Stochastic search in inductive concept learning

Dunja Mladenić, Darko Zupanič
Marko Grobelnik and Nada Lavrač
Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Phone: (+38)(61) 159 199, Fax: (+38)(61) 161 029
nada.lavrac@ijs.ac.mail.yu

Function: learning
Knowledge: if-then rules

Abstract

Concept learning can be viewed as search of the space of concept descriptions. The hypothesis language determines the search space. In standard inductive learning algorithms, the structure of the search space is determined by generalization/specialization operators. Algorithms perform locally optimal search by using a hill-climbing and/or a beam-search strategy. To overcome this limitation, concept learning can be viewed as stochastic search of the space of concept descriptions. The proposed stochastic search method is based on simulated annealing which is known as a successful means for solving combinatorial optimization problems. The stochastic search method, implemented in a rule learning system **ATRIS**, is based on a compact and efficient representation of the problem and the appropriate operators for structuring the search space. Furthermore, by heuristic pruning of the search space, the method enables also handling of imperfect data. The paper introduces the stochastic search method, describes the **ATRIS** learning algorithm and gives results of the experiments.

1 Introduction

Concept learning can be viewed as search of the space of concept descriptions [11]. The choice of the description language of concepts L_C determines the search space, called the *hypothesis space*. Depending on the choice of the hypothesis language, *inductive learning systems* can learn either attribute descriptions or first-order relational descriptions. The topic of this paper is only *attribute-value learning*, in which training examples and induced hypotheses are described with their global features in terms of a fixed collection of attributes.

In inductive learning systems, the structure of the search space is determined by generalization/specialization operators. Search can proceed bottom-up or top-down. In top-down search, based on specialization operators, various search strategies can be applied. Usually, only locally optimal search is performed by using a hill-climbing and/or a beam-search strategy. For example, a tree induction system ASSISTANT [3] performs hill-climbing, and rule induction systems AQ15 [9] and CN2 [5] perform beam-search. Their success in dealing with real-world problems lies also in the choice of *heuristics* used to guide the search and to stop the construction of trees/rules.

The goal of this research is twofold. First, in order to overcome the limitation of locally optimal search, the paper proposes the use of a stochastic search strategy. Furthermore, in order to cope with imperfect data, appropriate heuristics are used to stop the search of rules. The search method is based on a variant of simulated annealing [12], named Markovian neural network [7], which has proved successful in solving combinatorial optimization problems. On the other hand, to handle noise, the selected stochastic search strategy is based on the heuristics used to minimize the expected classification error of induced rules by applying the m-estimate [2].

The paper describes the stochastic search method implemented in a rule induction system ATRIS. Section 2 introduces the concept learning problem and the VL1-like concept description language. Section 3 describes the method of stochastic search based on Markovian neural networks. Section 4 describes the ATRIS stochastic rule learning system. Section 5 gives the results of the experiments.

2 Problem definition

In the classical task of learning from examples, the *target concept* C is a subset of the given *instance space* of objects, labeled $\oplus$ if they are elements of the target concept (*positive examples*), and $\ominus$ (*negative examples*) otherwise. Instead of being classified into one of the two *classes* ($\oplus$ or $\ominus$), an object can be classified into one of the mutually exclusive classes C_k, $k = 1, \ldots, N$. The learning task is defined as follows: Given is the description language of examples L_E, the hypothesis language L_C, the coverage relation between L_C and L_E and the quality criterion defined on strings of L_C. The task is to find a *concept description*, expressible in L_C, satisfying the quality criterion.

Usually, the quality criterion requires the induced hypothesis to be complete and consistent with respect to the examples, i.e., that it covers all positive examples and no negative example. When learning from imperfect data, the consistency and completeness criteria need to be relaxed in order to avoid overly specific hypotheses.

Let L_E be the language of attribute value tuples, and L_C a VL1-like language of *if-then* rules. In a VL1-like hypothesis language, a *selector* relates an attribute to a value or a disjunction (so-called *internal disjunction*) of values. Illustrated by examples from a simple robots domain (see also Sections 4 and 5), $[Holding = balloon \vee flag]$ is a selector. A conjunction of selectors forms a *complex*. A hypothesis is a set of *if-then* rules of the form *Class* **if** *Complex*, for example,

robot_friendly **if** $[Is\_smiling = yes] \bigwedge [Holding = balloon \vee flag]$.

3 Stochastic search method

Simulated annealing is a method for solving large scale (possibly NP complete) optimization problems. For a selected evaluation function, which is to be minimized, the method performs stochastic search in a discrete state space; the search space is typically factorially large so that it cannot be explored exhaustively. This section presents a method of stochastic search based on Markovian neural networks [7] which perform 'Markovian' simulated annealing.

An optimization problem can be formulated as follows [7]: Let $\mathcal{S}$ be a state space consisting of a finite set of states s_i, and $f : \mathcal{S} \rightarrow \Re$ an evaluation function called *energy*. The task is to find such s_{min} that $\forall s_i \in \mathcal{S} : f(s_{min}) \leq f(s_i)$, i.e., to find the best state s_{min} with respect to f.

In addition, there is a *transition* operator ρ which determines a set $\mathcal{R}_{s_i}$ consisting of states which can be reached from the given state s_i with exactly one transition. If s_{cur} denotes the given (current) state, then

$$\mathcal{R}_{s_{cur}} = \{s_k;\ s_k = \rho(s_{cur})\}. \tag{1}$$

Note that $s_{cur} \notin \mathcal{R}_{s_{cur}}$. Let $\mathcal{R}^{<}_{s_{cur}}$ be a set of those states from $\mathcal{R}_{s_{cur}}$ with a lower value of the evaluation function tha s_{cur}:

$$\mathcal{R}^{<}_{s_{cur}} = \{s_k;\ s_k \in \mathcal{R}_{s_{cur}} \wedge f(s_k) < f(s_{cur})\}.$$

Furthermore, we define a set of *successors* of state s_{cur}, denoted by $\mathcal{S}_{s_{cur}}$, as a subset of $\mathcal{R}_{s_{cur}}$ (i.e., $\mathcal{S}_{s_{cur}} \subseteq \mathcal{R}_{s_{cur}} \bigcup \{s_{cur}\}$) as follows:

$$\mathcal{S}_{s_{cur}} = \begin{cases} \mathcal{R}^{<}_{s_{cur}} & if \quad \mathcal{R}^{<}_{s_{cur}} \neq \emptyset \\ \mathcal{R}_{s_{cur}} \bigcup \{s_{cur}\} & otherwise \end{cases} \tag{2}$$

In other words, if there exists a state $\rho(s_{cur})$ with a lower value of f than that of s_{cur}, the set of successors consists of states with lower value of f. Otherwise, if a transition does not lead to any 'better' state, the state of successors consists of all the transition states and the current state itself.

Let $\mathcal{S}_{s_i}$ be the set of successors of a given state s_i, and $T > 0$, a parameter called *temperature*. Let the probability $p^{(T)}(s_i \mapsto s_j)$ of a transition from state s_i with energy $f(s_i)$ into s_j with energy $f(s_j)$, denoted by $p_{i,j}^{(T)}$, be defined as follows:

$$p_{i,j}^{(T)} = \frac{1}{N_i^{(T)}} \exp\left(\frac{f(s_i) - f(s_j)}{T}\right) \tag{3}$$

where $N_i^{(T)}$ is a normalization coefficient assuring that $p_{ij}^{(T)} \in [0, 1]$:

$$N_i^{(T)} = \sum_{s_m \in \mathcal{S}_{s_i}} \exp\left(\frac{f(s_i) - f(s_m)}{T}\right)$$

What follows is a stochastic search algorithm described in the above terms.

Given:
$\mathcal{S}$ ~ set of all possible states
$\rho : \mathcal{S} \to \mathcal{S}$ ~ transition operator
$\mathit{fStopCriterion} : \mathcal{S} \to \{\mathit{false}, \mathit{true}\}$ ~ function which decides about terminating **while** loop – it could depend also on other arguments (e.g., time, temperature, no. of iterations, etc.)
$\mathit{fTemp} : \Re \to \Re$ ~ temperature change on each step – the function is problem dependent (e.g., $\mathit{fTemp}(T) \leftarrow T \times 0.99$)

Input:
$s_{ini} \in \mathcal{S}$ ~ initial state
$T_{ini} \in \Re$ ~ initial temperature – a problem dependent parameter

Output:
$s_{min} \in \mathcal{S}$ ~ best solution found – the one with the lowest energy

Algorithm:
var $T \in \Re$ ~ temperature
var $s_{cur} \in \mathcal{S}$ ~ current state
var $\mathcal{S}_{s_{cur}} \subseteq \mathcal{S}$ ~ set of succesors of the current state s_{cur}
var $s_{next} \in \mathcal{S}$ ~ next state – the selected successor of the current state
var $s_{min} \in \mathcal{S}$ ~ best solution

$T \leftarrow T_{ini}$
$s_{cur} \leftarrow s_{ini}$
$s_{min} \leftarrow s_{cur}$

while not $fStopCriterion(s_{min})$ **do** {find 'best' state}
  $\mathcal{S}_{s_{cur}} \leftarrow$ computed by formula (2) according to s_{cur} and ρ
  $s_{next} \leftarrow$ randomly selected state from set $\mathcal{S}_{s_{cur}}$ according to
    transition probabilities $p_{cur,j}^{(T)}$ (formula (3))
  if $f(s_{next}) < f(s_{min})$ **then** $s_{min} \leftarrow s_{next}$ **endif**
  $s_{cur} \leftarrow s_{next}$
  $T \leftarrow fTemp(T)$
endwhile
output s_{min}

Temperature T is lowered in every iteration; this corresponds to simulated annealing which, in the limit, aims to reach the state of minimum energy. To do so, a successor of the current state is randomly selected; however, the algorithm assures that successors with a lower value of f have a higher probability of selection.

Kovačič [7] has implemented the algorithm as a neural network. He has given a convergence proof for the Markovian neural network that asynchroneously updates its neurons' states. Furthermore, he has shown that an asynchronous Markovian neural network is equivalent to an inhomogenous Markov chain. For our purpose, the network implementation is not needed; rather, we borrow his ideas and adapt them to the inductive rule learning framework.

4 ATRIS: A stochastic rule induction algorithm

This section describes the ATRIS rule induction algorithm: its covering algorithm, the internal representation of complexes, pre-processing of training examples, the stochastic search mechanism, heuristics for stopping rule construction, and post-processing of induced rules.

4.1 The basic covering algorithm

The ATRIS learning algorithm consists of three main steps: *pre-processing* of the training set, *construction* of a hypothesis and its *post-processing*:

pre-process *Training_set*
initialize *Hypothesis* {*Hypo* $\leftarrow$ empty list}
for each *Class* **do**
  repeat {covering}
    initialize *Complex* {*Complex* $\leftarrow$ [ ]}
    while not necessity stopping criterion **do** {finding 'best' complex}
      specialize/generalize *Complex* for *Class*
    endwhile

append (*Class* **if** *Complex*) to *Hypothesis*
remove covered positive examples from *Training_set*
until sufficiency stopping criterion
endfor
post-process *Hypothesis*

The main step is the *construction* of a hypothesis. It consists of two loops, referred to as *covering* and *finding 'best' complex.* Finding of the best complex starts with an empty complex and continues by heuristically applying specialization/generalization operators. The two loops are controlled by two *stopping criteria*, used to decide whether to stop rule construction. In domains with perfect (exact, non-noisy) data, the *sufficiency criterion* requires completeness (coverage of all positive examples) and the *necessity criterion* requires consistency (no covered negative example). In domains with imperfect data, the two criteria are implemented as heuristics for evaluating the expected classification error, aimed at avoiding overly specific hypotheses.

Pre-processing, search of the 'best' complex, stopping criteria and post-processing will be further elaborated in the sections that follow.

4.2 Pre-processing

Suppose that the learning task is to find a description of friendly and unfriendly robots [15] from examples of robots described by six attributes: *A1 - Head_shape* $\in$ {*round, square, octagon*}, *A2 - Body_shape* $\in$ {*round, square, octagon*}, *A3 - Is_smiling* $\in$ {*no, yes*}, *A4 - Holding* $\in$ {*sword, balloon, flag*}, *A5 - Jacket_color* $\in$ {*red, yellow, green, blue*}, and *A6 - Has_tie* $\in$ {*no, yes*}. A solution of this problem is given in Section 5. In the internal bit representation, similar to the one used by De Jongh and Spears [6], an example complex [*Is_smiling = yes*] $\bigwedge$ [*Holding = balloon* $\vee$ *flag*] is represented as follows:

$$\begin{array}{cccccc} A1 & A2 & A3 & A4 & A5 & A6 \\ 000 & 000 & 01 & 011 & 0000 & 00 \end{array} \tag{4}$$

In addition, ATRIS can also use background knowledge. This is done in *pre-processing.* For example, the built-in background knowledge checks for the equality of attribute values for pairs of attributes of the same type (i.e., attributes with the same set of values). In our domain, ATRIS will introduce two new attributes $A1 = A2$ and $A3 = A6$, which will be named $A7$ and $A8$, respectively. Using this idea, originating from the LINUS relation learning system [8], an extended set of tuples is generated and used in learning. For example, let a tuple (an example of friendly robots) be as follows:

$$\begin{array}{cccccc} A1 & A2 & A3 & A4 & A5 & A6 \\ square & square & yes & balloon & red & yes \end{array}$$

The corresponding extended tuple is then the following:

A1	*A2*	*A3*	*A4*	*A5*	*A6*	*A7*	*A8*
square	*square*	*yes*	*balloon*	*red*	*yes*	*true*	*true*

4.3 Search of the 'best' complex

According to the terminology introduced in Section 3, the covering algorithm from Section 4.1 is further elaborated. In short, searching for the 'best' complex starts with an empty complex and continues by stochastically applying the specialization/generalization operators to the current complex, until the necessity stopping criterion is satisfied.

In formal terms, let states s in the state space $\mathcal{S}$ be individual VL1 complexes. Let the evaluation function f be the *expected classificiation error*, described in Section 4.4, computed by the Bayesian m-estimate.

Let s and s' be two complexes. A *transition operator* ρ maps a complex s to a set of complexes $R_s = \{s';\ s' = \rho(s)\}$. An individual complex is represented by a bit string. For example, let s_{cur} be the extension of the current complex given in equation(4), which is internally represented as follows:

000 000 01 011 0000 00 01 01

Then $R_{s_{cur}} = \{s';\ s' = \rho(s_{cur})\}$ is the set of all different strings of 21 bits such that exactly one of the bits in a string changes its value. In this way, four operators are implemented; two specialization operators: adding a selector to the complex and removing a value from an internal disjunction, and symetrically, two generalization operators: deleting a selector and adding a value to an internal disjunction. Such an approach is non-standard; it makes sense only due to a number of iterations performed by the stochastic search algorithm which gradually refines the current hypothesis.

To construct the ATRIS algorithm, both the initialization of a *Complex* and the **while** loop of the covering algorithm from Section 4.1 need to be substituted by the stochastic search algorithm from Section 3. The obtained algorithm performs a-kind-of simulated annealing aiming to reach the state of minimum energy, i.e., the complex with a minimal expected classification error.

4.4 Heuristic estimates of the quality of rules

Several probability estimates can be applied to evaluate the quality of a rule, the *relative frequency* being the most common one. For an individual rule, it is computed as $p(C) = \frac{n_C}{N}$, where n_C is the number of positive examples of class C and N the number of all examples covered by the rule. This is unreliable when estimating probabilities from a small training set. To alleviate this problem, the *Laplace estimate* can be used. This estimate, however, relies on the assumption that the prior probability of each class is uniformly distributed [10], which is rarely true in practice. To avoid this assumption, Cestnik [2] proposed the m-estimate which was successfully used in the 'naive' Bayesian classifier [2] and in tree pruning [4].

What follows is a formula for computing the *expected classification accuracy* $p(C)$ using the m-estimate:

$$p(C) = \frac{n_C + m \times p_a(C)}{N + m} \tag{5}$$

This estimate takes into account prior probabilities of classes. The prior probability $p_a(C)$ can be estimated by the relative frequency of positive examples in the initial training set $\frac{n_C}{N}$. The parameter m can be set subjectively; it expresses our confidence in the experimental evidence (training examples). The parameter allows for adapting to the learning domain. For example, the actual value of m should be set according to the amount of noise in the examples (larger m for more noise). As m grows towards infinity, the m-estimate approaches the prior probability of the corresponding class.

The m-estimate is a general probability estimate. By appropriately setting its two parameters m and $p_a(\oplus)$, the other two estimates can be obtained: relative frequency $p(C) = \frac{n_C}{N}$ (for $m = 0$), and Laplace estimate $p(C) = \frac{n_C+1}{N+2}$ (for $m = 2$, $p_a(C) = \frac{1}{2}$).

Since the learning problem in ATRIS is stated as a minimization problem, a variant of expected classification accuracy $p(C)$ is used. The function f to be minimized is the *expected classification error*, computed as $1 - p(C)$. In ATRIS, this same heuristic implements also the sufficiency and the necessity stopping criteria of the covering algorithm in Section 4.1. The actual probability estimate used in the expected classification error heuristic is the m-estimate.

4.5 Post-processing

The stochastic search algorithm typically finds a good 'global' solution, which can still be locally optimized.

Post-processing mechanism in ATRIS refines each induced hypothesis by a simple local optimization. This is performed on each individual complex (internally represented as a bit string) as described in the algorithm below:

```
for each Hypothesis do
    for each Complex do
        for each Bit in Complex do
            change value of Bit                              {Bit ← not Bit}
            compute the expected classification accuracy on the set
              of all complexes of a given class
            if the expected classification accuracy increases then
                keep the new value of Bit
            else
                change Bit to the old value
```

endif
endfor
endfor
endfor

The way of post-processing hypotheses is non-standard; usually, in domains with imperfect data, post-processing is performed by removing selectors from a rule and by removing rules, which is referred to as the *reduced error pruning* [13, 1]. In further work we will verify whether additional improvements in classification accuracy can be obtained by using both local optimization and reduced error pruning.

5 Experiments and results

The algorithm was tested in three toy robot domains, referred to as $Monk1$, $Monk2$ and $Monk3$, which were used to compare a number of different inductive learning algorithms [15, 14]. The learning task is to find a description of friendly and unfriendly robots from examples of robots described by six attributes *Head_shape, Body_shape, Is_smiling, Holding, Jacket_color*, and *Has_tie* (see Section 4.2).

The experimental method was the following. In all domains, the temperature parameter T was initially set to 1 and was geometrically reduced, allowing for 600 iterations of stochastic search (function $fTemp(T) \leftarrow 0.996 \times T$). Different values of the m parameter were tested, ranging from 0.1 to 64. For a fixed m, results are averaged over four runs. Results are given for the best m.

Results of applying different learning systems [14], together with the results of our experiments, are given in Table 1.

In the $Monk1$ domain, from 432 possible examples 124 are given as the training set. ATRIS has achieved a 100% accuracy, inducing exact rules:

$$
\begin{array}{lll}
robot\_friendly & \textbf{if} & [Head\_shape = Body\_shape] \\
robot\_friendly & \textbf{if} & [Jacket\_color = red] \\
robot\_unfriendly & \textbf{if} & [Jacket\_color = yellow \vee green \vee blue] \bigwedge \\
 & & [Head\_shape \neq Body\_shape]
\end{array}
$$

Exact rules were induced also by some other systems, but typically in a less compact form. In ATRIS, the equality of two attributes of the same type is built-in; as a consequence, it was simple to induce this compact description.

The $Monk2$ domain has 169 training examples. The exact rule states that *exactly two of the six attributes have their first value*. This domain is very difficult for standard learning systems. Our interpretation is that relatively good results of ATRIS are due to its 'global' stochastic search. For each of the two classes, the most compact rules consist of nine complexes.

The $Monk3$ domain is noisy. The training set consists of 122 randomly selected examples; among them there are 5% misclassifications. Our results are among the better. However, some of the systems reached a 100% classification accuracy which was not achieved by ATRIS.

	Monk1	Monk2	Monk3
AQ17-DCI	100	99.8	
AQ17-HCI	100	93.1	100
AQ17-FCLS		92.6	97.2
AQ14-NT			100
AQ15-GA			100
Assistant Professional	100	81.25	100
mFOIL	100	69.2	100
ID5R	81.7	61.8	
IDL	97.2	66.2	
ID5R-hat	90.3	65.7	
TDIDT	75.7	66.7	
ID3	98.6	67.9	94.4
ID3, no windowing	83.2	69.1	95.6
ID5R	79.7	69.2	95.2
AQR	95.9	97.7	87.0
CN2	100	69.0	89.1
CLASSWEB	59.49	67.36	65.28
PRISM	86.3	72.2	90.3
Backpropagation	91.7	100	87.7
Cascade Correlation	100	100	97.2
ATRIS	100	93.7	97.2

Table 1: Results of the system ATRIS are compared with the results of different learning systems in three toy robot domains. All of the systems are attribute-value rule/tree learning systems, except for: CLASSWEB (a clustering algorithm), Backpropagation and Cascade Correlation (neural networks), and mFOIL (inductive logic programming). The table, except for the last row, is reproduced from [14].

6 Discussion

The paper presents a new search method to be applied in inductive rule learning. The method is based on a variant of simulated annealing, named Markovian neural network, which has proved successful in solving combinatorial optimization problems. Instead of hill-climbing and/or beam-search, the method allows for 'global' search, leading to sub-optimal solutions.

The method was implemented in a rule induction system ATRIS, aimed at overcoming the limitations of locally optimal search. Another important feature is the possibility of handling imperfect data. This is achieved by using the heuristic which minimizes the expected classification error of induced rules.

Initial results indicate that the method is comparable to other learning systems. By further experimentation with slower annealing and by different settings of the m parameter, better results could still be achieved. Next, exhaustive experimentation on real-world domains is still to be performed. Nevertheless, initial results are very promising. Many improvements of the method can already be envisioned. Instead of applying the covering algorithm, more sophisticated search could be performed, building more than one complex at a time. Post-processing can also be improved. In addition to the existing local optimization, reduced error pruning could be performed; in this way, selectors and complexes could be eliminated, potentially causing better classification accuracy on unseen cases. On the other hand, due to the 'globally optimal seach' which is not exhausted at the local level, a more elaborate local optimization could also lead to improved accuracy.

References

[1] Brunk, C.A. and Pazzani, M.J. (1991) An investigation of noise-tolerant relation concept learning algorithms. Eighth International Workshop on Machine Learning. Evanston, IL: Morgan Kaufmann.

[2] Cestnik, B. (1990) Estimating probabilities: A crucial task in machine learning. Proc. of ECAI 90, Stockholm, Sweden.

[3] Cestnik, B., Kononenko, I. and Bratko, I. (1987) ASSISTANT 86: A knowledge elicitation tool for sophisticated users. In: Bratko, I. and Lavrač, N. (eds.) Progress in machine learning. Wilmslow: Sigma Press.

[4] Cestnik, B. and Bratko, I. (1991) On estimating probabilities in tree pruning. Fifth European Working Session on Learning, EWSL 91. Porto, Portugal: Springer-Verlag.

[5] Clark, P. and Boswell, R. (1991) Rule induction with CN2: Some recent improvements. Proc. of EWSL 91. Porto, Portugal: Springer-Verlag.

[6] De Jongh , K.A. and Spears, W.M. (1991) Learning concept classification rules using genetic algorithms. Proc. IJCAI-91. Australia: Sydney. Morgan Kaufmann.

[7] Kovačič, M. (1991) Markovian neural networks. Biological Cybernetics 64, 337-342. Springer-Verlag.

[8] Lavrač, N., Džeroski, S. and Grobelnik, M. (1991) Learning nonrecursive definitions of relations with LINUS. Proc. of EWSL 91. Porto, Portugal: Springer-Verlag.

[9] Michalski, R.S., Mozetič, I., Hong, J. and Lavrač. N. (1986) The multipurpose incremental learning system AQ15 and its testing application to three medical domains. Proc. of AAAI-86, 1041-1045. PA: Philadelphia. Morgan Kaufmann.

[10] Mingers, J. (1989) An empirical comparison of pruning methods for decision tree induction. Machine Learning 4 (2). Kluwer Academic Publishers.

[11] Mitchell, T. (1982) Generalization as search. Artificial Intelligence 18, 203-226.

[12] Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (19..) Combinatorial minimization: Method of simulated annealing. In Numerical Recipes: The art of scientific computing, 326-334. Cambridge University Press.

[13] Quinlan, J.R. (1987) Simplifying decision trees. International Journal of Man-Machine Studies, 27, 221-234.

[14] Thrun, S.B. et al. (1991) The MONK's problems: A performance comparison of different learning algorithms. Technical Report. Carnegie Mellon University. 1991.

[15] Wnek, J., Sarma, J., Wahab, A., and Michalski, R.S. (1991) Comparison learning paradigms via diagramatic visualisation: A case study in single concept learning using symbolic, neural net and genetic algorithm methods. Technical Report. George Mason University, Computer Science Department, 1990.