unnecessary growth of individuals we have limited the number of subsets in individuals so that $k \leq 5$.

Each subset in individual is associated with a function that is extracted from data. Thus each individual is actually representing a set of functions such as $\{F_1, F_2, \ldots, F_k\}$, where F_i is a function defined over S_i. It is important to note that during mutation and crossover if a subset is changed, the associated function is also changed since a new F'_i is extracted for the new subset S'_i in the offspring.

The function F_i created for any given subset $S_i = \{X_{i1}, \ldots, X_{im}\}$ in an individual uses a non-algebraic form of representation (see Sect. 1). As explained in [8], this form of representation reduces the difficulty of constructing complex features. Each F_i is defined by assigning Boolean class labeles to all the tuples in the Cartesian product $X_{i1} \times \ldots \times X_{im}$. The class assigned to each tuple t depends on the training samples that *match* the tuple, that is, the training samples whose values for attributes in S_i are equal to the corresponding values in tuple t.

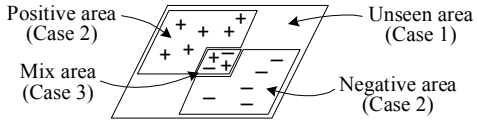

Fig. 2. Space of samples defined by attributes in subset S_i

More precisely, the class assigned depends on the class labels of all those training samples matching the tuple, as discussed next case by case:

Case 1. If there are no training samples matching t, a class label is assigned to $F_i(t)$ stochastically, according the class distribution in the training data.
Case 2. If all training samples matching t belong to the same class, this is the class assigned to $F_i(t)$.
Case 3. If there is a mixture of classes in the samples matching t, the class assigned to $F_i(t)$ depends on the numbers of tuples labelled by Case 2 as positive and negative, p_2 and n_2 respectively. In particular, if $p_2 = n_2 = 0$, the class is assigned stochastically (as in Case 1); if $p_2 > n_2$, the negative class is assigned; and otherwise, the positive class is assigned.

This procedure for extracting the definition of F_i from data partitions the subspace defined by S_i into four areas, as illustrated in Fig. 2. Each F_i identifies similar patterns (Case 2) of interaction among S_i and *compresses* them into the negative or postive area. The unseen area (Case 1) is covered by stochastically predicting the most frequent class. Note this covering of the unseen area means *generalization*, and thus may involve prediction errors.

As we will see next, the GA's fitness evaluation is applied to individuals composed of several F_i, each defined over a subset S_i. Thus, each individual encapsulates several interactions into features, and this allows the GA to simultaneously construct and evaluate features, which turns out to be essential when several high-order interactions exist in data.

2.2 Fitness Function

After feature extraction, for each $Ind = \langle S_1, \ldots, S_k \rangle$, data are projected onto the set of new features $\{F_1, \ldots, F_k\}$ and the goodness of the individual is evaluated by the following formula:

$$Fitness(Ind) = \frac{min(|\pi^+ - \pi^-|, |\pi^- - \pi^+|)k + \|\pi^+ \cap \pi^-\|(k+1)}{r(k+1)} + \frac{\sum |S_i|}{k|S|} \qquad (1)$$

where π^+ is set of positive tuples and π^- is set of negative tuples obtained by projecting data into $\{F_1, \ldots, F_k\}$, r is the total number of tuples in training data, the single bars $|z|$ denote the number of attributes (or tuples) in subset (or relation) z, and the double bars $\|p\|$ denotes the number of examples in training data that match with the tuples in relation p. The objective of GA is to minimize the value of $Fitness(Ind)$. The first term in this formula estimates how good is the

set of newfeatures for classifying data. It is divided by $k+1$ to favor individuals with larger number of subsets. The aim is to prefer several simple features to few complex features. The complexity of features is evaluated in the last term by measuring the fraction of attributes participating in constructing features.

To reduce overfitting, we use 90% of training data for generating functions and all training data for fitness evaluation. Aside from that, the empirical evaluation of the system will be based on unseen data (see Sect. 3).

2.3 GA Operators

The genetic operators have the role of converging GA to optimal solution. We use mutation and crossover operators. Our objective is to generate different subsets of attributes with their associated functions and combine them to eventually find the group of functions defined over subsets of interacting attributes. To achieve this objective, we apply operators in two levels: attributes level to generate different subsets and features; and, subsets level to make different combination of subsets and features. Therefore we have two types of mutation and two types of crossover, illustrated in Fig. 3 where colons are used to seperate subsets of attributes, bars mark the crossover points, and underlined are genes in the parents that will be substituted by the operator.

Mutation Type-1–Mutation in Attributes Level: This operator is performed by considering the individual as a bit-string of size $k.n$ where k is the number of subsets and n is the number of original attributes. The traditional mutation is applied over the bit-string to flip bits of the string. By flipping a bit, we eliminate/add an attribute from/to any subset in any given individual.

Mutation Type-2–Mutation in Subsets Level: For this operator an individual is considered as a sequence of subsets and mutation is performed over any subset as whole, by replacing the subset with a new generated subset. Therefore, after this operation, some subsets are eliminated from the given individual and new subsets are added to produce a new combination of subsets.

Crossover Type-1–Crossover in Attributes Level: This operator applies classical two-point crossover considering the individual as a bit-string. Its aim is to generate new subsets by recombining segments of subsets from the parents. The two crossing points in the first parent are selected randomly. On the second parent, the crossing points are selected randomly, subject to the

Mutation in Attributes Level	**Mutation in Subsets Level**
Parent1 = ⟨1**0**010**0**10:0**1**0**1**0100:000**1**0111⟩	Parent2 = $\langle S_1, \underline{S_2}, S_3, S_4, S_5 \rangle$
Child1 = ⟨11010011:01110100:00000111⟩	Child2 = $\langle S_1, \underline{S_2'}, S_3, S_4, S_5 \rangle$
Crossover in Attributes Level	**Crossover in Subsets Level**
Parent3 = ⟨1001|0010:010|10100:00010111⟩	Parent5 = $\langle S_{11}, \underline{S_{12}} \rangle$ Mask1= ⟨10⟩
Parent4 = ⟨0010|1011:10010001:111|01100⟩	Parent6 = $\langle S_{21}, \underline{S_{22}, S_{23}}, S_{24}, \underline{S_{25}} \rangle$ Mask2= ⟨01101⟩
Child3 = ⟨10011011:10010001:11110100:00010111⟩	Child5 = $\langle S_{11}, S_{22}, S_{23}, S_{25} \rangle$
Child4 = ⟨00100010:01001100⟩	Child6 = $\langle S_{12}, S_{21}, S_{24} \rangle$

Fig. 3. GA operators

restriction that they must have the same distance from the subsets boundary in bit-string representation as they had in the first parent [19]. Depending on where the crossing points are situated this operator may generate new subsets from subsets of parents and/or recombine subsets. This operator may change the length of the individual but we impose the limitation of $k \leq 5$; otherwise, new crossing points are selected until the produced offspring have $k \leq 5$ subsets.

Crossover Type-2–Crossover in Subsets Level: The aim of this operator is to generate different combinations of subsets by exchanging subsets of parents. It considers individuals as sequence of subsets and performs uniform crossover. Two crossover masks are generated randomly to define the cutting points. This operator may change the length of the individuals and therefore has the restriction of $k \leq 5$ same as above. Crossover Type-2 only recombines subsets and does not generate any new subset.

3 Empirical Analyses

We analyzed MFE/GA by empirically comparing it with other similar methods. For implementing GA we used PGAPack Library [20] with default parameters except those indicated in Table 1. The type of mutation and crossover operators (see Sect. 2.3) is specified by flipping a coin when individuals are selected for reproduction. We used 90% of training data for constructing functions and all training data for evaluating the constructed feature using (1).

To compare MFE/GA with other CI methods, we performed experiments over synthetic problems. Since our objective is to learn concepts with more than one interaction, artificial problems of this kind were selected for experiments. These problems were used as prototypes to exemplify complex interactions in real-world hard problems. Therefore, similar results are expected for real-world problems, where the main difficulty is complex interaction.

These concepts are defined over 12 Boolean attributes $a_1, \ldots, a_{12}$ as follows:
- $cp(i,j) = Parity(a_i, \ldots, a_6) \wedge Parity(a_7, \ldots, a_j)$
- $cdp(i,j) = Parity(a_i, \ldots, a_4) \wedge (Parity(a_{(i+j)/2}, \ldots, a_8) \vee Parity(a_j, \ldots, a_{12}))$
- $P(3,6) \wedge (l) = Parity(a_3, \ldots, a_6) \wedge$ (exactly l attributes in $\{a_7, \ldots, a_{12}\}$ are true)
- $P(3,6) \vee (l) = Parity(a_3, \ldots, a_6) \vee$ (exactly l attributes in $\{a_7, \ldots, a_{12}\}$ are true)

All above concepts have complex interaction among attributes that can be represented by several smaller interactions.

For each concept MFE/GA was run 20 times independently using 5% of shuffled data for training and the rest for final evaluation. When MFE/GA is finished, its performance was evaluated by the accuracy of C4.5 [21] on modified data after adding constructed features, using 95% unseen data as test data.

Table 1. GA's modified parameters

GA Parameter	New Value	GA Parameter	New Value
Population Size	100	Mutation Probability	0.01
Max Iteration	350	Num. of Strings to be Replaced	90
Max No Change Iteration	100		

Table 2. Summary of CI methods

CI Method	Algebraic Representation	Feature Evaluation	Genetic Search	CI Method	Algebraic Representation	Feature Evaluation	Genetic Search
Fringe	Yes	No Individually	No	MRP	No	Individually	No
Grove	Yes	Individually	No	DCI	No	Individually	Yes
Greedy3	Yes	Individually	No	MFE	No	No Individually	Yes
LFC	Yes	Individually	No	GA			

Table 3. System comparison by average accuracy

Concept	Num. of Relevant Attributes	Prior Best Result	MRP	DCI + C4.5	MFE/GA + C4.5
cp(4,9)	6	86.1 (Fringe)	99.0 (1.6)	97.2 (1.8)	98.9(4.0)
cp(3,10)	8	73.4 (C4.5-rules)	89.9 (1.2)	81.6 (1.6)	**95.7 (8.9)**
cp(2,11)	10	73.9 (C4.5)	91.7 (5.7)	68.0 (1.2)	**97.1 (4.8)**
cdp(3,11)	6	98.4 (Fringe)	97.3 (3.9)	97.6 (1.6)	99.2 (3.1)
cdp(2,10)	9	78.1 (Fringe)	92.0 (6.5)	67.7 (2.0)	86.6 (10.3)
cdp(1,9)	12	62.5 (C4.5-rules)	**81.3 (3.0)**	56.4 (2.5)	71.1 (7.4)
P(3,6)∧(2)	10	88.1 (C4.5)	90.4 (4.5)	81.0 (1.5)	**95.9 (2.9)**
P(3,6)∧(3)	10	83.4 (C4.5)	87.6 (5.4)	75.9 (2.4)	**94.1 (5.4)**
P(3,6)∧(3 or 2)	10	70.4 (Fringe)	79.0 (2.3)	65.4 (2.6)	**90.8 (5.6)**
P(3,6)∨(2)	10	70.5 (Fringe)	**97.1 (2.9)**	57.9 (2.2)	90.3 (6.6)
P(3,6)∨(3)	10	68.0 (Fringe)	95.9 (4.4)	59.3 (1.7)	92.1 (7.1)
P(3,6)∨(3 or 2)	10	75.1 (Fringe)	80.4 (6.0)	69.5 (2.2)	**92.5 (7.5)**

We compared MFE/GA with C4.5 and C4.5-Rules [21], which are similarity-based learners, Fringe, Grove and Greedy3 [22], and LFC [23], which are greedy-based CI methods that use algebraic representation of features and MRP [17] which is a greedy CI method with a non-algebraic form of representation (see Sect. 1). Among CI methods only Fringe constructs several features at once. Other methods consider features one at a time. We also compared MFE/GA with DCI [18], which applies GA to reduce the problem of local minima (see Sect. 1). This method uses a feature representation similar to MFE/GA. However, it constructs and evaluates one feature during each generation. Therefore, when an interaction among a large set of attributes exists in the concept, this method fails to learn the concept. Table 2 summarizes the CI methods that are used in our experiments.

Table 3 gives a summary of MFE/GA's average accuracy over 20 runs and its comparison with other systems' average accuracy. In the third column of the table, we show the best results among C4.5, C4.5-Rules, Fringe, Grove, Greedy3 and LFC, as reported in [17]. Numbers between parentheses indicates standard deviation. Bolds means with a significant level of 0.05, this accuracy is the best between MRP and MFE/GA.

In these experiments, LFC, Greedy3 and Grove never appear as the best competitor among CI methods with algebraic representation. This may be due to the fact that they evaluate each proposed feature individually. When more than one interaction exist among attributes, a feature that encapsulates a single interaction may not provide, by itself, enough information about the final target concept, and therefore, is evaluated as an irrelevant feature.

MRP gives better result than other greedy methods because of its non-algebraic form of representing features. When high interaction exists among

attributes, a more complex algebraic feature is needed to abstract the interaction. However, a non-algebraic representation may capture the structure of the interaction and abstract it more easily.

Experiments in [18] show that the genetic-based method, DCI, outperforms MRP when the concept consists of only one complex interaction over large number of attributes, e.g. $Parity(a_1, \ldots, a_8)$. But for concepts with more than one interaction, like those in Table 3, DCI cannot achieve good accuracy because it is incapable of constructing more than one feature. It finds a subset of interacting attributes and constructs one feature to encapsulate the interaction. But as the interaction is complex, the constructed feature is not good enough to outline all the interaction and, hence, MRP's greediness outperforms this method.

However, while the number of interacting attributes grows, the concept becomes more complex to be learned. Since MRP constructs and evaluates features separately with a greedy search, its performance decays. MFE/GA successfully breaks down the interaction over relevant attributes into two or more interactions over smaller subsets of attributes using a global search; and therefore, it gives better accuracy than other methods in most concepts of Table 3.

The synthetic concepts $cdp(i, j)$ illustrates well the different behaviors and advantages of MRP and MFE/GA. Each of the concepts $cdp(1, 9)$, $cdp(2, 10)$ and $cdp(3, 11)$ involves three parity relations combined by simple interactions (Conjunction and Disjunction of Parity). The three concepts differ in the degree of parity involved (4, 3, and 2, respectively) but perhaps more importantly, they also differ in the ratio of relevant attributes (12/12, 9/12, and 6/12, respectively). This is the reason why, obviously, all results in Table 3 indicate that $cdp(1, 9)$ is the most difficult concept to learn: it has no irrelevant attributes that when projected away allow the complex substructures of the concept to became apparent, when learning from only 5% of data.

This affects MFE/GA in a higher degree than it affects MRP, probably due to differences between both systems' biases. In particular, considering $cdp(1, 9)$, MRP's focus on learning one single best relation probably guides learning toward $Parity(a_1, \ldots, a_4)$. As stated above, the interactions that combine parity features in this concept are simple (conjunction and disjunction). So MRP easily finds its way toward that parity feature. Had it used only that single feature to classify unseen data, it would have obtained even higher accuracy than it does (up to 87%). Perhaps, due to overfitting, MRP's heuristic function does not allow the system to reach such theoretically best possible performance on this concept. However, MRP's bias gets it closer to the goal than MFE/GA.

MFE/GA's bias is, in some sense, opposite to MRP's. It focuses on learning multiple features at once to evaluate them in combination. This higher flexibility in searching a large and complex feature space makes the system more dependent on data quality (since features are extracted from training data). Thus, MFE/GA exploits better than MRP the redundancy in data for $cdp(3, 11)$. Since concepts $cdp(2, 10)$ and $cdp(3, 11)$ are defined, respectively, over 9 and 6 attributes of a total of 12 attributes, the 5% training data is likely to contain repeated parts of the concept structure, that become more apparent when projecting away the

irrelevant attributes. However, this does not make these concepts easy to learn by non-CI methods, due to the complexity of the interactions involved. MRP tries to learn these two concepts as a single relation each, whereas MFE/GA beats it by seeing the multiple features involved at once. On the other hand, for $cdp(1,9)$, all 12 attributes are relevant, and so there is little or no redundancy in the 5% training data. Therefore, MFE/GA's more flexible search gets trapped in a local minima, overfitting data, whereas MRP is favored by its strong bias for *one best* relation, which in this case does indeed exists and it is easy to find.

It is important to note that in all experiments MFE/GA generates close approximations to the sub-interactions; excluding experiments over $cdp(1,9)$, in more than 86% of experiments our method successfully finds the exact subsets of interacting attributes.

Our use of synthetic concepts allows us to analyze system behavior deeply before moving on to try to solve real-world problems with difficulties similar to those exemplified by these synthetic concepts.

4 Conclusion

This paper has presented MFE/GA, a new CI method for constructing a combination of features to highlight the relation among attributes when high complex interaction exists in the target concept. The method applies a non-algebraic form of representing features, which is more adequate for constructing features when interaction is complex.

The genetic approach provides the ability of constructing and evaluating several features at once. Most CI methods consider features individually. When the complex interaction is of higher order and need to be represented by more than one feature, each feature by itself may not give enough information about the interaction and therefore is evaluated as irrelevant. The new features should be considered in combination when evaluated, as if they build up a theory composed of intermediate concepts that bridge the gap between a primitive low level representation and a high level complex concept.

Our experiment shows the advantage of non-algebraic representation of features over algebraic representation. Also it shows that CI methods that consider features individually fail to learn concepts when number of interacting attributes grows. Our method with GA-based global search and non-algebraic representation successfully finds the combination of features that represent interaction and outperforms other CI methods when the interaction becomes more complex.

References

1. Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases, [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Department of Information and Computer Science, Irvine,CA (1998)
2. Rendell, L.A., Seshu, R.: Learning hard concepts through constructive induction: framework and rationale. Computational Intelligence, **6**:247–270 (Nov., 1990)

3. Freitas, A.: Understanding the crucial role of attribute interaction in data mining. Artificial Intelligence Review, **16(3)**:177–199 (Nov., 2001)
4. Pérez, E.: Learning despite complex interaction: an approach based on relational operators, PhD Thesis, University of Illinois, Urbana-Champaign (1997)
5. Qian, N., Sejnowski, T.J.: Predicting the secondary structure of globular proteins using neural network models. Molecular Biology, **202**:865–884 (Aug., 1988)
6. Dietterich, T.G., Michalski, R.S.: Inductive learning of structural description: evaluation criteria and comparative review of selected methods. Artificial Intelligence, **16(3)**:257–294 (Jul., 1981)
7. Aha, D.W.: Incremental constructive induction: an instance-based approach. In Proc. of the Eighth International Workshop on Machine Learning, pages 117–121, Evanston, Illinois (1991) Morgan Kaufmann
8. Shafti, L.S., Pérez, E.: Genetic approach to constructive induction based on non-algebraic feature representation, In Proc. of the Fifth International Symposium on Intelligent Data Analysis, pages 509–520, Berlin, Heidelberg (2003) LNCS
9. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan (1975)
10. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin, Heidelberg, New York (1999)
11. Samuel, A.: Some studies in machine learning using the game of checkers II: recent progress, IBM J. Res. Develope. **11**:601–617 (1967)
12. Zupan, B., Bratko, I., Bohanec, M., Demsar, J.: Function decomposition in machine learning, LNAI **2049**:71-101 (2001)
13. Bensusan, H. Kuscu I.: Constructive induction using genetic programming. In Proc. of ICML'96 Workshop of Evolutionary Computing and Machine Learning, Bari, Italy (1996) T. Fogarty and G. Venturini (Eds.)
14. Hu, Y.: A genetic programming approach to constructive induction. In Proc. of the Third Annual Genetic Programming Conference, pages 146-157, Madison, Wisconsin (1998) Morgan Kauffman
15. Vafaie, H., De Jong, K.: Feature space transformation using genetic algorithms. IEEE Intelligent Systems and Their Applications, **13(2)**:57–65 (Mar.–Apr., 1998)
16. Ritthoff, O., Klinkenberg, R., Fischer, S., Mierswa, I.: A hybrid approach to feature selection and generation using an evolutionary algorithm, In Proc. of UK Workshop on Computational Intelligence (Sep., 2002) The University of Birmingham
17. Pérez, E. Rendell, L.A.: Using multidimensional projection to find relations. In Proc. of the Twelfth International Conference on Machine Learning, pages 447–455, Tahoe City, California (Jul. 1995) Morgan Kaufmann
18. Shafti, L.S., Pérez, E.: Constructive induction using non-algebraic feature representation, In Proc. of the Third IASTED International Conference on Artificial Intelligence, pages 134–139, Benalmadena, Spain (2003) Acta Press
19. DeJong, K. A., Spears, W.M., Gordon, D.F.: Using Genetic Algorithms for Concept Learning. Machine Learning, **13**:161–139 (1993)
20. Levine, D.: Users guide to the PGAPack parallel genetic algorithm library. Technical Report ANL-95/18, Argonne National Laboratory (Jan. 1996)
21. Quinlan, R.J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, California (1993)
22. Pagallo G., Haussler, D.: Boolean feature discovery in empirical learning. Machine Learning, **5**:71–99 (1990)
23. Ragavan, H., Rendell, L.A.: Lookahead feature construction for learning hard concepts. In Proc. of the Tenth International Conference on Machine Learning, pages 252–259, Amherst, Massachusetts (1993) Morgan Kaufmann