AN EXPERIMENTAL COMPARISON OF SYMBOLIC & SUBSYMBOLIC LEARNING PARADIGMS: PHASE I - LEARNING LOGIC-STYLE CONCEPTS

by

J. Wnek
R. S. Michalski

Proceedings of the First International Workshop on Multistrategy Learning, MSL-91,
Harpers Ferry, W. VA., Nov. 7-9, 1991.

An Experimental Comparison of Symbolic and Subsymbolic Learning Paradigms: Phase I - Learning Logic-style Concepts

Janusz Wnek and Ryszard S. Michalski
Center for Artificial Intelligence
George Mason University
4400 University Dr., Fairfax, VA 22030
jwnek@aic.gmu.edu, michalski@aic.gmu.edu

Accepted for the First International Multi-Strategy Learning Workshop
Harpers Ferry, WV, November 7-9, 1991

An Experimental Comparison of Symbolic and Subsymbolic Learning Paradigms: Phase I - Learning Logic-style Concepts

Janusz Wnek and Ryszard S. Michalski
Center for Artificial Intelligence
George Mason University
4400 University Dr., Fairfax, VA 22030
jwnek@aic.gmu.edu, michalski@aic.gmu.edu

Abstract

The paper discusses and experimentally compares five different methods for concept learning from examples. The first three are symbolic methods, specifically, a decision tree learning method (C4.5), a rule learning method (AQ15), and a constructive rule learning method (AQ17-HCI). The other two are non-symbolic methods, one, a neural net trained by a backpropagation algorithm, (BpNet), and a second, classifier system employing a genetic algorithm (CFS). All methods have been experimentally applied to various concept learning problems. This paper reports the first phase of experiments where concepts to be learned were proposed by human subjects, and thus "cognitively oriented." The second phase will involve learning other types concepts. To analyze the performance of the programs, a diagrammatic visualization system, DIAV, was employed. DIAV presents learned and target concepts as images in a planar model of a multidimensional space, and permits one to visualize exact error of a learning process. In several experiments, symbolic methods, in particular the AQ17-HCI method, consistently outperformed subsymbolic methods in terms of both, predictive accuracy and simplicity of learned descriptions.

Key words: symbolic learning, neural networks, genetic algorithms, diagrammatic visualization.

1 Introduction

At present, there is a growing interest in developing multistrategy systems that integrate different learning strategies within one system. From this perspective, it is important to get an insight into the performance of various learning methods, and to determine the areas of their applicability. To this end, this paper reports a study on comparing the performance of symbolic and subsymbolic methods on the same class of problems. Symbolic methods included in the study are represented by decision trees algorithm C4.5, decision rule learning algorithm AQ15, and constructive decision rule learning algorithm AQ17-HCI. Genetic algorithms were represented by the CFS classifier system, and neural networks by the backpropagation algorithm.

An important difference between symbolic and subsymbolic learning approaches is in the cognitive aspects of the representation they use. Knowledge represented by logic-based rules and decision trees (especially when they are small) is relatively easy to comprehend. This is not the case with knowledge represented by classifier systems or neural networks. Therefore, to evaluate the applicability of the methods to different problem areas, three problem types are distinguished based on the way they originated:

1) Learning concepts which reflect typical human descriptions of classes of entities

("cognitively-oriented" concepts). The concepts were generated by human subjects, and were in the form of logic-style expressions.
2) Learning concepts that represent some technical problems or transformations.
3) Learning concepts generated randomly.

This classification is useful because in some applications (e.g., in expert systems for human disease diagnosis) the comprehensibility of the learned knowledge is a crucial condition; while in some other applications (e.g., adaptive control of house temperature) this issue may be irrelevant. The random concepts are used to test the ability to learn any type of concepts.

In the first phase of our study, presented here, we compared various methods on problems of the first type, specifically, on learning concepts that are generated by human subjects. The current study therefore can be viewed as favoring methods employing symbolic representations, because such representations are partially suitable for representing "cognitively-oriented" concepts. The second phase is concerned with learning problems of the second and the third type, e.g., a 6- and 11-multiplexer, parity functions, text-to-speech mapping (Sejnowski and Rosenberg, 1987), and several randomly generated problems, which will be published in a separate paper.

This study follows several other efforts of comparing different methods and paradigms. For example, Fisher and McKusick (1989) compared ID3 and a neural net using BP algorithm on the problems of learning diagnostic rules for thyroid diseases and soybean plant diseases, and a few artificial problems. The comparison was based on the performance accuracy of testing examples and the training time. Their conclusion was that the neural net gave a better performance, but required a significantly longer training time and more training examples than ID3. Mooney et al (1989) compared ID3 with perceptron and backpropagation using the domain of soybean diseases, chess-end games, audiological disorders, and the Nettalk data set. Their conclusion was that the accuracy of classifying new examples was about the same for all the three systems, but the neural net performed better than ID3 when there was noise in the data. Weiss and Kapouleas (1989) compared ID3, predictive value maximization, neural net using BP, and a few statistical methods. They found that the statistical classifiers performed consistently better in terms of accuracy in classifying testing examples. Recently, Dietterich, Hild and Bakiri (1990) compared ID3 with a neural net using BP on the task of text-to-speech mapping. Their major conclusion was that neural net consistently outperformed ID3 in terms of the performance accuracy, and attributed this result to capture better statistical information by the neural net. In another recent study, Bergadano et al. (1991) compared POSEIDON (an extended version of AQ15 using a two-tiered concept representation) with exemplar-based type and decision tree type (ASSISTANT) learning programs. Their study involved two real world problems: labor contracts and congressional voting. Descriptions learned by POSEIDON outperformed those produced by the other methods, both in terms of performance accuracy on new examples and in terms of the description's simplicity.

In the past few years symbolic methods utilizing various forms of *constructive induction* (Michalski, 1978; Rendell and Shesu, 1990) showed improvement in performance accuracy and smaller complexity of descriptions learned over pure selective methods (Pagallo and Haussler, 1990; Wnek and Michalski, 1991).

This study differs from the studies mentioned above in that it experimentally analyzes five different methods. The methods are compared in terms of the *exact error rate* (rather than a statistical estimate), and also in terms of the complexity of the descriptions learned. The target and learned concepts are represented graphically by a novel technique of a *diagrammatic visualization* (Wnek & Michalski, 1991). This technique permits one to display an *error image* that locates all errors precisely.

The comparison of the methods was done both in terms of the accuracy of the descriptions learned, and in terms of their complexity. The concept of *rule complexity of a representation*, or briefly, *R-complexity* was introduced in order to have some way to approximate the "cognitive" complexity of representations learned by these diverse methods.

2 Learning Systems Compared

As mentioned earlier, the symbolic paradigm was represented by the AQ15 and AQ17-HCI rule learning programs, and the C4.5 decision trees learning program. Of the various neural net algorithms developed, BpNet, the back-propagation has been the most popular. The shell for the classifier system based on genetic algorithm was developed by R. Riolo (1988).

2.1 Decision tree learning program C4.5

The C4.5 program is a derivative of the ID3 program (Quinlan, 1986). ID3 builds the decision tree as the representation for the concept. Each interior node of the tree is associated with an attribute value while the leaf node represents the concept class, which is a conjunction of the attribute values. The arc from the interior node represents a value of the attribute. Each path in the decision tree can then be considered as a distinct decision rule, which is mutually exclusive.

The algorithm starts with a training set of examples belonging to different classes. It selects a random subset of examples (window) and compares the information measures of each attribute. The attribute having the highest score is selected as the root of the tree. From this it generates a decision tree, adds misclassified objects and continues until the trial decision tree correctly classifies all objects not in the window. The algorithm iterates until each node has only events that belong to one class. The entire process is repeated by default 10 times.

2.2 Rule learning program - AQ15

AQ15 generates a set of concept descriptions from examples of concepts. The descriptions are in the form of decision rules. They can be built and optimized according to a variety of problem dependent criteria (Michalski et al, 1986). The rules are expressed in the attributional calculus language, VL1 (Michalski and Larson, 1983). The main procedure of AQ15 is based on the AQ algorithm that builds a concept description from a set of positive and negative examples. Below is a simplified version of the AQ algorithm:

1. Select a *seed* example from the set of positive training examples for a given decision class.
2. Generate a set of alternative most general rules (a *star*) that cover the seed & do not cover negative examples (using the "extension against" generalization operator).
3. Select the "best rule" from the star, and remove from the set of training examples those covered by the rule.
4. If the set of training examples is not empty, go to step 1, otherwise, repeat the process for other classes.

The AQ15 has many other procedures and various control parameters. In all experiments the preference criterion was to "minimize the number of rules and the number of conditions in them." The other major control parameters were: "no truncation," "strict matching," and "intersecting covers."

2.3 Constructive rule learning program - AQ17-HCI

AQ17-HCI is based on the AQ algorithm and utilizes a constructive induction method in which problem-relevant attributes are identified and/or generated by analyzing consecutively created inductive hypotheses (Wnek & Michalski, 1991). The algorithm used in AQ17-HCI (Hypothesis-driven Constructive Induction) is:

1. Induce rules for each decision class from a subset of training examples using AQ15.
2. Analyze the rules to identify irrelevant attributes and/or attribute values.
3. For each decision class generate one candidate attribute that corresponds to a subset of the highest quality rules.
4. Modify training examples by adding newly generated attributes and removing irrelevant ones.
5. Induce rules from the modified training set.
6. Evaluate the predictive accuracy of the rules on remaining training examples. If the performance does not exceed a predefined threshold, go to step 2.
7. Induce rules from the complete training set using all relevant initial and derived attributes.

2.4 Neural net - BpNet

Backpropagation is a learning technique for feedforward networks, i.e., networks in which the interconnections form no feedback loops. We consider a network of units in which a weighted sum of the inputs is performed, the result of this sum (also called the activation level of the unit) is being fed through a non-linear element, with a differentiable input output function S. Here we use a sigmoid function.

Each of the five backpropagation nets was trained using the corresponding positive and negative examples coded as binary strings. The learning parameters in BpNet system were set to: eta = 0.25, alpha = 0.50; the number of hidden units was experimentally determined and set to 10% of the total number of input and output units. Networks have been trained until they reached r.m.s. (root mean square) error below 0.0007.

2.5 Classifier system - CFS

The program performs the following steps of the *major cycle* of the classifier system:

- compares messages with classifiers and record all matches
- calculates bids, runs a competition, generates new messages by activating the strongest classifiers; matches and activates effectors
- redistributes the payoff between classifiers by applying the bucket-brigade algorithm (BBA)
- applies genetic algorithms operators: crossover and mutation to generate new classifiers.

The classification was done by CLASS system which implements domain-dependent parts of the classifier system, e.g., emulation of detectors and effectors, payoff function. The system works in the stimulus-response mode (no internal messages are posted). The population of 60 classifiers is trained as the examples are fed into the system. The training process takes about 50 classifier system cycles per example. The reward for correct/incorrect answer is 6/-1, respectively (a full reward is paid to all active classifiers). The CLASS system has two effectors. Each of them consists of two basic parts: a condition and an action which is a classification.

3 Experimental Methodology

The above methods were compared by applying them to the same group of concept learning problems. In our experiments, five target concepts were generated by different human subjects. For each target concept, increasing sets of training examples were generated, in order to determine the convergence of the learned concepts to the target concepts. The learned descriptions were compared in terms of the *exact error rate*, a representation-dependent complexity, and an estimate of the representation-independent *R-complexity* (*rule complexity*).

Exact error is defined as the cardinality of the set-difference between the learned concept and the target concept.

Exact error rate is the ratio between exact error and the size of event space. It was measured as a function of the number of training examples.

A precise measure of the *R-complexity* of a representation is defined by the number of conjunctive statements (rules) in the minimal DNF expression that is logically equivalent to the given representation. Since finding such a minimal DNF expression for any given representation may be difficult (it is generally an NP-hard task), we use an estimate of the R-complexity. For a method learning a rule-based representation, the number of rules generated by the method is simply taken as an estimate of R-complexity. For a decision tree learning method, the R-complexity is estimated by the number of leaves in the tree (since each leaf corresponds to a rule).

For learning in neural nets and classifiers, the R-complexity is estimated by determining the number of conjunctive statements needed to re express the learned concept as a DNF formula. Such a formula can be obtained by determining a "cover" of the *image* of the learned concept in its *diagrammatic representation*. This representation employs the *General Logic Diagram* (GLD), which transforms a multidimensional space into a plane (Michalski, 1973; Wnek & Michalski, 1991; see section 3.3).

3.1 ROBOTS domain

The experimental domain for testing the methods was the world of ROBOTS in the EMERALD[1] system, a large-scale system integrating several machine learning programs. For simplicity, the robots are described by just six multiple-valued attributes (Figure 1). The event space (the space of all possible robot descriptions) is 432, and the space of possible concepts is the set of all non-empty subsets of this space, i.e., 2^{432} -1.

Attribute	Attribute value
Head Shape	round, square, octagonal
Body Shape	round, square, octagonal
Smiling	yes, no
Holding	sword, balloon, flag
Jacket Color	red, yellow, green, blue
Tie	yes, no

Figure 1. Attributes used to describe concepts in the first experiment.

3.2 Target concepts

In the experiment, five human subjects, (undergraduate computer science students), were asked to create five different concept descriptions characterizing various selections of robots from a predefined set of robots in the EMERALD system (20 examples). Each such concept represents a partitioning of the event space into those robots that belong to the concept (positive examples) and those that do not (negative examples) (Figure 2).

C1: *Head is round and jacket is red or head is square and is holding a balloon*
C2: *Smiling and is holding balloon or head is round*
C3: *Smiling and not holding sword*
C4: *Jacket is red and is wearing no tie or head is round and is smiling*
C5: *Smiling and holding balloon or sword*

Figure 2. The target concepts.

[1] EMERALD was developed at the Center for Artificial Intelligence at George Mason University; an earlier version of it was developed at the University of Illinois at Urbana Champaign (Kaufman, Michalski and Schultz, 1989).

Each concept description was the basis for generating five sets of training examples. The initial set was generated by the subjects who created the concepts: subset Pos1 consisted of 6% of the complete set of positive examples and subset Neg1 consisted of 3% of the complete set of negative examples of a concept. The remaining sets were generated by adding to the initial set an appropriate number of randomly generated examples: Pos2 and Neg2 (10% pos and 10% neg), Pos3 and Neg3 (15% pos and 10% neg), Pos4 and Neg4 (25% pos and 10% neg), Pos5 and Neg5 (100% pos and 10% neg). Figures 3&4 visualize the ROBOTS domain, concepts, and initial training sets for each concept (6% pos and 3% neg examples).

3.3 Diagrammatic visualization

To illustrate learned concept descriptions and locate their errors, as well as to determine their R-complexity, a *diagrammatic visualization* technique was employed (Wnek & Michalski, 1991). This technique uses a planar diagram for representing a multidimensional space spanned over multi-valued attributes, and permits one to display the *error image*, which is an exact characterization of the errors (the set-difference between the target and learned concepts).

In such a diagram, each combination of the values of the attributes (an instance of a concept) is represented as a small cell. For example, the top left corner cell in the diagram in Figure 3 represents a "robot" example described by the following attribute-value vector: **H**ead **S**hape = **r**ound, **B**ody **S**hape = **r**ound, **SM**iling = **y**es, **HO**lding = **s**word, **J**acket **C**olor = **r**ed, **TI**e = **y**es. Positive and negative training examples are marked with + and -, respectively. Concepts and errors are represented as black or shaded areas. The areas of the *target concept* not covered by the *learned concept* represent *errors of omission*, while areas of the learned concept not covered by the target concept represent *errors of commission*. The union of both types of errors represents the *error image* .

4 Representations learned

Figure 5 presents an example of the representation learned by each method from 6% pos and 3% neg examples of the concept *C1:*

(*"Head is round and jacket is red or head is square and is holding a balloon"*).

4.1 A decision tree generated by C4.5

The method used two attributes in describing the concept C1: Jacket Color and Head Shape (Figure 5A). The learned concept can be read as follows:

If Jacket Color is red and Head Shape is round or
Jacket Color is red and Head Shape is square or
Jacket Color is green and Head Shape is square
then C1

4.2 A decision rule generated by AQ15

The method generated one rule description. It consists of three conditions. Each condition tests one attribute. The internal disjunctions (inside condition) make the rule more concise (Figure 5B).

4.3 A decision rule generated by AQ17

The rule generated by AQ17-HCI has the same description as the one generated by AQ15 however, it was generated in a transformed, smaller description space (see also Figure 6). The example shows a change in the ROBOTS representation space by removing irrelevant attributes and some of the attribute values. Transformation A-TRANS removes three attributes: Body Shape, SMiling, and TIe. Transformation V-TRANS replaces two Jacket Color values: yellow & blue with dummy value X (Figure 5C).

4.4 A neural net generated by BpNet

Figure 5D shows a 20-node net trained by a backpropagation algorithm. In each pair (a, b) a is the weight of the link to the left hidden node, and b is the weight to the link to the right node.

4.5 Classifiers generated by CFS

Each line in the Figure 5E represents one classifier in the following format: No, Id, Classifier, Strength, and BidRatio (Riolo, 1988). The total population for representing the concept consists of 60 classifiers. Each of the classifiers (condition-action rules) is in the following form:

condition1, condition2 / action

Each condition consists of a string of fixed length 16 built from the tertiary alphabet {0, 1, #}. A condition string with prefix "m" is matched by any message that has 0's and 1's in exactly the same positions as the 0's and 1's in the condition string. The # in the condition is considered as a "wildcard" symbol that can match a 0 or a 1. A classifiers condition-part is satisfied when both of its conditions are matched. When the condition-part of a classifier is satisfied, the classifier becomes active, i.e. its action-part produces one or more output messages. These messages will activate output interface to generate final classification.

In order to selectively activate only some of the classifiers satisfied during a major-cycle, classifiers compete to become active. That is, each classifier that has its condition-part satisfied makes a bid to become active. A competition is then carried out and only the highest bidders are allowed to become active. *BidRatio* is a number between one and zero that is a measure of the classifier's specificity, i.e., how many different messages it can match. *Strength* is meant to be a measure of a classifier's "usefulness" to the system. The higher a classifier's strength, the more it bids.

The number of classifiers and about 20 other parameters were determined experimentally; the remaining parameters from the total of about 150 took default values). For details see (Wnek, 1990).

5 Experimental results

The individual diagrams in Figures 6 & 7 present the target concept C1 and the concepts learned by C4.5, AQ15, AQ17-HCI, BpNet, and CFS. In this case, the training set consists of 6% of all possible positive examples (84), and of 3% of all possible negative examples (348).

The target concept is represented as a union of black and sparsely shaded areas. The learned concept is represented as a union of black and densely shaded areas. The part of the target concept that was learned by a system is represented by black areas (Figures 6 & 7).

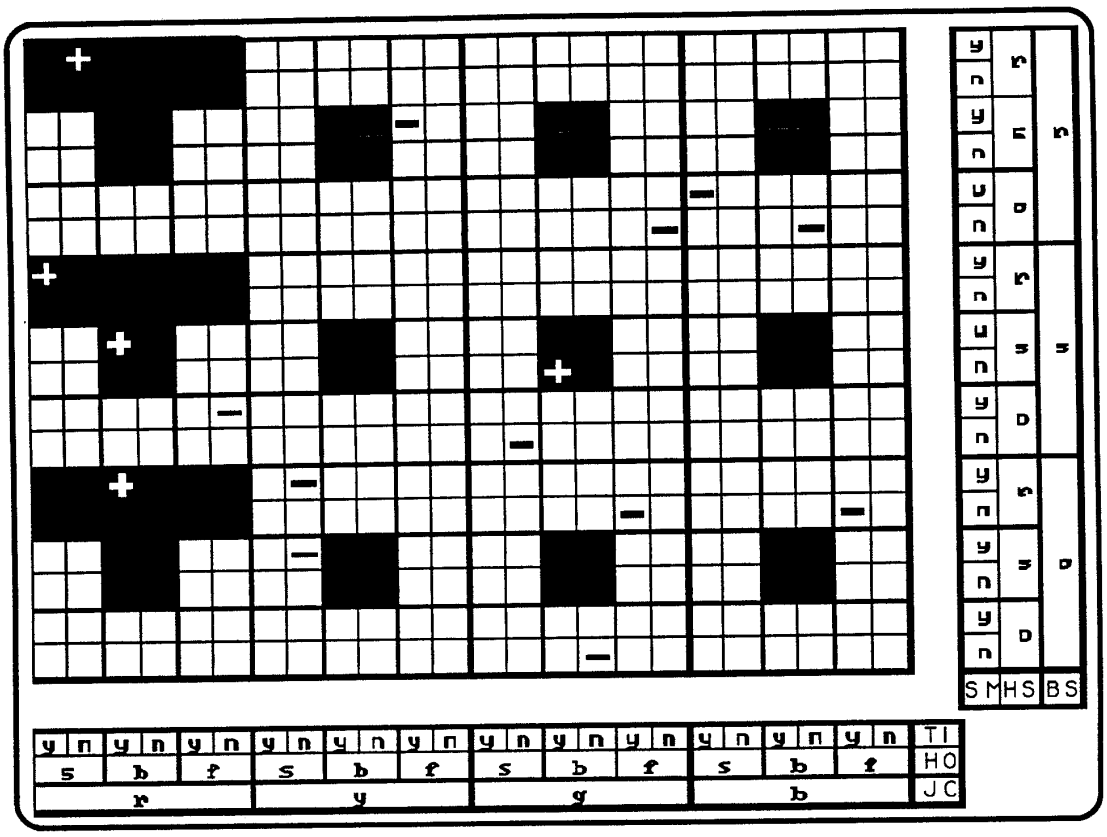


Figure 3. A visualization of the target concept C1 and the initial set of training examples.

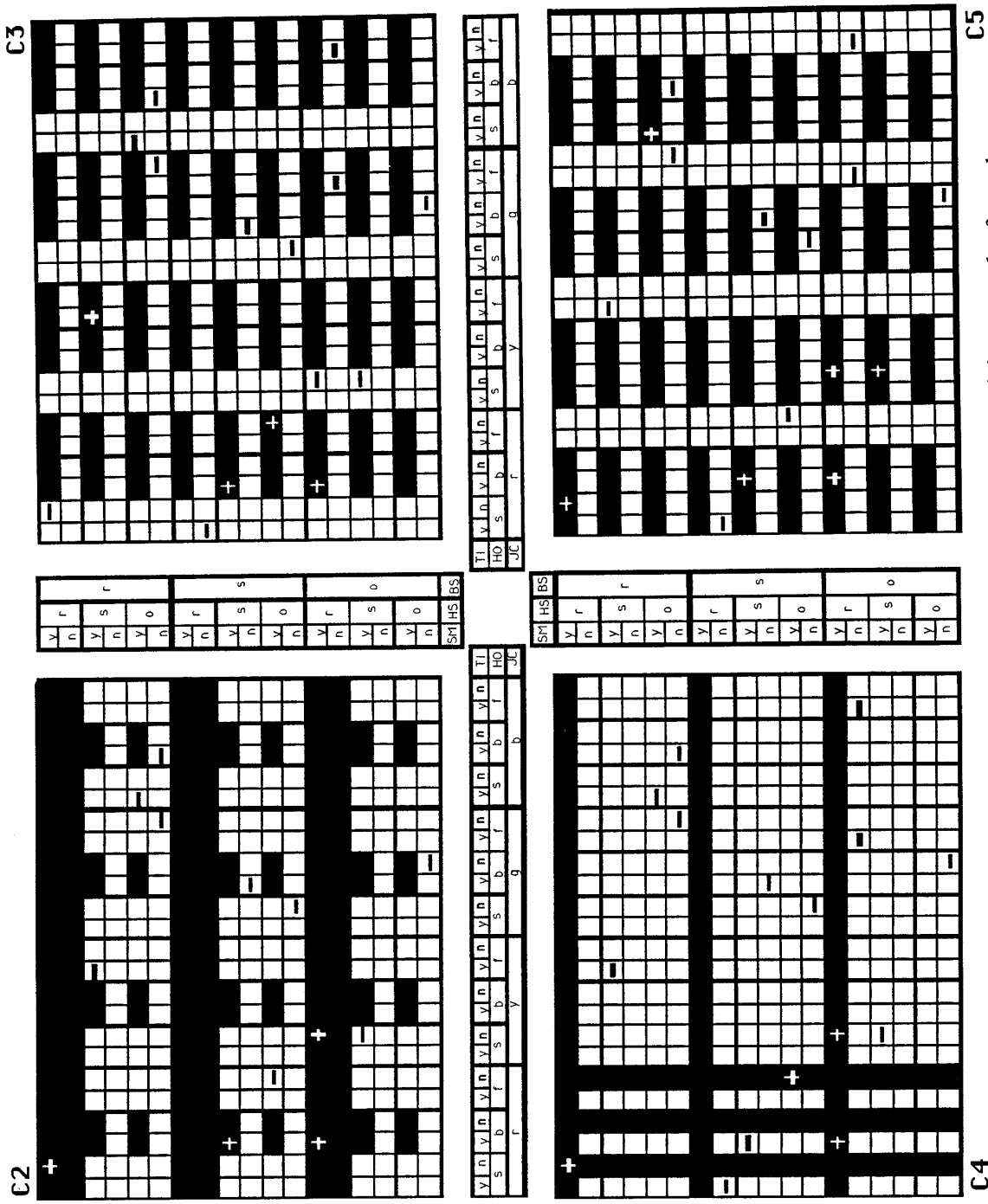
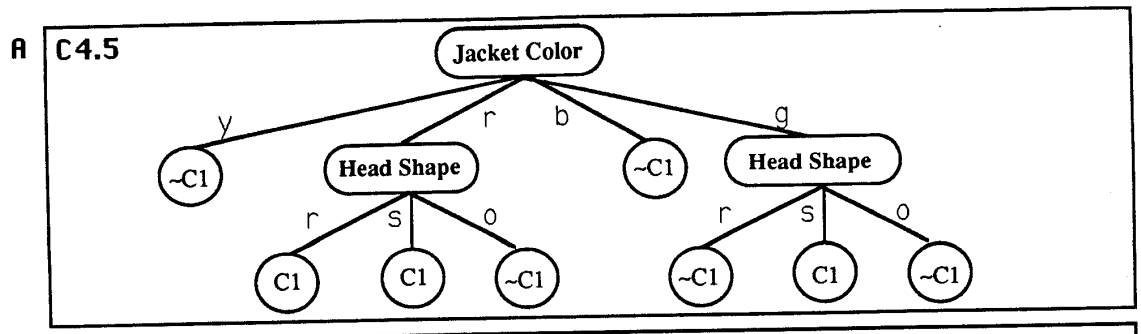


Figure 4. A visualization of target concepts C2, C3, C4 & C5 and initial training examples for each concept.

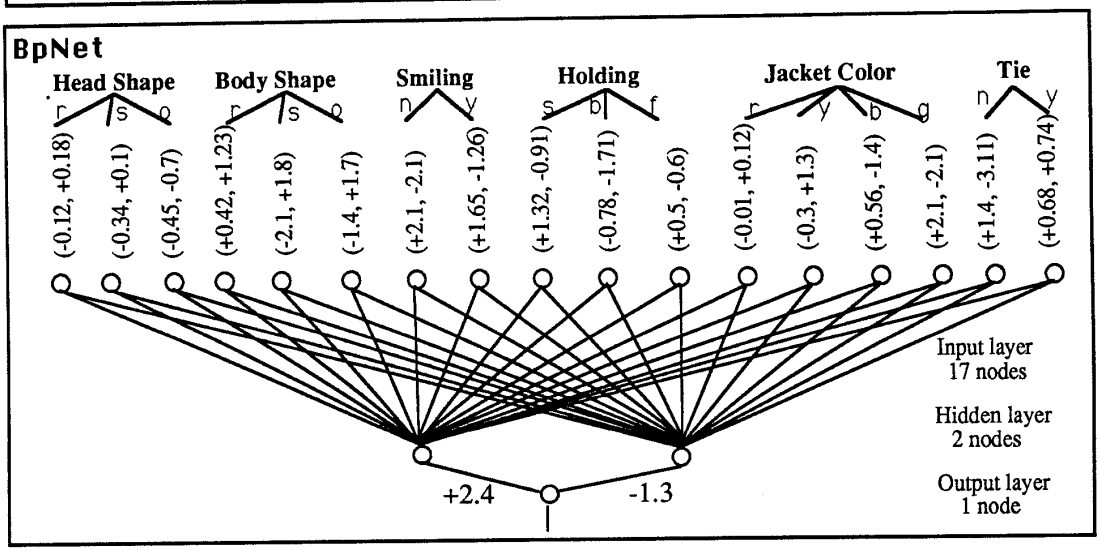


B AQ15

If (**Head Shape** = round or square) & (**Holding** = sword or balloon) & (**Jacket Color** = red or green) then C1

C AQ17-HCI

If (**Head Shape** = round or square) & (**Holding** = sword or balloon) & (**Jacket Color** = red or green) then C1



E CFS

No	Id	Condition1							Condition2	Action	Strength	BidRatio
			HS	BS	SM	HO	JC	TI				
1	1227	m000000	01	0#	1	10	#1		1, m00#00# 01 00 1 10 #1 #	/ 101101100#0000 0#	222	0.81
2	1236	m000000	01	0#	1	10	#1		1, m00#00# 01 00 1 10 #1 #	/ 101101100#0000 0#	219	0.81
3	1217	m0#0000	00	10	1	00	0#		0, m##0000 #0 10 1 00 #0 #	/ 100#1101101100 11	208	0.78
•												
•												
60	0017	m000000	01	01	1	10	01		1, m00000# 01 00 1 10 11 1	/ 101101100#0000 0#	34	0.25

Figure 5. Representations of the concept C1 learned from the initial set.

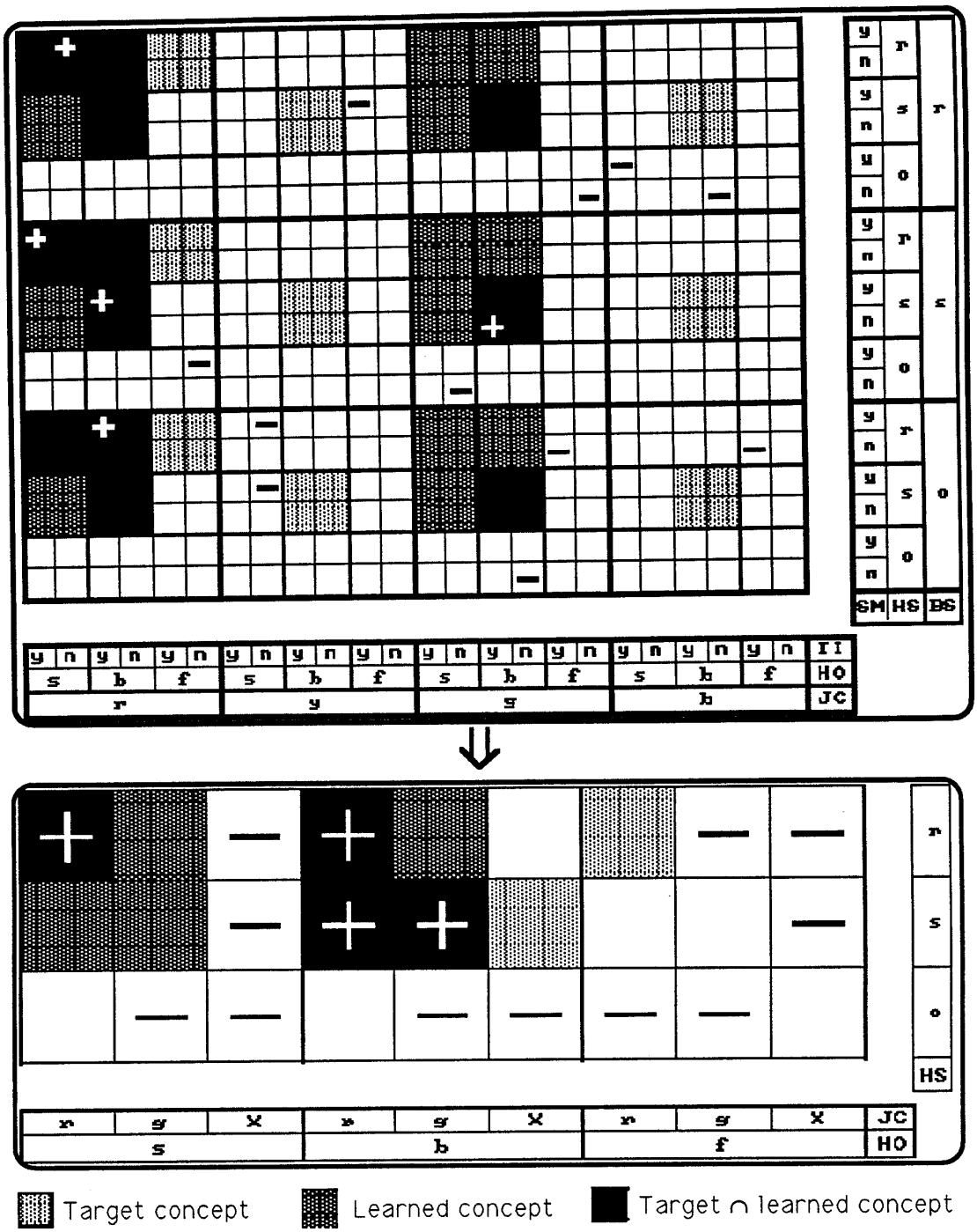


Figure 6. An illustration of the description space transformation done by AQ17-HCI:
A-TRANS: (HS, BS, SM, JC, HO, TI) -> (HS, JC, HO)
V-TRANS: [JC(r, y, g, b)] -> [JC(r, g, X)]

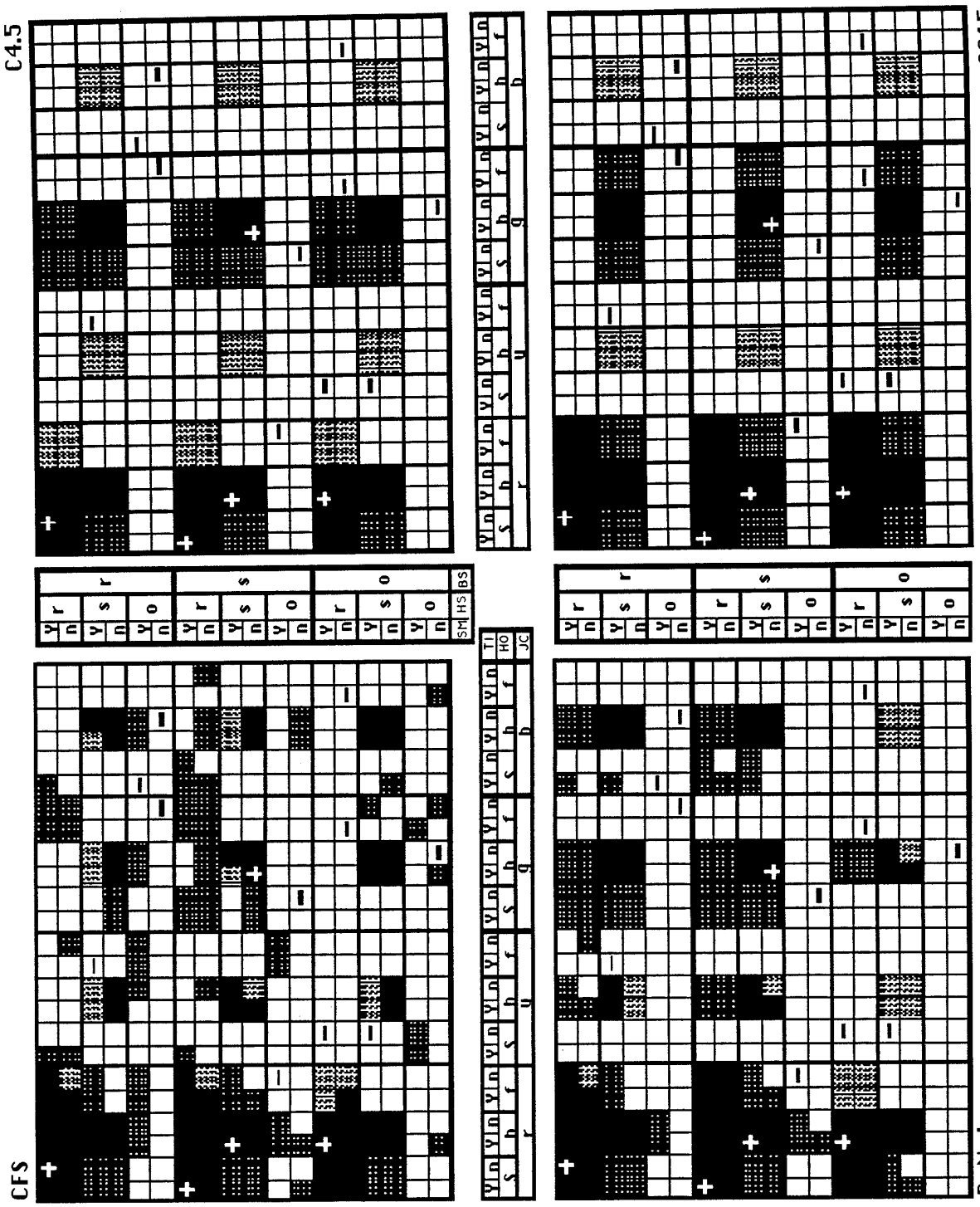


Figure 7. The concept C1 learned by different methods from the initial set of examples (6% pos and 3% neg examples).

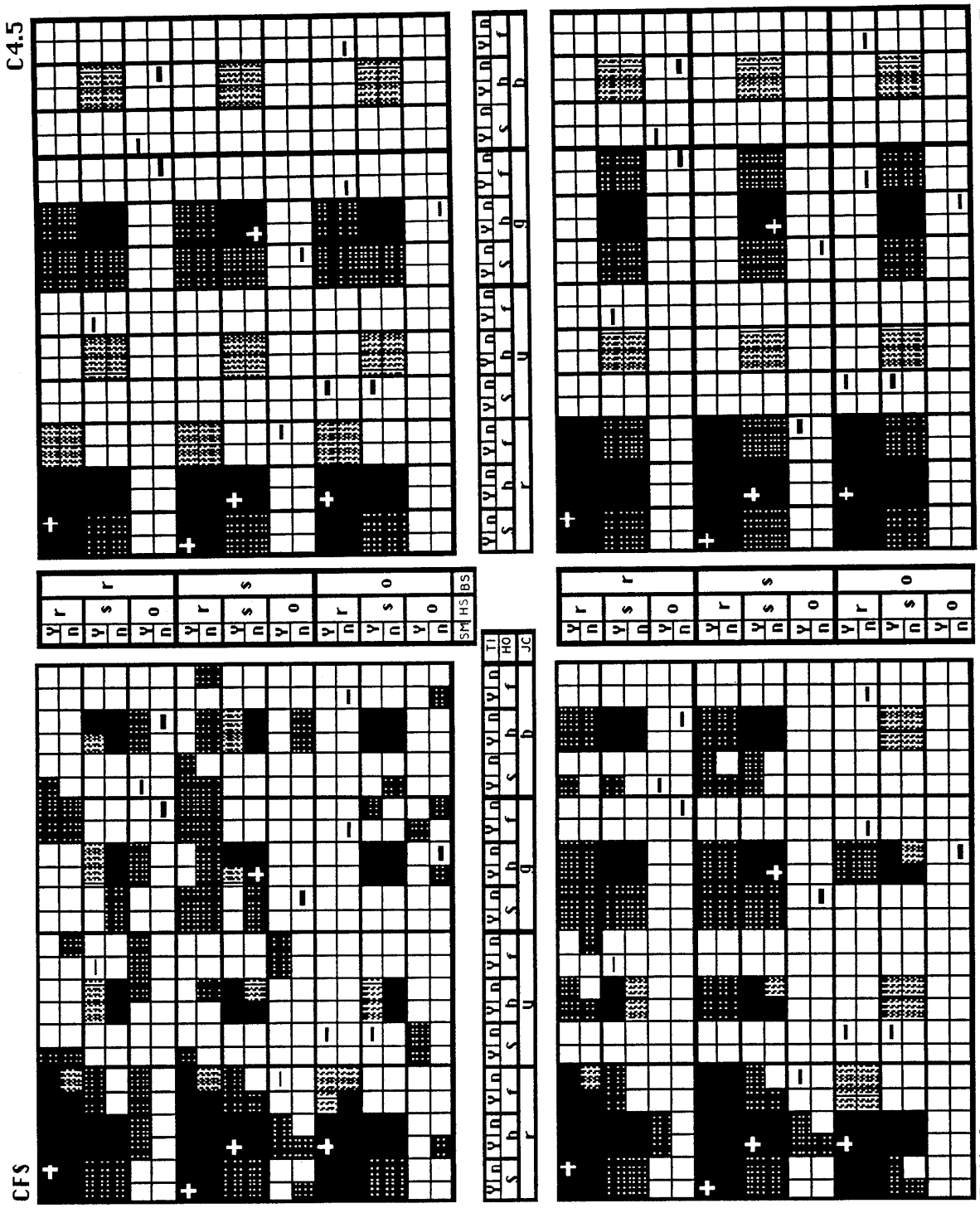


Figure 8. The concept C1 learned by different methods from the initial set of examples (6% pos and 3% neg examples).

	Percentage of Pos and Neg Training Examples				
	(6%, 3%)	(10%, 10%)	(15%, 10%)	(25%, 10%)	(100%, 10%)
Genetic Alg. (CFS)	21.3%	20.3%	22.5%	19.7%	16.3%
Neural Nets (BpNet)	9.7%	6.3%	4.7%	7.8%	4.8%
Decision Trees (C4.5)	9.7%	8.3%	1.3%	2.5%	1.6%
Decision Rules (AQ15)	22.8%	5.0%	4.8%	1.2%	0.0%
Decision Rules (AQ17-HCI)	4.8%	1.2%	0.0%	0.0%	0.0%

Table A. The average error rate of learned descriptions.

	Percentage of Pos & Neg Training Examples				
	(6%, 3%)	(10%, 10%)	(15%, 10%)	(25%, 10%)	(100%, 10%)
Genetic Alg. (CFS)	49	45	51	48	41
Neural Nets (BpNet)	35	26	12	22	12
Decision Trees (C4.5)	3.1	2.8	2.5	2 5	2.5
Decision Rules (AQ15)	2.6	2.2	2	1.6	1.6
Decision Rules (AQ17-HCI)	2.4	2.0	1.6	1.6	1.6

Table B. The dependence of the R-complexity on the number of training examples.

The *errors of omission* are represented by sparsely shaded areas. The *errors of commission* are represented by densely shaded areas (Figures 6, 7 & 8). The union of both types of errors represents the *error image* (Figure 8).

Figure 9 and Tables A & B summarize results of all experiments. Figure 9 and Table A show the average exact error rate, and Table B gives the average R-complexity of descriptions learned from different training sets. Pairs (a,b) in the top row of Tables A & B denote the percentage of positive and negative examples, respectively, used in experiments.

The results in Figure 9 and Table A represent the average exact error rate for all five concepts learned from two different subsets of training

examples (Pos1 & Neg1, Pos2 & Neg2, ...). In these experiments, target concepts were precisely defined and there was no noise in training data sets. Therefore, as results from AQ15 were used complete and consistent concept descriptions, rather than truncated descriptions (Bergadano, et al., 1990). For the same reason, as results from C4.5 were used unpruned decision trees. The average error rate (4.8% in the case of 100% positive and 10% negative training) of the BpNet-generated concepts was primarily due to an inadequate learning of concepts C1 and C4. The error rate of the CFS-generated descriptions was much higher then that of the other descriptions, and what is most surprizing, did not improve much with the growth of training sets. For AQ15-generated descriptions, the erro-r rate of descriptions learned from less than 100% positive examples was primarily due to the description of concept C1.

Decision trees generated by C4.5 produced some error even when 100% positive examples were given. The error may sometimes be reduced if the function for converting trees into rules is applied (this however, involves pruning a tree and simplifying rules).

The average R-complexity was determined over five concept descriptions learned for each of the 10 training sets. The R-complexity of the CFS- and BpNet-generated descriptions was derived by counting the estimated minimum number of rules needed to represent (to cover exactly) the image of the concepts in the diagrammatic representation.

The complexity of rules obtained by AQ17-HCI may differ from AQ15 due to new attributes' descriptions. With respect to the R-complexity, attribute generation process pays off when learned concepts are more complex and consist of larger number of rules and conditions inside

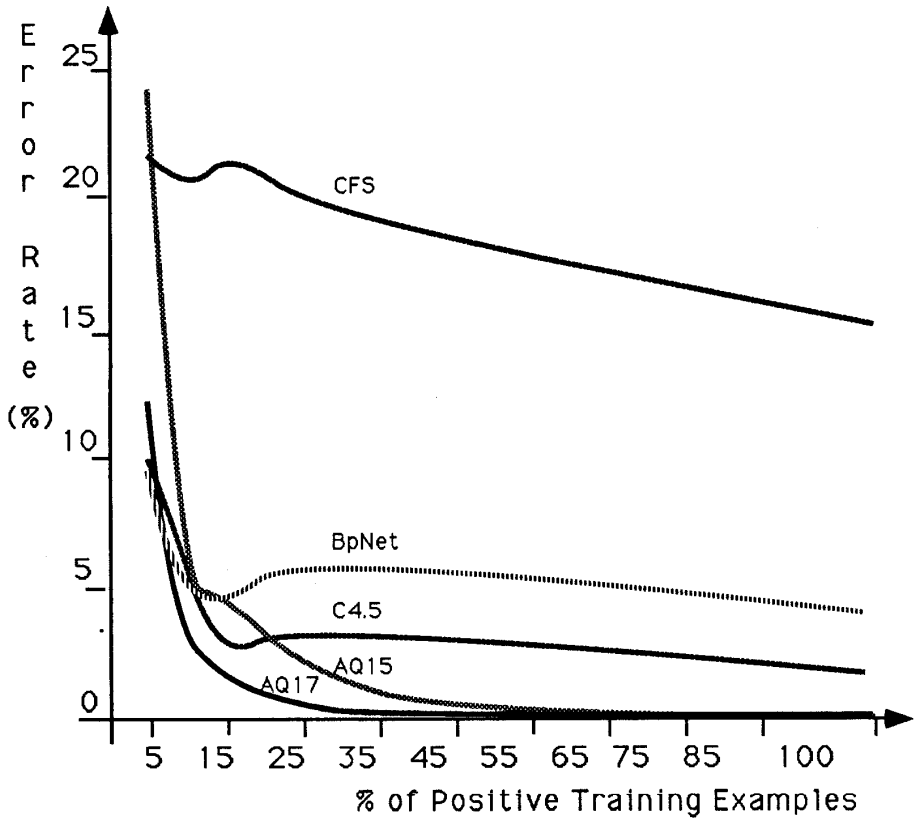


Figure 9. Learning curves for concepts from ROBOTS domain.

rules. In the experiments with concepts C1-C5, AQ17-HCI used mainly the *Description Space Reduction Transformation*, where some of the attributes and the attribute vales are removed from the description space.

6 Summary and Future Work

Five learning methods, C4.5, AQ15, AQ17-HCI, BpNet, and CFS were compared in terms of the average exact error rate and the average R-complexity of concept descriptions learned from the same sets of training data. The results showed that for the "cognitively oriented" target concepts used in the experiments, the symbolic methods, C4.5, AQ15, and AQ17-HCI, outperformed the subsymbolic methods, BpNet and CFS, both in terms of the average exact error rate and the R-complexity of the descriptions learned. One interesting finding is that increasing the number of training examples from 6%pos and 3%neg to 100%pos and 10%neg resulted in only a slight improvement of the performance of the CFS-generated descriptions (from 21.3% to 16.3%). Other interesting findings are that even with 100% positive examples the neural net, the genetic algorithm, and to a smaller degree the decision tree method did not learn the concept precisely. The CFS classifier system seems to be not well suited for the classification-type problems. To test further this finding, in the future research we plan to experiment also with another implementation of CFS and another genetic algorithm-based system.

The diagrammatic visualization proved to be a very useful method for illustrating learning processes. For symbolic learning systems there is a direct mapping between the description acquired by the system and the concept image in the diagram. For non-symbolic methods, the concept image is determined by mapping into the diagram the decisions suggested by the description for each possible instance. The so obtained concept image can then be used for re-representing the concept in the form of a DNF description (or logic-style rules). As knowledge encoded in a neural net or a population of classifiers is hard to comprehend, this way the diagrammatic visualization enables us to get an insight into what was learned by these methods. Because the size of the description space that can be represented in a diagram is limited by the size of the display, the effect of the technique is the strongest for relatively small domains (no more than 10-12 attributes). For larger problems one can create a hierarchy of diagrams.

As mentioned earlier, target concepts were generated by human subjects, and therefore the study favored methods that use symbolic representations, as such representations are more closely related to human representations. Studying how systems learn such human-generated concepts is important for applications where knowledge that needs to be acquired is in such "cognitively-oriented forms," and/or applications where the knowledge learned needs to be understandable by human experts.

There are problem domains in which these factors are not relevant. Therefore, to make a complete evaluation of the relative performance of these methods, the future research will investigate other problem types, for example, special technical problems, such as text-to-speech mapping (Dietterich 1990) and boolean DNF and parity functions (Wnek and Michalski, 1991). Such studies should give us an insight into the performance of different methods in learning any kind of concepts. Future research might also compare the performance of the methods in learning from noisy data or from inconsistent example sets, and in learning imprecisely defined concepts.

Acknowledgments

The authors thank George Tecuci and Jianping Zhang for comments on the paper.

This work was done in the GMU Center for Artificial Intelligence. Research of the Center is supported in part by the Defense Advanced Research Projects Agency under the grants administrated by the Office of Naval Research No. N00014-87-K-0874 and N00014-91-J-1854, in part by the Office of Naval Research under grants No. N00014-88-K-0397, No. N00014-88-K-0226, and No. N00014-91-J-1351, and in part by the National Science Foundation Grant No. IRI-9020226.

References

Bergadano, F., Matwin, S., Michalski, R.S., Zhang, J., "Learning Two-Tired Descriptions of Flexible Concepts: The POSEIDON System," accepted to *Machine Learning Journal,* 1991

Dietterich, T.G., Hild, H., Bakiri, G., "A Comparative Study of ID3 and Backpropagation for English Text-to-Speech Mapping," *Proceedings of the 7th International Conference on* Machine Learning, 1990.

Fisher, D. H., McKusick, K. B., "An Empirical Comparison of ID3 and Back-propagation," *Proceedings of IJCAI-89*, Detroit, MI, pp. 788-793, August 1989.

Kaufman, K.A., Michalski, R.S., and Schultz, A.C., "EMERALD1: An Integrated System of Machine Learning and Discovery Programs for Education and Research," George Mason U., *Reports of Machine Learning and Inference Laboratory*, No.MLI-89-12, 1989.

Michalski, R.S., and Larson, J.B., "Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of program AQ11, George Mason University, *Reports of Machine Learning and Inference Laboratory*, No. MLI-83-11, 1983.

Michalski, R., Mozetic, I., Hong, J., and Lavrac, N., "The AQ15 Inductive Learning System: An Overview and Experiments," George Mason University, *Reports of Machine Learning and Inference Laboratory*, No. MLI-86-6, 1986.

Michalski, R.S., "Pattern Recognition as Knowledge-Guided Computer Induction," Computer Science Dept., University of Illinois, Urbana, 1978.

Mooney, R., Shavlik, J., Towell, G., Gove, A., "An Experimental Comparison of Symbolic and Connectionist Learning Algorithms," *Proceedings of IJCAI-89*, Detroit, MI, pp. 775-780, 1989.

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

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

Rendell, L. A., Cho, H. H., Seshu, R., "Improving the Design of Similarity-Based Rule -Learning Systems," *International Journal of Expert Systems,* 2, 1, pp. 97-133, 1989.

Rendell, L. and Seshu, R., "Learning Hard Concepts Through Constructive Induction: Framework and Rationale," *Computational Intelligence,* **6**, pp. 247-270, 1990.

Riolo, R.L., "CFS-C: A Package of Domain Independent Subroutines for Implementing Classifier Systems in Arbitrary, User-Defined Environments," Logic of Computers Group, Division of Computer Science and Engineering, University of Michigan, 1988.

Rumelhart, D. E., Hinton, G. E., Williams, J. R., "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing, Vol.1*, Rumelhart & McClelland (eds.), pp. 318-362, 1988.

Sejnowski, T.J., and Rosenberg, C.R., "Parallel networks that learn to pronounce English text. *Complex Systems,* Vol. 1, pp. 145-168, 1987.

Utgoff, P. E., "ID5: An Incremental ID3," *Proceedings of the 5th International Conference on* Machine Learning, Ann Arbor, MI, 1988.

Weiss, S. M., and Kapouleas, I., "An Empirical Comparison of Pattern Recognition of Neural Nets, and Machine Learning Classification Methods," *Proceedings of IJCAI-89*, Detroit, MI, pp. 781-787, 1989.

Wnek, J., and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments," *Proceedings of the IJCAI-91 Workshop on Evaluating and Changing Representations,* K.Morik, F.Bergadano, W.Buntine (Eds.), pp. 13-22, Sydney, Australia, 1991.

Wnek, J., and Michalski, R.S., "Diagrammatic Visualization of Learning Processes," George Mason University, *Reports of Machine Learning and Inference Laboratory*, No. MLI-91, 1991.