Performance-Driven Knowledge Representation

Lawrence B. Holder
Department of Computer Science
University of Illinois
405 North Mathews
Urbana, IL 61801

Abstract

Integration of machine learning methods with knowledge based systems requires sophisticated control mechanisms for applying methods appropriate to the performance task. Performance-driven knowledge transformation controls the application of learning methods based on their ability to achieve desired performance goals while preserving the performance on other tasks. A means-ends approach to performance-driven knowledge transformation is presented along with experimental results from an implementation. The results indicate that performance-driven knowledge transformation is able to maintain multiple performance goals by applying appropriate machine learning methods to transform a knowledge base.

1 Introduction

Intelligent systems acquire knowledge in order to perform some task. The level of performance depends upon the quality of the acquired knowledge. Machine learning methods attempt to improve the quality of the knowledge, and thus improve performance on the desired task. Integration of machine learning methods with knowledge-based systems will reduce the dependency of system performance on the quality of knowledge initially entered by the knowledge engineer. For example, instead of extracting high-level rules from a domain expert, the knowledge engineer might need to collect only previous instances with known conclusions. The machine learning component of the knowledge-based system may then generalize the instances into knowledge of the appropriate quality.

Several machine learning methods have been developed that successfully derive higher-level knowledge from examples. However, these methods work in isolation and do not consider the need for multiple learning methods within a single knowledge-based system. For instance, if the knowledge takes the form of examples and the goal is improved accuracy, then the system may choose to invoke an empirical learning method that constructs generalized rules describing the examples. On the other hand, if the knowledge takes the form of higher-level rules and the goal is improved response time, then the system may choose an analytical learning method to construct a macro from the rule inference chain used to solve the current problem. Knowledge-based systems must be able to select learning methods appropriate for the desired performance improvement. In addition, application of the learning method must preserve the performance goals of other knowledge used for other tasks.

Knowledge-based systems need the capability of applying multiple learning methods in order to adapt to new performance requirements while maintaining previously stated goals.

For example, once a certain level of accuracy has been achieved, the emphasis may shift to the time needed to arrive at an accurate decision. The desired accuracy may have been achieved by rote learning examples, and then the desired speed may be achieved by generalizing these examples to a few general rules. Current individual learning systems are incapable of a corresponding shift of attention.

Performance-driven knowledge transformation controls the application of learning methods based on their ability to achieve desired performance goals while preserving the performance on other tasks. The next section describes work related to performance-driven knowledge transformation. Then, the approach is discussed in detail. Next, results are presented from preliminary experimentation with an implementation of the approach in the Peak system.

2 Related Work

Experimentation with empirical learning programs has shown that more emphasis should be placed on performance during the construction of the learned concepts. Evidence for this can be found in the results obtained with the decision trees generated by Quinlan's ID3 program (Quinlan 1986). Quinlan found that *pruning* the rules extracted from a decision tree can improve the accuracy of the rules on unseen examples (Quinlan 1987). Compared to the original rules, the pruned rules performed better on a set of unseen test examples. Recent experiments by Mingers have confirmed this result (Mingers 1989).

Further evidence for the necessity of performance for guiding concept learning can be found in experimentation on explanation-based learning systems. In experimentation with the Prodigy system, Minton found that performance degrades as the number of rules grows large (the *utility problem*) (Minton 1988). In order to learn a concept, the system acquires several rules whose disjunction forms the system's understanding of the concept. As the number of rules increase, the cost of determining the applicability of a rule may outweigh the benefits of applying, and thus, retaining the rule. Minton's solution is to maintain empirical estimates of match costs, application savings and frequency of application for each rule. These estimates are used to compute a utility value for the rule. If this value becomes negative, the rule is no longer considered. Minton found that maintenance of a rule's utility value and compression of the rule's conditions result in a substantial performance improvement.

This related research illustrates that learning should be constrained by the desired performance. If the knowledge base satisfies the performance goals, then no learning is necessary. If performance is violated, then only enough learning should be done to achieve the performance without violating previously satisfied performance goals.

3 Approach

Performance-driven knowledge transformation controls the application of learning methods based on their ability to achieve desired performance goals. Each task for the knowledge base defines a performance space. The dimensions of the performance space are the performance goals (e.g., completeness, correctness, response time) to be maintained by the knowledge base for that task. The current state of the knowledge base is represented by a point in the performance space for each task. A knowledge transformation can be viewed as a move of the current knowledge base from one point in the performance space to another. Figure 1 shows the performance space for one task. The task consists of three performance goals G_1, G_2 and G_3. The location of two knowledge bases K_1 and K_2 are shown for the task.

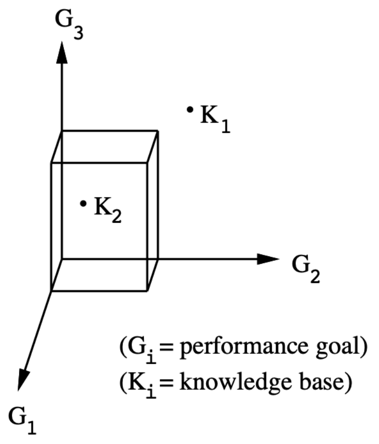

Figure 1: Performance Space for One Task

The desired performance for each task defines a hyper-rectangle in that task's performance space. When the knowledge base moves outside the performance hyper-rectangle in some performance space, performance-driven knowledge transformation selects a learning method to transform the knowledge base so that the corresponding point in the performance space for the current task moves inside the desired performance hyper-rectangle without moving the points for other tasks outside the desired hyper-rectangle in the performance spaces. Referring to Figure 1, knowledge base K_1 violates the performance goals for the task. Transforming knowledge base K_1 into K_2 achieves the desired performance goals.

This research investigates a means-ends approach to performance-driven knowledge transformation. When a performance goal violation is detected while solving a problem from some task, the means-ends approach uses information about the context of the goal violation (e.g., the difference between desired and actual performance) to select a transformation operator for reducing this difference while maintaining other performance levels. Application of the operator yields a new knowledge base. If the new knowledge base achieves the violated performance goals and preserves other performance goals, then the current knowledge base is replaced by the new knowledge base. Otherwise, another transformation operator is selected for application.

In the following discussion, certain assumptions have been made about the knowledge in the knowledge base and the performance element using this knowledge. The knowledge base is a set of Horn clause rules. The performance element is a deductive retriever similar to Prolog. Performance is measured while the performance element attempts to solve a query posed by the user. Attached to the query are the performance goals to be maintained during solution. Performance goal violations occur when the measured performance exceeds the desired thresholds.

3.1 Performance Perspective

Using performance goals as a means of guiding the maintenance and repair of a knowledge base requires a precise definition of performance. The definition of performance depends on the perspective. Four perspectives are applicable for describing the performance of a knowledge base:

- **External** performance is the performance measured from outside the knowledge base, regardless of any internal knowledge transformations.
- **Current** performance is the performance the system currently maintains for the previously seen queries.
- **Expected** performance is the performance the system expects to demonstrate on future queries. Expected performance is usually the same as current performance.
- **Absolute** performance is the performance that the current state of the knowledge would support if given every possible query.

When the user specifies a threshold for some performance measure, the proper perspective must be used to evaluate the performance of the knowledge base. *Absolute* performance is rarely available due to a lack of knowledge about the instance space. *Absolute* performance is inappropriate, because the distribution over the entire instance space may not give equal probability to each instance. *External* performance provides information about the rate of convergence towards absolute performance. Changes in *external* performance indicate the need for an increase or decrease in the extent of the knowledge transformations. *Current* performance evaluates the knowledge only on previously seen queries. *Expected* performance is the best measure of the current state of the knowledge base, because the objective of the knowledge base is to maintain its expected ability to perform the task within desired thresholds on possibly unseen queries.

Performance-driven knowledge transformation should measure both *expected* and *external* performance. Knowledge transformations are triggered only when *expected* performance falls below desired levels. *External* performance should then be used in the selection of an appropriate transformation operator. The greater the difference between *external* and *expected* performance, the more drastic a transformation operator should be recommended by the system.

3.2 Information on Goal Violations

Once a goal violation has been detected, several pieces of information are available for selecting an appropriate knowledge transformation operator. First, as described in the previous section, the difference between expected and external performance indicates the extent of the necessary transformation.

Second, after the performance element attempts to solve a query, the violated and preserved goals are known. Each goal contains information about the performance measure that this goal constrains, the desired threshold on the measure, the observed value of the measure on previously seen queries (including the query just processed), and the difference between the observed and desired performance (the error). The performance measure constrained by a violated goal is useful for selecting transformation operators capable of improving this performance measure. The magnitude of the error indicates the extent of the transformation. The performance measure constrained by a satisfied goal is useful for selecting transformation operators capable of preserving this performance measure. The magnitude of the error indicates the extent to which the selected operator may degrade performance on the satisfied goals in order to achieve performance on the violated goals.

A third source of information that will be available upon detection of a performance goal violation is the *task history.* Each task known to the knowledge base maintains a task history of previously seen queries from the task. The task history serves two purposes. First, the task history represents an empirical estimate of the distribution over the possible queries of the task. This distribution can be used to verify the achievement of violated performance goals in transformed knowledge. Second, an entry in the task history contains information about the query-solving episode. One useful piece of information about a query-solving episode is the trace of the knowledge accessed during the solution.

The *knowledge trace* is an and/or tree that records the knowledge accessed during the solution of the query and indicates which rules (if any) support the response to the query. Information about the shape of a task's knowledge traces constrains the selection of knowledge transformations. For example, wide, shallow knowledge traces indicate that the knowledge consists of specific instances of the task; whereas narrow, deep knowledge traces indicate a more general set of rules for proving queries from the corresponding task.

Finally, past success of the transformation operators provides information upon performance goal violation. As the knowledge base transforms to meet performance goals, a record is kept of the old and new knowledge bases along with the operator responsible for the transformation. If the new knowledge base achieves a violated goal while preserving non-violated goals, then the system increases the operators applicability for achieving and preserving the appropriate goals. Over time, collection of this information will allow the system to make a more informed operator selection based on past experience.

3.3 Verification of Knowledge Base

Because no operator application is guaranteed to achieve the desired results, the system must verify that the knowledge base resulting from an operator application achieves the desired performance. Verification can be accomplished by re-solving the queries in the task history. The size of the task history can be changed to tradeoff performance convergence rates for transformation speed. As the system learns operator applicability, there is less chance of multiple verification being necessary to repair one goal violation; thus, the task history size can be increased over time.

4 Experimentation

This section illustrates an experiment conducted with the Peak system. The experiment follows the shuttle landing control database available from the machine learning databases maintained by University of California at Irvine. The problem is to determine whether to land the shuttle manually or automatically based on environmental attributes. The corresponding task is labeled the *landing* task, and the queries are of the form `landing(ENV,?x)`. The `ENV` in the query represents the environmental situation to be evaluated. The performance element attempts to fill in the `?x` with the recommended landing control: `auto` or `noauto`.

Prior to query answering, the user inputs the performance thresholds to be maintained by the knowledge base while answering *landing* queries using the performance element (a backward-chaining deductive theorem prover for Horn clauses). For this experiment, three performance goals are specified: *correctness*, *completeness* and *response time*. The correctness goal specifies that the answers to queries must be correct 90% of the time. The completeness goal specifies that the query must be answered 95% of the time. That is, the answer should be either `auto` or `noauto` and not *"I don't know"*. The response time goal specifies that the performance element must respond within 10 seconds.

Two knowledge transformation operators are available: rote learning and empirical learning. Application of the rote learning operator asks the user for the correct answer to the query. A new rule is added to the knowledge base having the instantiated query as the consequent, and the facts defined before query execution as the antecedent. The empirical learning operator utilizes the ID3 program to build a decision tree from examples in the knowledge base. The examples are rules such as those learned by the rote operator. Each path in the resulting decision tree is converted to a rule. The examples are replaced by the new rules in the transformed knowledge base.

Starting with an empty knowledge base, PEAK attempts to solve *landing* queries, while maintaining the performance goals. Figure 2 plots the three performance goals for 200 randomly chosen queries from the shuttle landing control domain.

Figure 2a illustrates how PEAK maintains response time performance below 10 seconds. For the first 30 queries, response time increases as the number of rote-learned rules increases. Eventually, the large number of rules in the knowledge base cannot be traversed within the response time threshold.

While processing the 30th query, PEAK was unable to solve the query, generating a completeness failure. PEAK first trys to transform the knowledge base by rote-learning a new rule. However, verification of the new knowledge base uncovers a response time failure. Because the rote learning operator was ineffective, PEAK chose to apply the ID3 operator. ID3 generalized the 29 learned instances into 8 general rules. As Figure 2a indicates, the resulting transformation drastically improves response time performance.

The plot of completeness performance in Figure 2b illustrates how PEAK quickly learns the initial query knowledge. After the ID3 transformation, completeness remained above the 95% threshold for the remainder of the 200 queries.

The correctness plot in Figure 2c shows how performance starts at 100% and converges to the desired 90% threshold. The initial values of 100% for correctness are due to the fact that many of the initial queries could not be answered. Correctness performance only measures the correctness of answered queries. Immediately following the application of ID3, correctness falls to 94% due to the next two queries being incorrectly answered according to the new knowledge base. As query answering continues, the over-generalization in the rules eventually brings correctness down below the 90% threshold. Correctness violations occur at queries 89, 98, 153 and 163. In each case, PEAK uses the rote-learning operator to memorize the incorrectly answered query and restore 90% correctness performance.

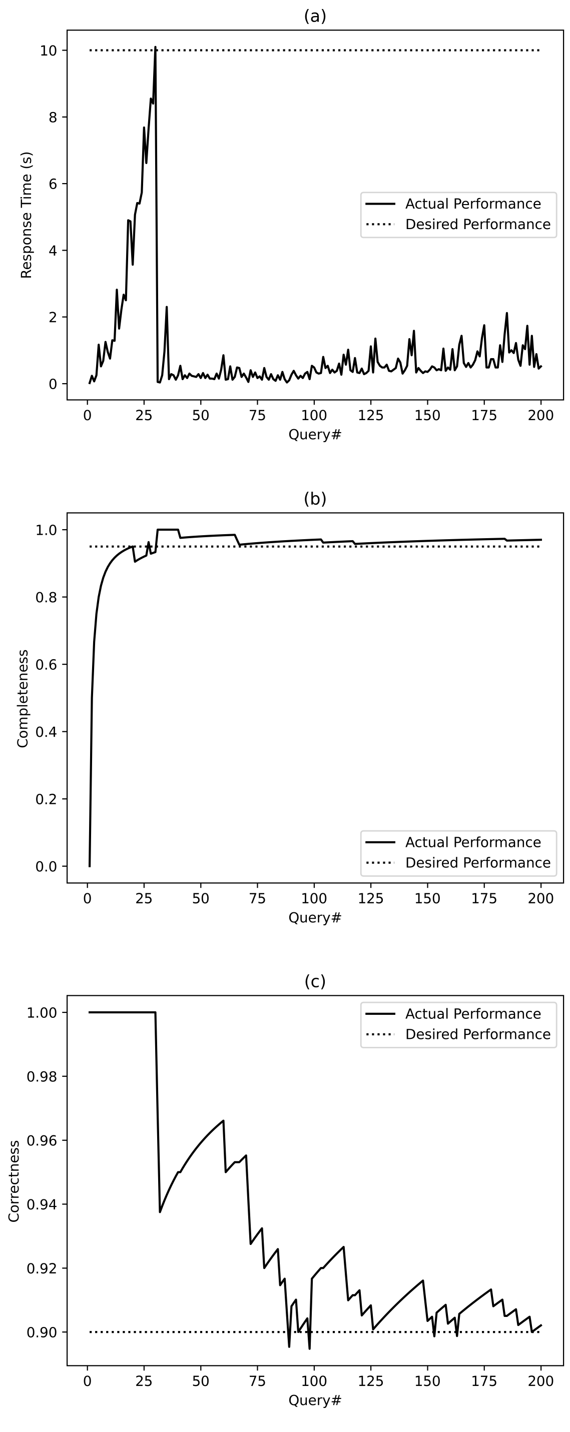

Figure 2: Plots of Performance Obtained by PEAK

The final knowledge base after completion of the 200 queries consists of the 12 rules shown below.

1. landing(x,noauto) ← sign(x,nn) & wind(x,head) & stability(x,xstab) & error(x,MM) & magnitude(x,Medium) & visibility(x,yes)
2. landing(x,noauto) ← sign(x,pp) & wind(x,tail) & stability(x,xstab) & error(x,MM) & magnitude(x,Low) & visibility(x,yes)
3. landing(x,noauto) ← sign(x,nn) & wind(x,head) & stability(x,stab) & error(x,MM) & magnitude(x,OutOfRange) & visibility(x,yes)
4. landing(x,noauto) ← sign(x,nn) & wind(x,tail) & stability(x,xstab) & error(x,MM) & magnitude(x,Low) & visibility(x,yes)
5. landing(x,auto) ← error(x,MM) & visibility(x,yes)
6. landing(x,auto) ← stability(x,stab) & error(x,SS) & magnitude(x,Strong) & visibility(x,yes)
7. landing(x,auto) ← visibility(x,no)
8. landing(x,noauto) ← error(x,XL) & visibility(x,yes)
9. landing(x,noauto) ← error(x,LX) & visibility(x,yes)
10. landing(x,noauto) ← stability(x,xstab) & error(x,SS) & magnitude(x,Strong) & visibility(x,yes)
11. landing(x,noauto) ← error(x,SS) & magnitude(x,OutOfRange) & visibility(x,yes)
12. landing(x,noauto) ← error(x,SS) & magnitude(x,Low) & visibility(x,yes)

Rules 5-12 are the general rules learned by ID3. Rules 1-4 are the specific instances learned to repair the over-generalization in ID3's rules. After 200 queries, the knowledge base converged to 8 general rules describing major trends in the shuttle landing domain and four specific rules for special cases not handled correctly by the general rules.

One final observation from Figure 2 is the convergence of the performance towards the desired thresholds and not towards the maximum possible performance. This indicates how performance-driven knowledge transformation utilizes flexibility in one dimension of performance to improve performance in another dimension.

5 Conclusions

The goal of knowledge acquisition is the ability to perform some task. The goal of learning is to improve performance on the task. Performance-driven knowledge transformation controls the application of learning methods to maintain the desired performance levels of the knowledge base. Future experimentation with the Peak system will analyze the convergence properties of a knowledge base with respect to the available transformation operators and the performance goals. The results will provide both theoretical and applied mechanisms for the integration of learning methods with knowledge-based systems.

Acknowledgments

I would like to thank Robert Stepp, Diane Cook, Brad Whitehall and Robert Reinke for their helpful suggestions on this work.

References

Mingers, J. 1989. "An Empirical Comparison of Pruning Methods for Decision Tree Induction." *Machine Learning* 4, no. 2 (Nov.): 227-243.

Minton, S. 1988. "Quantitative Results Concerning the Utility of Explanation-Based Learning." In *Proceedings of the National Conference on Artificial Intelligence* (St. Paul, MN, August), 564-569.

Quinlan, J. R. 1986. "Induction of Decision Trees." *Machine Learning* 1, no. 1, 81-106.

Quinlan, J. R. 1987. "Generating Production Rules from Decision Trees." *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (Milan, Italy, August), 304-307.