Constructive Induction Using a Non-Greedy Strategy for Feature Selection

Arlindo L. Oliveira
Dept. of EECS
UC Berkeley
Berkeley CA 94720
aml@ic.berkeley.edu

Alberto Sangiovanni-Vincentelli
Dept. of EECS
UC Berkeley
Berkeley CA 94720
alberto@ic.berkeley.edu

Abstract

We present a method for feature construction and selection that finds a minimal set of conjunctive features that are appropriate to perform the classification task. For problems where this bias is appropriate, the method outperforms other constructive induction algorithms and is able to achieve higher classification accuracy. The application of the method in the search for minimal multi-level boolean expressions is presented and analyzed with the help of some examples.

1 Introduction

Concept learning algorithms that build concept descriptions from labeled examples work by extending the classification of known examples to other points in the input space in accordance with some specified bias. The ability to perform generalization is strongly dependent on the set of features used. For example, if one is trying to learn the *xor* concept, any generalization based on the Hamming distance between two instances will give the wrong results.

Chosing an appropriate set of auxiliary features to use when performing classification is a difficult problem, usually described as constructive induction. The approaches developed so far can be viewed as belonging to one of two classes. Analytical approaches [Drastal et Al. 1989, Utgoff 1986] generate new features by using a domain specific theory. The usefulness of the features derived depends on the amount of appropriate domain knowledge available. If there exists good domain knowledge this may be an effective method to generate an adequate set of auxiliary features. Empirical approaches, on the other hand, usually start by using the original features and recursively constructing more complex ones [Pagallo & Haussler 1990, Schilmer & Granger 1986].

Given a set of either primitive or composite features the problem of selecting a set of features that is adequate for the classification task at hand is still difficult. New features are usually selected from the available pool by a greedy method where a new feature is chosen at a time, like, for example in ID3 [Quinlan 1986] and AQ15 [Michalski et Al. 1986]. These greedy methods may fail to select important features in the cases where important information can only be obtained by looking simultaneously at a set of several features. On the other hand, feature selection can be done in a non-greedy way like, for instance, in [Almuallim & Dietterich 1990]. This method may, in principle, lead to a better selection but is computationally more expensive.

To illustrate this problem, consider again the *xor* problem. A greedy algorithm will never select (except by chance) a 2 bit xor as a valuable feature, since it provides, in general, no information about the desired class. However, in an N bit *xor* problem, the set of N/2 2 bit *xor* (i.e., $\{x_1 \oplus x_2, x_3 \oplus x_4, ...x_{N-1} \oplus x_N\}$) is a useful set of features, in the sense that the dimensionality of the problem is now reduced by a factor of 2. This set of features, however, cannot be found by a greedy algorithm that examines one potential new feature at a time.

Despite the limitations of greedy selection, efficient algorithms for constructive induction applicable to problems where the concepts accept a compact DNF (Disjoint Normal Form) representations have been proposed. For example, FRINGE [Pagallo & Haussler 1990] is able to select the appropriate conjunctions to be used as the features for a set of concepts that accept a compact DNF representation. A generalization of the FRINGE heuristics proposed in [Yang et Al. 1991] is able to build either conjunctive or disjunctive features that make it possible to learn concepts with a compact DNF or a compact CNF (Conjunctive Normal Form) representation. In [Wnek & Michalski 1991] similar results are obtained using the latest version of the AQ family. Methods based on the use of logic synthesis techniques to minimize two-level networks have also been used to efficiently derive the minimal DNF representation [Oliveira & Sangiovanni 1991] for this type of problems.

For many concepts of interest, neither a compact DNF nor a compact CNF exists, but they can be represented by a compact factored form. A factored form is defined recursively: 1) a literal is a factored form; 2) a sum of factored

forms is a factored form; 3) a product of factored forms is a factored form. For example, $((a \vee b) \wedge (c \vee d)) \vee (e \wedge f)$ is a factored form. Compact factored forms may lead to exponentially long DNF or CNF representations. Finding appropriate features when learning concepts with a compact factored form is more difficult than when the concepts accept compact DNF or CNF forms because no feature provides complete information about the value of the output. In the logic synthesis community, the problem of finding adequate intermediate features is known as the factorization problem [Brayton et Al. 1990].

In this paper we describe an algorithm for feature selection (or boolean factorization) that does not have the limitations that are present when a greedy feature selection is performed. The algorithm derives a minimal set of features that are sufficient to distinguish between all the positive and negative examples in the training set. A detailed analysis of a restriction of a similar bias was carried out in [Almuallim & Dietterich 1990]. In that work the authors study the properties of learning algorithms that implement the following bias: between two classification functions that are consistent with the training data, choose the one that depends on the smaller number of input features. They show that any probably-approximately (PAC) learning algorithm that implements this bias (the MIN-INPUT-FEATURES bias) has a sample complexity that depends only logarithmically in the number of irrelevant input features. The work described here is a generalization of that approach since the features in the minimal set do not not need to be restricted to input features but can be conjunctions of other features.

2 Definitions

We consider the problem of supervised concept learning in an attribute based description language. We assume the attributes are boolean and the domain is noise free.

Let the input examples be specified by n boolean variables $\{x_1, ..., x_n\}$ and let a literal be either a variable or its negation. The instance space, $\mathcal{U}$, is the n dimensional boolean space $\{0,1\}^n$. A concept $\mathcal{C}$ and a hypothesis $\mathcal{H}$ are subsets of $\mathcal{U}$. Elements of $\mathcal{U}$ are identified with n literal conjunctions in the obvious way. Instances (or examples) extracted from $\mathcal{U}$ according to some distribution are presented to the learner and labeled positive or negative, according to whether they are or are not in $\mathcal{C}$. These instances are the training set, $\mathcal{T}$. Let P be the set of positive examples in $\mathcal{T}$ and $\mathcal{N}$ and the set of negative ones.

The task of the learner is to output, from the set of possible hypothesis, one that closely approximates the target concept, $\mathcal{C}$. To do this, the learner is free to define auxiliary features. Features are boolean functions defined in terms of other existing features. A feature f_i is said to have a value 1 for one example if the values of the input variables defined for that example cause the boolean function associated with the feature to take the value 1. Otherwise, the feature is said to take the value 0. A conjunctive feature is one that is a conjunction of other features, while a disjunctive one is a disjunction.

We will commonly describe boolean functions using the $+$ sign to denote disjunction and the implicit multiplication sign to denote conjunction.

3 Approach description

Since the number of possible features is very large selecting an appropriate set of features is a difficult problem. In the absence of domain knowledge this problem is generaly solved by selecting the single most promising feature, according to some measure of utility. This approach may fail to identify useful features if classification information is correlated among multiple features.

We propose to select a set of features that, considered as a whole, maximizes some utility function. In this paper, we will consider only one specific utility function, the number of features. This means that we will select **a set of features of minimal cardinality that is sufficient to classify all the examples presented in the training set**. This particular utility function is appropriate for many problems where the MIN-FEATURE bias is adequate.

A formal argument that supports this assumption can be made by considering the VC-dimension[1] of all functions implemented over a set of k features. This family of functions has a VC-dimension of 2^k, and, since smaller VC-dimension means that one is able to learn using a smaller number of examples [Blumer et Al. 1986] we should usually be better off by using as few features as possible. This simplified analysis did not consider the contribution to the VC-dimension given by the ability of the classifier to choose the features. If arbitrary complex features can be chosen, the overall VC-dimension of the family of hypothesis available to the classifier increases and the above reasoning may become invalid. However, having the ability to chose between two different sets of features of similar complexity, the generalization performance should generaly be better when the set with smaller cardinality is chosen.

3.1 The $\mathcal{O}$ matrix and feature covers

Consider a matrix $\mathcal{O}$ where each column corresponds to a positive example in the training set and each row corresponds to a negative example in the same set. To every element of $\mathcal{O}$ in row r and column c, $\mathcal{O}_{rc}$ we associate a positive example, e_c^+ and a negative one, e_r^-.

Consider now a given feature f_i. This feature can be either a feature originally present in the problem specification or any boolean function of the original set of features. This

[1] The $VC-dimension$ is a simple combinatorial property of a set of functions that is related with the ability of a function on that set to fit an arbitrary set of data points. See [Blumer et Al. 1986] for details.

feature will have the value 1 for some examples in the training set and the value 0 for the remaining ones. Let $\mathcal{T}_i \subset \mathcal{T}$ be the set of examples in the training set that cause feature f_i to have the value 1 and $\mathcal{T}_i' = \mathcal{T} \setminus \mathcal{T}_i$ the set of examples that cause feature f_i to have the value 0.

Given a positive example (e_c^+) and a negative one (e_r^-), feature f_i can be used to distinguish between them iff one of the following two conditions holds:

a) $e_c^+ \in \mathcal{T}_i \wedge e_r^- \in \mathcal{T}_i'$

b) $e_c^+ \in \mathcal{T}_i' \wedge e_r^- \in \mathcal{T}_i$.

In the first case, feature f_i assumes the value 1 for example e_c^+ and the value 0 for e_r^- while in the second the opposite is true.

We will say that feature f_i covers element $\mathcal{O}_{rc}$ of $\mathcal{O}$ if feature f_i can be used to distinguish between e_c^+ and e_r^-. A given feature will, in general, cover several elements of $\mathcal{O}$. In particular f_i will cover all elements in the columns that correspond to positive examples in $\mathcal{P} \cap \mathcal{T}_i$ and the rows that correspond to negative examples in $\mathcal{N} \cap \mathcal{T}_i'$. It will also cover all elements in the columns $\mathcal{P} \cap \mathcal{T}_i'$ and rows $\mathcal{N} \cap \mathcal{T}_i$. If the order of the rows and columns is properly rearranged, the set of elements in $\mathcal{O}$ that are covered looks like two rectangles with the corners touching. We call them the rectangle and the co-rectangle associated with feature f_i. Figure 1 shows the elements of the $\mathcal{O}$ matrix covered by feature f_i.

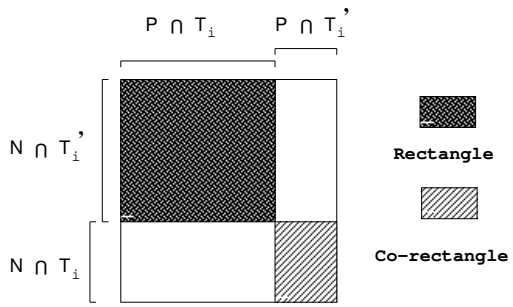

Figure 1: Elements of $\mathcal{O}$ covered by feature f_i

If every element of $\mathcal{O}$ is covered by at least one feature in $S = \{f_1, ..., f_k\}$ then S is called a cover for $\mathcal{O}$. In the noiseless case, a useful set of features should make it possible to distinguish between all positive and all negative examples. Since we want to minimize the number of features used, we want to find a **set S of minimal cardinality such that each element of the $\mathcal{O}$ matrix is covered by at least one feature in S.**

The following two properties are easy to prove:

Property 1: Feature f_i and feature $f_j = \overline{f_i}$ (obtained by negating the boolean function associated with f_i) cover the same elements in the $\mathcal{O}$ matrix.

Property 2: The set of input features, the conjunctions in any DNF representation of the concept and the terms in any CNF representation of the concept are all covers for $\mathcal{O}$.

Using property 1, one can restrict the type of features used. Since a disjunction is always the negation of some conjunction, restricting the features to be conjunctions does not loose generality. A useful disjunctive feature will always be found, although disguised as a conjunction. By property 2 it is clear that the problems of finding minimal DNF or minimal CNF representations are particular cases of the minimal $\mathcal{O}$ cover problem.

3.2 Example

Consider the concept defined by the boolean function $x_1 \oplus (x_2 x_3)$.

Both the minimum DNF and the minimum CNF description require 3 terms (or factors). However there is a cover with only 2 features, $\{x_1, x_2 x_3\}$.

Figure 2 shows how every element in $\mathcal{O}$ is covered by at least one of the features.

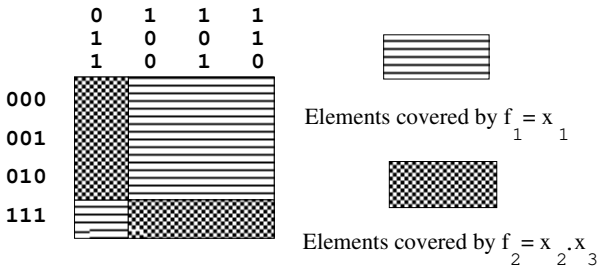

Figure 2: Cover formed by $f_1 = x_1$ and $f_2 = x_2 x_3$

4 The covering algorithm

The objective is to find a minimal set of features that are a cover for the $\mathcal{O}$ matrix. It is known [Garey & Johnson 1979] that finding minimum[2] DNF or CNF representations are NP-complete problems[3]. Since it was shown in section 3.1 that these problems are particular cases of the minimal cover problem, an algorithm that finds all the time a guaranteed minimum in polynomial time should not be expected.

We restrict the composite features to be conjunctions of input features or of already existing composite features. Because property 1 of section 3.1 is valid, this is equivalent to the simultaneous consideration of conjunctive and disjunctive features.

[2]We use the commonly accepted convention that minimal means a solution where all neighbor alternatives have higher values and reserve the word minimum for the absolute minimum over the whole space of solutions.

[3]Technically, the decision problems associated with these tasks are the NP-complete problems.

Consider the general problem of finding a minimal cover of elements in set A when a covering relation is established between these elements and the elements in the set to be covered, B. In our case, A would be the set of possible features and B the elements of $\mathcal{O}$. There are two major approaches for the problem of finding a minimal subset of A that covers all elements in B. If there is a way to generate a suitable set of candidate elements of A, the size and complexity of the covering problem can be reduced by considering only this set of candidates. For instance, when looking for a minimal DNF form, a good set of candidates is the set of prime implicants of the function.[4] It is easy to prove that there is always at least one optimal solution that consists only of prime implicants. This approach was used successfully in several two-level minimization packages, of which ESPRESSO [Brayton et Al. 1984] is the most successful example. However, when no good rules are available for the process of candidate selection, this reduction in the size of the problem cannot take place and the covering problem cannot be solved using reasonable resources.

An alternative method for solving a covering problem was proposed in [Oliveira & Sangiovanni 1991]. Since no good algorithm for the generation of a suitable set of candidates was known, we adopted an algorithm similar to that one, which is based on the use of a search tree.

4.1 Searching for a smaller cover

The algorithm is based on a search that proceeds by constructing a tree. Consider that we have a cover of k features, C. The target is to find a cover with $k-1$ features.

We build a search tree where each node except the root corresponds to a tentative cover with $k-1$ features. The root corresponds to the starting k feature cover.

Children of the root node are created by removing each one of the features in C. Therefore, the root has k children. Consider now another node in the tree, n_m. Let C'_m be the tentative cover associated with node n_m. Children of node n_m are created by the following algorithm:

1. Select an element $\mathcal{O}_{rc}$ not covered by any feature in C'_m.
2. Let f_i^m be the i*th* feature present in C'_m. For each f_i^m, change it in order to make it cover the element $\mathcal{O}_{rc}$. For each possible way in which this can be done, create a child node with the same C'_m except that feature f_i^m is replaced by the result obtained after the change is performed.

Step 2 of this process, the change of a feature in order to make it cover element $\mathcal{O}_{rc}$ can be done by either making feature f_i^m evaluate to 1 at example e_c^+ and to 0 at example e_r^- or the opposite. For the first case, the following steps are performed:

1. Remove from f_i^m any predicates that are in conflict with e_c^+. For example, if $f_i^m = x_1x_3$ and $e_c^+ = 1001$ remove x_3 from f_i^m obtaining $g_i^m = x_1$.
2. If g_i^m evaluates to 0 at e_r^- let $f_i^{m+1} = g_i^m$, set $l = 1$ and go to 4.
3. Consider the l extra predicates that can be added to g_i^m in order to make it evaluate to 0 at e_r^- while still evaluating to 1 at e_c^+. For example, if $e_r^- = 1111$, generate the following features: $f_i^{m+1} = x_1\overline{x_2}$, $f_i^{m+2} = x_1\overline{x}_3$ and set $l = 2$.
4. Check if features $f_i^{m+1}, f_i^{m+2}...f_i^{m+l}$ are valid. Discard any invalid ones. Use the valid features in $\{f_i^{m+1}, f_i^{m+2}...f_i^{m+l}\}$ to create children of node n_m.

The second case is dealt with using the same steps but with the roles of e_c^+ and e_r^- reversed.

This procedure tries to cover a new element of $\mathcal{O}$ while minimally changing only one feature in the tentative cover of node n_m. Children of node n_m will all have element $\mathcal{O}_{rc}$ covered by one feature n their tentative cover. At each step, the node which has less elements of $\mathcal{O}$ remaining to be covered is selected and its children generated. The check for validity in step 4 can be used to eliminate features that are not desired in the final solution. For example, one can restrict the search to features that correspond to cubes of no more than a fixed number of literals. Other restrictions based on the problem domain are also possible. An additional measure needs to be taken to guarantee termination: when creating a new node n_k with a tentative cover C'_k check to see if another node n_j with $C'_j = C'_k$ already exists. If so, do not create node n_k.

It is possible to prove that if this algorithm is carried to its end a cover with $k-1$ features will be found, if one exists. In practice, this result is of limited interest because the size of the tree is very large. In practice, the time and memory requirements of this algorithm are not unreasonable for many problems of interest because, in general, a good solution is found after a search in a very shallow tree.

4.2 The outer loops

The algorithm described above searches for a cover with $k-1$ features starting from one with k features. The process is started by creating a cover which contains the input features and iterated until no further reduction in the size of the cover is attained.

The algorithm is invoked again and tries to find new composite features that are conjunctions of literals corresponding to the features now present. The process can be iterated several times. This iteration stops when a set with only one feature is sufficient to cover the $\mathcal{O}$ matrix.

[4] An implicant of a function is a conjunction of literals that evaluates to 0 at all points where the function takes the value 0. An implicant is called prime if the removal of any literal in the conjunction makes the resulting expression evaluate to 1 at some point where the original function evaluates to 0.

5 Experimental results

MIFES, an implementation of the algorithm described in section 4 was used to compare the performance of the approach with alternative algorithms. We performed two experiments:

5.1 Experiment A

This is the same experiment described as experiment 3 in [Almuallim & Dietterich 1990]. Specifically, we selected 50 arbitrary concepts each depending on at most 5 (out of n) input attributes. For each of these concepts and for different values of n we measured the dependence of the accuracy of the hypothesis returned while successively increasing the sample size. We compared the performance of MIFES with the performance of ID3 [Quinlan 1986], FRINGE [Pagallo & Haussler 1990] and FOCUS [Almuallim & Dietterich 1990] under two different conditions:

1. MIFES was instructed to select only features consisting of conjunctions of no more than 1 literal. This is equivalent to selecting a minimal set of input attributes that is enough to characterize the examples, and is essentially the criterion implemented in FOCUS. We refer to the algorithm running with this restriction as MIFES_1.
2. MIFES was allowed to select composite features consisting of conjunctions of an arbitrary number of literals. We refer to this version as MIFES_N.

Figure 3 shows a pattern typical of the learning curves we observed for one of the concepts with $n = 16$. Also plotted is the optimal solution that can be reached by an algorithm that knows in advance the set of variables that f depends on. By simply returning one function defined over the relevant variables that is compatible with the observable data this algorithm achieves the optimal result[5]. In general, if there are p relevant variables and a sample with s examples is presented, such an algorithm will see, in the average, k_s different combinations of these p variables, where $k_i = k_{i-1} + (n - k_{i-1})/n$ and $k_0 = 0$. Therefore, the algorithm will get right a fraction $\frac{k_s}{2^p}$ of the examples in the test set and will guess right approximately 50% of the remaining ones, obtaining a final performance $P_{opt} = \frac{k_s + 2^p}{2^{p+1}}$.

MIFES_1 obtained essentially the same results as FOCUS and much better results than either FRINGE or ID3. This is somehow to be expected since both MIFES_1 and FOCUS implement the MIN-INPUT-FEATURES bias that is appropriate for this problem. The result obtained by MIFES_N is much more interesting, though. Being more general than either FOCUS or MIFES_1 in the type of concepts it can learn efficiently, it lags slightly behind when the number of examples has an intermediate value because many other

[5]This is true only if nothing can be known about the prior distribution of the classes.

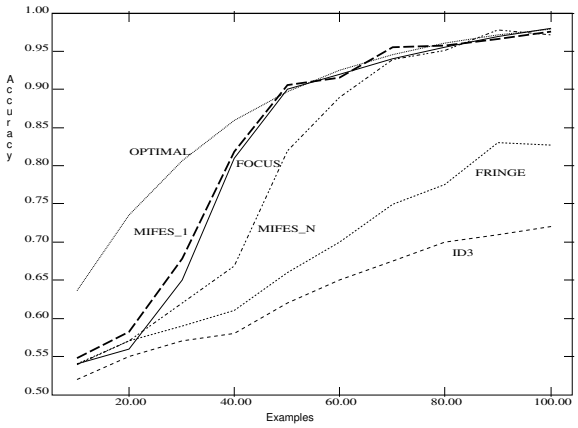

Figure 3: Learning curve for concept $f(x_1, ..., x_{16}) = x_1x_2x_3\overline{x}_4 + x_1x_2x_3x_4\overline{x}_5 + x_1x_2\overline{x}_3x_4x_5 + x_1\overline{x}_2x_3 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2x_3x_4x_5 + \overline{x}_1\overline{x}_2x_3x_4x_5 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1\overline{x}_3x_4x_5 + \overline{x}_1\overline{x}_3\overline{x}_4\overline{x}_5$ which has 5 relevant features out of 16.

compatible hypothesis defined over sets of 5 or less composite features still exist. However, when the number of examples is sufficient to isolate the right hypothesis the performance of MIFES_N increases rapidly and it performs much better than either FRINGE or ID3.

5.2 Experiment B

This experiment was designed to test the ability of the algorithm to obtain the minimum multi-level realization of a given concept. For that we selected 5 concepts that accept small multi-level representations but that require longer DNF or CNF representations: The boolean functions that identify each of these concepts are the following:

$$\mathcal{C}_A(x_1...x_{12}) = (x_1x_2 + x_3x_4 + x_5x_6)(x_7x_8 + x_9x_{10} + x_{11}x_{12})$$

$$\mathcal{C}_B(x_1...x_{18}) = (x_1x_2 + x_3x_4 + x_5x_6)(x_7x_8 + x_9x_{10} + x_{11}x_{12})(x_{13}x_{14} + x_{15}x_{16} + x_{17}x_{18})$$

$$\mathcal{C}_C(x_1...x_9) = x_1x_2x_3 + x_4x_5x_6 + x_7x_8x_9$$

$$\mathcal{C}_D(x_1...x_{18}) = (x_1x_2x_3 + x_4x_5x_6 + x_7x_8x_9)(x_{10}x_{11}x_{12} + x_{13}x_{14}x_{15} + x_{16}x_{17}x_{18})$$

$$\mathcal{C}_E(x_1...x_{27}) = (x_1x_2x_3 + x_4x_5x_6 + x_7x_8x_9)(x_{10}x_{11}x_{12} + x_{13}x_{14}x_{15} + x_{16}x_{17}x_{18})(x_{19}x_{20}x_{21} + x_{22}x_{23}x_{24} + x_{25}x_{26}x_{27})$$

Table 1 lists the results obtained. The second column gives the number of examples used for each concept. This number of examples was selected empirically to allow for only one hypothesis no more complex than the target complex to be consistent with the training data. Examples were selected randomly using the random number generator *random*.

Column 3 and 4 give the minimum number of new conjunctive features that are required if realizations corresponding to minimum DNF and CNF forms are to be obtained. The fifth column gives the minimum number of conjunctive features required to implement a minimum multi-level representation of the target concept. Finally, column 6 gives the

number of conjunctive features created by MIFES.

Concept	Ex	DNF	CNF	Fac	MIFES
$\mathcal{C}_A$	300	10	17	9	9
$\mathcal{C}_B$	500	28	25	14	14
$\mathcal{C}_C$	100	4	28	4	4
$\mathcal{C}_D$	300	10	55	9	9
$\mathcal{C}_E$	800	28	82	14	34

Table 1: Concept characteristics and results.

MIFES was able to find the minimum multi-level representations for the first 4 examples. For the last one, if failed to do so and found a multi-level representation that was actually more expensive (in terms of conjunctions count) than the minimum DNF form. We conjecture that better search procedures are required to solve problems of this complexity.

6 Conclusions

We presented and implemented a non-greedy algorithm that selects a minimal set of features that is sufficient to distinguish between all the positive and negative examples in the training set. The algorithm can be used to select minimal sets of features subject to different restrictions.

Both the applicability and some limitations of the method were analyzed with the help of some examples. We have shown that the algorithm can implement the MIN-INPUT-FEATURES bias in an efficient way and can also be used to find minimal multi-level realizations of the target concept characterist function.

The current version of the search algorithm may, however, prove too slow to be usable in real life problems. More efficient realizations of the search procedure are certainly possible. The modification of the algorithm to make it possible to deal with problems with noise and problems with non-boolean attributes is also an interesting direction for future research.

Acknowledgements

This work was supported by the Air Force Office of Scientific Research (AFOSR/JSEP) under Contract No. F49620-90-C-0029 and the Portuguese INVOTAN committee.

References

[Almuallim & Dietterich 1990] H. Almuallim & T. G. Dietterich "Learning with many irrelevant features", Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 547-552, 1990, Anaheim, CA, AAAI Press.

[Brayton et Al. 1984] R. K. Brayton, G. D. Hachtel, C. T. McMullen & A. L. Sangiovanni-Vincentelli "Logic Minimization Algorithms for VLSI Synthesis", Kluwer Academic Publishers, 1984.

[Brayton et Al. 1990] R. K. Brayton, G. D. Hachtel & A. L. Sangiovanni-Vincentelli "Multilevel Logic Synthesis" Proceedings of the IEEE, Vol. 78:2, February 1990.

[Blumer et Al. 1986] A. Blumer, A. Ehrenfeucht, D. Haussler & M. K. Warmuth "Classifying Learnable Geometric Concepts with the Vapnik-Chervonenkis Dimension", Proceedings of the Eighteenth Annual ACM Symposium of Theory of Computing, pp 273-282, Berkeley, CA, 1986.

[Drastal et Al. 1989] G. Drastal, G. Czako & S. Raatz "Induction in an Abstraction Space: A Form of Constructive Induction", Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 708-712, Detroit, MI, Morgan Kaufmann.

[Garey & Johnson 1979] M. R. Garey & D. S. Johnson "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman, San Francisco, 1979.

[Michalski et Al. 1986] R. S. Michalski, I. Mozetic, J. Hong, & N. Lavrac "The multipurpose incremental learning system AQ15 and its testing application to three medical domains", Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 1041-1045, Philadelphia, PA, 1986.

[Oliveira & Sangiovanni 1991] A. L. Oliveira & A. Sangiovanni Vincentelli "LSAT - An Algorithm for the Synthesis of Two-Level Threshold Gate Networks", Proceedings of the International Conference on Computer Aided Design, pp. 130-133, 1991, Santa Clara, CA, IEEE Computer Society Press.

[Pagallo & Haussler 1990] G. Pagallo & D. Haussler "Boolean Feature Discovery in Empirical Learning", Machine Learning, 5, pp. 71-99, 1990.

[Quinlan 1986] J. R. Quinlan "Induction of Decision Trees" Machine Learning, 1, pp.81-106, 1986.

[Schilmer & Granger 1986] J. C. Schilmer & R. H. Granger Jr. "Incremental Learning from Noisy Data", Machine Learning, 1, pp. 317-354.

[Utgoff 1986] P. E. Utgoff "Shift of Bias for Inductive Concept Learning" in Machine Learning: An Artificial Intelligence Approach, 1986, Morgan Kaufmann.

[Wnek & Michalski 1991] J. Wnek & R. S. Michalski, "Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments" Proceedings of IJCAI-91 Workshop on Evaluating and Changing Representations in Machine Learning, pp. 13-22, Sydney, Australia, 1991.

[Yang et Al. 1991] D. Yang, L. Rendell & G. Blix "Fringe-Like Feature Construction: A Comparative Study and a Unifying Scheme", Proceedings of the Eight Workshop in Machine Learning, pp. 223-227, 1991, Chicago, IL, Morgan Kaufmann.