ClimBS: Searching the Bias Space

Foster John Provost
Intelligent Systems Laboratory
Department of Computer Science
University of Pittsburgh

Abstract

In recent years, researchers have begun to study systems that address the selection of inductive bias explicitly. This paper briefly reviews the literature on such systems and then introduces a model of inductive bias selection as state space search, which is instantiated in a testbed system with biases as states and bias transformation operators used to move from state to state. The testbed allows a system developer to address different dimensions in the bias space and different policies for bias selection by adding the appropriate operators. The ClimBS system has been developed in this testbed as one policy for bias selection modeled after manual bias selection strategies; the system's performance is measured on several domains from the UCI repository and on a synthetic domain. Finally, the paper provides a summary of experiments designed to analyze empirically the system's performance.

1 Introduction

In order for an inductive learning program to define concepts that generalize beyond its training examples, it must incorporate inductive bias—assumptions that lead it to prefer certain inductive steps over others. Mitchell [1] defined bias as "any basis for choosing one generalization over another other than strict consistency with the observed training instances." Systems bias their learning in many ways, including using restricted description languages, heuristics to search the hypothesis space, and domain knowledge to guide the search. The strength of a bias is the fraction of hypotheses considered by the learner with that bias relative to all possible hypotheses [2]. The basic problem is that a strong bias is necessary for a learning system to run efficiently, but picking a strong bias that will still allow the concept to be learned is seldom easy. The term inductive policy has been used to describe the strategy used to select a particular inductive bias (implementation) based on the underlying assumptions of the domain [3]. This paper presents a testbed system for studying and implementing different policies for bias selection, and one system developed within this testbed.

Selection of the bias of learning systems historically has been done by the systems' users. In many learning systems there is some element of the inductive process that could be characterized as dynamic bias selection. For example, Samuel's checkers program [4] performed term selection, substituting terms from an initially specified set during learning. More recently, systems have begun to address the issue of automatic bias selection as such, modifying the bias of existing learning systems. I will briefly review these; a more comprehensive discussion can be found in [5].

Utgoff [2] addresses the bias adjustment problem of adding new terms to the concept description language. The bias is shifted when the current hypothesis space can be shown not to include a concept that is consistent with the observed training examples. The STABB system introduces two heuristics for adding new terms. Recently, the issues involved in such constructive induction have received a good deal of attention; an overview can be found in [6].

The VBMS system [7] attempts to choose a satisfactory learning algorithm based on aspects of the domain, such as the number of features and the number of examples. VBMS clusters algorithms based on their past performance; when it is faced with a new problem, the system selects the "best" algorithm and tries it. If this algorithm does not perform satisfactorily, VBMS tries the next best and so on. When VBMS runs out of "good" candidates it picks algorithms randomly and tries them until all algorithms are exhausted (updating its problem/algorithm performance statistics).

The Competitive Relation Learner (CRL) [8] is a generalized recursive splitting algorithm that includes multiple decomposition strategies, multiple learning strategies, multiple decomposition evaluation functions, as well as other parameters. CRL is combined with a bias optimization program called the Induce and Select Optimizer (ISO). ISO first probes randomly in the bias space, each probe evaluated by forming CRL's hypothesis with the given bias. ISO then attempts to describe a
surface over the bias space and use this surface to guide selection of the next bias.

Gordon [9] uses actively requested examples to test explicit "biasing assumptions" (irrelevance beliefs) and subsequently to select appropriate "bias adjustments" (mask and unmask terms). Spears and Gordon have explored the use of genetic algorithms (GAs) to produce a multistrategy concept learner which can adaptively select from among alternative learning strategies. In [10], they consider adding two alternative strategies: (i) dropping a feature from a disjunct, and (ii) adding a disjunct to the current classification rule. They also begin to explore the effect of interference among biases.

Russell [11] advocated a framework for building a theory of induction: (i) construct the space of possible classes of higher level regularities, (ii) search the space for interesting classes, (iii) analyze the results of the search, (iv) apply the results. The space of higher order regularities comprises those that can be represented in formal logic. Russell and Grosof have continued this work, looking at inductive bias from the perspective of formal logic and non-monotonic reasoning (see [12], [13]). They consider prior knowledge and learned knowledge as biases for a learning system. Retractable beliefs are represented as defaults in the system, which are prioritized to resolve conflicts. Along similar lines, desjardins [14] studies the use of probabilistic knowledge to choose feature sets.

This paper takes advantage of the SBS model, which views inductive bias selection as state space search, and the SBS Testbed which instantiates the model with biases as states and bias transformation operators used to move from state to state. The Testbed allows a system developer to address different dimensions in the bias space and policies for bias selection by adding the appropriate operators. By viewing bias selection from this perspective, heuristic strategies for bias space navigation can be enumerated and studied.

There are many dimensions in the bias space of an inductive learner that may be important for a bias selection system to consider, for example: the form of the concept description language (CDL), the content of the CDL (e.g., what terms are used), heuristic search parameters, etc. The ClimBS system, implemented within the SBS Testbed, performs a hill climbing search in the bias space while constructing its concept description as it searches the bias space.

2 The SBS model and the SBS Testbed

The SBS model (Searching the Bias Space) was developed to facilitate both the study of heuristic strategies for bias selection and the development of systems that automatically adjust the bias of inductive learning systems. The SBS model views bias selection as search, and is described in detail in [5]. This paper deals with a particular instantiation of the model on which the SBS Testbed system is based: in the bias space, the states are individual biases, sets of assumptions restricting and ordering the hypothesis space; bias transformation operators are used to move from one bias to the next.

In the SBS Testbed, the underlying learning system represents its bias as a partial domain model (PDM), and search operators change the bias by modifying the PDM. The PDM can be viewed as a set of hooks to the learning system (e.g., algorithm parameters, goodness criteria, the set of attributes used to describe the domain, a collection of domain knowledge used to restrict the hypothesis space, etc.). A Testbed system will move from bias to bias, running the learning system with each. The goal is to induce a concept description that satisfies some performance criteria.

The Testbed is a second-order system built around the MC-RL system, a multiclass version of the RL4 rule learning system [15] that uses a beam search to explore the rule space. MC-RL uses a PDM to (partially) specify its inductive bias; the Testbed allows the incorporation of bias transformation operators to modify the PDM and thereby adjust MC-RL's bias. The ClimBS system, described below, was implemented in the Testbed. The concluding section briefly discusses other policies.

3 Heuristic strategies for bias space search

By framing bias selection as a search problem, several strategies emerge as search heuristics. First we can impose structure on the bias space. Structuring the bias space means providing navigational guidelines that will allow a system to conduct an orderly search. In the SBS model, one can represent strategies for structuring the bias space by the choice of bias transformation operators, operator preconditions, and the strategies for applying the operators.

One strategy can be called strong-to-weak: bias space search starts at a relatively strong bias and changes the bias by iteratively applying bias transformation operators that progressively weaken the bias. Learning with a strong bias can take less time (search nodes, below) than learning with a weaker bias. The idea behind this strategy is that the knowledge gained quickly with a strong bias can be quite useful to the bias space search system. The strong bias may not be sufficient to learn the entire concept description; however it may learn part of the theory that can be used to guide and restrict subsequent search (cf., [2], [16]).

Additional structure can be given to the bias space by the system designer based on preconceived notions of which operators are more likely to be relevant (or along which dimensions the bias is more certain), or based on efficiency considerations; the details of such considerations are beyond the scope of this paper.
A second type of strategy deals with transferring knowledge from one bias to the next. Probably the most obvious strategy is to use previously acquired knowledge to guide the bias space search. The results of learning with the previous biases can affect bias space navigation by indicating what biases to try next. In addition, a bias space search system can construct its concept descriptions incrementally as it searches the bias space (cf. systems that construct their concept descriptions incrementally across example sets).

Besides using the knowledge gained with previous biases to construct the theory incrementally and to guide the bias space search, such knowledge can be utilized during the search with a given bias. Restrictions on the hypothesis space can be performed based on the elimination of hypotheses that can be shown not to be satisfactory with respect to the current bias or based on previously learned knowledge and heuristic methods. A simple heuristic technique (used in the implemented system, see below) is to allow the learner to consider only those hypotheses that cover an example not covered by the current concept description. Such techniques are beyond the scope of this paper, but are addressed in [17].

4 The ClimBS System -- Hill Climbing in the Bias Space

The ClimBS system, built in the Testbed, performs a search similar to hill climbing in a multidimensional bias space. ClimBS learns its concept description as it searches the bias space and uses a simple heuristic to direct the search with a given bias to those hypotheses (rules) that address incompleteness in the current description.

ClimBS' user specifies an initial bias in the form of a partial domain model (PDM), and bias transformation operators that operate on the PDM. ClimBS forms candidate biases by applying the transformation operators to the current bias; ClimBS then runs the learning system with each bias, collecting the concept descriptions learned with each. The biases are then compared based on the performance of the descriptions using a performance system and an evaluation function specified by the domain developer. The current bias is set to be the best of these biases and the process iterates until the learned description satisfies some stopping criteria. The performance system currently used is a simple inference engine (no chaining) that allows the user to specify strategies to resolve the conflicts that occur when a set of rules predicts that an example belongs to more than one concept. The defaults are to have the rules vote and to use predictive accuracy to compare rule sets. ClimBS' basic algorithm is described in Figure 1. The search differs from strict hill climbing in that it will continue to search by random walk if no operator application makes an improvement; and the operators have preconditions, structuring the bias space.

The basic algorithm has some alternatives whose analysis is beyond the scope of this paper, but deserve a brief mention. ClimBS can either compile its description monotonically (default) or use a greedy strategy to find the best subset of the augmented concept description. In addition, the default method of operation is to leverage the currently held concept description by using the simple inductive strengthening heuristic: a hypothesis should be considered only if it covers an example not covered by the concept description constructed with the previous biases. This heuristic focuses the learner on those hypotheses (rules) that address incompleteness in the current description. Finally, instead of choosing the first candidate bias that improves the performance, ClimBS can pick the one that provides the most improvement (the results presented below use the defaults).

The description in Figure 1 was simplified in two ways. In the innermost loop the operators are selected based on a probability of application associated with each operator. This allows, for example, operators that take a step back along a given dimension (to undo possible random missteps) to be applied with a lower probability than the forward operators. This facet of the system has only had a superficial empirical treatment; initial results indicate that the system is relatively insensitive to the precise settings of the operator probabilities. In addition, the system will quit if it reaches a local maximum, in the sense that all operators decrease the performance (seldom encountered). A related problem that is that of convergence. Occasionally, the system will reach a point where for a long time no operator makes any improvement. The system has a threshold on the number of nodes searched (usually set at 100,000); convergence criteria are currently under investigation.

The bias space that ClimBS navigates is delineated currently by four dimensions (examples of initial values given): (i) the positive performance threshold (0.9), (ii) the negative performance threshold (0), (iii) the rule complexity (1), and (iv) the beam width of the heuristic search (1). The positive threshold specifies the percentage of positive examples a satisfactory rule must cover. The negative threshold specifies the maximum percentage of the negative examples that a satisfactory rule may cover. The rule sets are to be used in a system that can take advantage of imprecise rules when gathering evidence for a classification. The rule complexity specifies the number of conjuncts that may appear in the antecedent of a rule. A change in the beam width of the system indicates a change in the confidence in the heuristic hypothesis evaluation function. Additional dimensions can be addressed by adding the appropriate operators to ClimBS' operator set. In principle, an operator can change any aspect of the PDM.
Input: training examples; evaluation examples (if null, set to training examples)
PDM: Partial Domain Model -- (initial bias)
B : set of bias transformation operators which operate on PDM
Output: Description of concept to be learned (set of rules)

ClimBS:
rules := Ø ; performance := 0
while stopping conditions are not met (from PDM) loop
for each opi ∈ B whose preconditions are satisfied (in order)
PDMi := result of applying opi to PDM
new-rules := learning system(PDMi , training examples)
new-rules := combine(rules, new-rules, evaluation examples)
new-performance := performance system(rules, evaluation examples)
if new-performance > performance then
  performance := new-performance
  rules := new-rules
return from for each
if no PDMi increased performance then pick randomly from those that didn't decrease
(if all PDMi decreased performance then local max -- done)
combine(rules, new-rules, evaluation examples)
default: union of rules and new-rules (monotonic construction)
learning system: MC-RL, multiclass rule learning system
performance system: default: simple inference engine, rules vote, classification accuracy as score

Figure 1 -- ClimBS' Basic Algorithm

For the results presented in the next sections, the bias space was structured such that the negative threshold was held constant while ClimBS considered modifying the bias along the other three dimensions. Starting with a very strong bias, when the positive threshold gets low (and fitting the data becomes more likely) the preconditions for the negative threshold operator are satisfied. ClimBS then considers increasing the negative threshold and resetting the bias along the other dimensions to its original, strong values (in a nested loop structure). This technique was modelled after methods that proved effective in manual bias selection. Subsequent results indicate that even without this rigid structure, equivalent results can be obtained [5].

5 Results -- UCI databases

ClimBS was run on several domains from the UCI Repository to test the generality of the bias space navigation strategies modelled. The initial state and the bias transformation operators were set and held constant across the domains, so that no user interaction was required. The first task was to predict the edibility of mushrooms (cf. [18]). The second domain concerns heart disease diagnosis. The third domain describes automobiles (circa 1985), where the classification task was to place them into price categories (<$10K, $10K<price<$20K, >$20K). The results are summarized in Table 1, which lists the domain, the numbers of training and test examples, and the average predictive accuracy on a test set (with 95% confidence intervals) of rules learned by ClimBS, MC-RL with manual bias selection (by the author), and a decision tree program (ID3 splitting criterion, no pruning) with the tree converted to rules, which were then pruned (individually, then as a set, similar to the method described in [19]) and used in the voting inference engine. The confidence intervals were determined using Student's t distribution; the statistics were collected over multiple runs with randomly selected training/test sets. In the Auto and Heart domains, all differences are significant at the 0.01 level, except between the last two columns of the Auto--this difference is significant at the 0.05 level.

These results indicate that ClimBS' was able to learn concept descriptions that perform well, compared to those learned by MC-RL with its bias selected manually, and compared to a decision tree program. The decision tree program's algorithmic bias was built in, and not easy to adjust to improve the performance. On the other hand, its static bias provided competitive results without any adjustment. ClimBS' improvement over MC-RL is due mainly to two factors. ClimBS had more patience than I did in selecting a good bias, and thus tried more tentative biases. Also, ClimBS searched the bias space for each new set of training examples, whereas I selected a good bias for the first training set, and used that bias for all subsequent sets.
6 Analysis on a synthetic domain

This section presents briefly the results of empirical analysis of ClimBS' performance on two learning tasks from a synthetic domain with 22 attributes each with between 2 and 12 values. The synthetic domain was chosen so that the "correct" concept could be known. The data for Task 1 were generated from a set of 10 rules that were scattered throughout ClimBS' bias space. Some of the rules are "heuristic" rules, in that they have non-null false positive coverage. Task 2 was to learn a set of 20 rules, similar to those in Task 1. The training sets contained 200 examples, the test sets 2000. Each data point in this section is the average of 10 runs (with 95% confidence interval).

6.1 A high level comparison

Table 2 compares various learning methods. The MC-RL system was run using the strongest bias (representable using its PDM) sufficient to learn the concept. The decision tree program (ID3 splitting criterion, no pruning) was tested, the trees converted to rules, and the rules pruned (individually, then as a set, similar to the method described in [19]). ClimBS was run with operators similar to those used above and each bias was evaluated with the training set or with a separate evaluation test set (distinct from final test set). The final entry shows the results of pruning ClimBS' rules. Compared are the number of nodes searched in learning (DT nodes for rough comparison only), the size of the resulting concept description (rules or tree nodes), and the accuracy on the test set.

The decision trees' poor performance indicates that the problem given is not trivial. The poor performance of the DTs before pruning is due partly to the negative coverage of the rules used to generate the data. Another reason is that for a majority of the 22 attributes, each is mentioned in some rule; thus the decision tree program has a difficult time choosing a good attribute on which to split. It is interesting to note that the decision trees generate 200 rules for 200 examples in every case, and that their predictive accuracy is less than the 50% that a naive algorithm might predict. A comparison along the nodes searched column indicates that MC-RL, once it is given the correct bias, can form accurate concept descriptions quickly. ClimBS takes almost two orders of magnitude more search than MC-RL. A comparison along the accuracy column shows that ClimBS finds biases with which it can learn a highly predictive concept description, and that using a separate evaluation set made no difference in the final accuracy.

<table>
<thead>
<tr>
<th>Domain (UCI db)</th>
<th>No. Exs. (train/test)</th>
<th>Ave. Test Accuracy (%) (w/ 95% conf.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ClimBS</td>
</tr>
<tr>
<td>Automobiles</td>
<td>(133/66)</td>
<td>85.9±4.5</td>
</tr>
<tr>
<td>Heart Disease</td>
<td>(200/103)</td>
<td>81.9±13.4</td>
</tr>
<tr>
<td>Mushrooms</td>
<td>(100/915)</td>
<td>95.5±1.3</td>
</tr>
</tbody>
</table>

Table 1 -- ClimBS' Performance on 3 UCI database domains

<table>
<thead>
<tr>
<th>Method</th>
<th>Task 1</th>
<th>Task 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nodes</td>
<td>Size</td>
</tr>
<tr>
<td>MC-RL</td>
<td>53±88</td>
<td>26±3</td>
</tr>
<tr>
<td>DT</td>
<td>672±84</td>
<td>305±5</td>
</tr>
<tr>
<td>DT rules</td>
<td>672±84</td>
<td>200±0</td>
</tr>
<tr>
<td>DT rules (pruned)</td>
<td>672±84</td>
<td>12±1</td>
</tr>
<tr>
<td>ClimBS (t.evail.)</td>
<td>1895±2609</td>
<td>30±2</td>
</tr>
<tr>
<td>ClimBS (sep.evail.)</td>
<td>2146±4009</td>
<td>31±3</td>
</tr>
<tr>
<td>ClimBS (pruned)</td>
<td>1895±2609</td>
<td>11±1</td>
</tr>
</tbody>
</table>

Table 2 -- comparative results for the synthetic domain
Pruning (on the training set) increases the accuracy of the decision tree rules (and is in fact necessary); in one case the accuracy of the ClimBS (MC-RL) rules was increased while in the other case it was decreased. In the latter case the difference was not significant at the 0.05 level, but it was observed across the board in Task 2; this seems to be partially due to overpruning, seeing as the sets of rules in Task 2 after pruning are half the size of the sets used to generate the data (remember that there was a good deal of overlap, so not all rules are necessary for relatively good performance). The concept size column indicates that pruning is necessary for achieving simple concepts in this domain with only 200 training examples.

The size of the space defined by the description language is $1.4 \times 10^{17}$ nodes. The hypothesis space defined by ClimBS' operators is smaller, $1.4 \times 10^8$ nodes, due to a restriction on the maximum number of conjuncts in a rule. Although ClimBS' search takes much longer than MC-RL's, it takes several orders of magnitude less time than an exhaustive search of its hypothesis space. ClimBS takes proportionally as little, if not less, search than MC-RL with respect to the size of the hypothesis space defined by its bias ($2.7 \times 10^5$ for MC-RL). However, MC-RL's heuristics are strong enough that (in this domain) eliminating the syntactic limit on the language complexity does not increase the search much if the bias is correct along the other dimensions (otherwise search can increase drastically).

6.2 (In)sensitivity to operator parameters

The operators used in ClimBS to generate the results above were based on strategies that had proven useful in manual bias selection. In this section I discuss the (in)sensitivity of ClimBS' results to changes in its operators. The operator (Section 5) that changed the beam width did so by doubling it. To address the concern that this would cause an exponential increase in the size of the hypothesis space, when it might not be necessary, the beam width operator was changed to use an additive increment. The results seemed not to be affected by this modification, nor were they sensitive to the precise value of the increment. The results in Section 6.1 were obtained using an increment of 2.

The operators used to generate the above results had ceilings on the increase of the beam width and the number of conjuncts in the left hand side of rules. In most cases, neither the number of nodes searched nor the accuracy of the resultant concept description was affected by the removal of these limits, which increased space of hypotheses considered to the entire syntactically defined space ($1.4 \times 10^{17}$ rules for this domain). Occasionally ClimBS' search would get lost and wander about with very large beam widths and large numbers of conjuncts until the maximum number of nodes (100,000) was reached. This seemed to be a problem of convergence, since the accuracy of the resultant descriptions was comparable with those that terminated normally. Detection of convergence is currently handled by the simple technique of counting the number of nodes searched since the last improvement and using a cutoff.

Changing the starting value for the positive threshold made little difference. If the value was too high, ClimBS wandered about randomly for a while but eventually reduced the threshold enough to learn some rules. This didn't take much time since the number of hypotheses which cover, for example, 90% of the positive examples of a concept is small. Decreasing the minimum positive threshold that ClimBS will allow had some effect. If ClimBS was forced (by the structure of the bias space) to consider rules which covered too few examples, a fitting of the data was observed and the accuracy dropped. In Task 2, using a separate set of examples to evaluate the biases seemed to remedy the situation (see Table 3). In Task 1 it did not help. The difference in accuracy in the Task 2 "training eval." accuracies of Tables 2 and 3 is significant at the 0.05 level. The decrease in search nodes is due to ClimBS quitting when it covered all the training examples (part of the stopping condition).

<table>
<thead>
<tr>
<th>Method</th>
<th>Nodes searched</th>
<th>Size of concept</th>
<th>Acc. on test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClimBS (training eval.)</td>
<td>9001±1180</td>
<td>38±4</td>
<td>96.0±0.7</td>
</tr>
<tr>
<td>ClimBS (separate eval.)</td>
<td>13341±5643</td>
<td>38±5</td>
<td>97.0±1.3</td>
</tr>
</tbody>
</table>

Table 3 -- Task 2: range of thresholds widened
7 Conclusions and Future Work

SBS is an effective model for both studying strategic considerations in automatic bias selection and for building automatic bias selection systems. The ClimBS system was shown to be effective in several of the UCI database domains and in a synthetic domain of overlapping, disjunctive rules, when given a set of bias transformation operators modelled after manual bias selection strategies. In addition, the system was found to be insensitive to the precise form of the operators.

The ClimBS system, with its hill climbing strategy, has shown effective only in dealing with dimensions in the bias space that can be given in a strong to weak ordering. It is not clear how the system would deal with categorical (or binary) biases (e.g., pick the most general/specific hypothesis). A different strategy might have to be employed.

SBS-based systems get leverage from the model-driven nature of the MC-RL system, but such a system must have a model of the bias selection process. Without a "meta-level" bias, we would be back to the problem of having no bias at all. An SBS-based system, in that it represents its bias selection operators explicitly, can accept and use different inductive policies (and even modify its own). An example of using a different inductive policy is to combine the learning of general rules with the collection of specific cases (cf. [20], [21]). An SBS-based system has been implemented with operators that first look for general rules and then look for very specific (case-like) rules [5].

Another policy decision regards the sensitivity of the learner to the costs of making different predictive errors. [3] shows that ClimBS' can be given different policies corresponding to different tradeoffs with respect to classification accuracy versus cost of making a mistake. It has also been shown [5] that incremental batch learning [22] can be implemented as bias selection, where the dimension in the bias space is the set of examples that will be used for learning. An operator can be given to an SBS-based system that changes the set of examples considered and updates the concept description accordingly.

The operators currently implemented in ClimBS move through the bias space by adjusting parameters to the underlying learning system. However, the SBS model is not limited to searching along dimensions that one would readily deem "parameters." For example, the content of the CDL is an obvious candidate for a bias transformation operator, and existing constructive induction operators would fit well into the current model (see [6] for an overview of existing constructive induction operators), as would operators for term selection. Another dimension to address is the collection of domain knowledge constraints used to bias the search. Additionally, the selection of "meta-level" bias will be straightforward. SBS also will provide a framework for discovering techniques for learning effective bias space search strategies.

Acknowledgements

I would like to thank Bruce Buchanan, Scott Clearwater, Randy Jones, Tom Mitchell, Jeff Schlimmer, Kurt Van Lehn, and the University of Pittsburgh machine learning discussion group for their helpful comments, Jeff for the decision tree code, and the UCI Repository, for the access to the machine learning database. Heart disease data were supplied by V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, MD, Ph.D.

References


