# Unifying Empirical and Explanation-Based Learning by Modeling the Utility of Learned Knowledge

**Lawrence B. Holder**
Department of Computer Science Engineering
University of Texas at Arlington
Box 19015, Arlington, TX 76019

## Abstract

The overfit problem in empirical learning and the utility problem in explanation-based learning describe a similar phenomenon: the degradation of performance due to an increase in the amount of learned knowledge. Plotting the performance of learned knowledge during the course of learning (the performance response) reveals a common trend for several learning methods. Modeling this trend allows a control system to constrain the amount of learned knowledge to achieve peak performance and avoid the general utility problem. Experiments evaluate a particular empirical model of the trend, and analysis of the learners derive several formal models. If, as evidence suggests, the general utility problem can be modeled using the same mechanisms for different learning paradigms, then the model serves to unify the paradigms into one framework capable of comparing and selecting different learning methods based on predicted achievable performance.

## 1 INTRODUCTION

As machine learning methods acquire increasing amounts of knowledge based on imperfect instances, the proliferation of low-utility knowledge increases, and performance degrades. The *general utility problem* in machine learning refers to the degradation of performance due to increasing amounts of learned knowledge [Holder, 1990]. This term derives from the *utility problem* used by Minton [1988] to describe this phenomenon in analytical learning, but generalizes to other machine learning paradigms.

Other researchers have observed the ubiquity of the utility problem in machine learning paradigms. Yoo and Fisher [1991] compare the utility problem in analytical learning to the problems of noise and overfit in empirical learning. This paper investigates the mechanisms underlying the general utility problem and attempts to unify empirical and explanation-based learning within the context of this problem. The next section introduces a tool for analyzing the general utility problem called the performance response. Section 3 uses the performance response in experimentation with several empirical learning methods, and Section 7.2 provides similar experimental results with analytical learning. Results indicate a global trend in the performance response of several different learners suffering from the general utility problem.

The ability to model this trend would allow a control system to constrain the learning method to learn just enough knowledge to achieve peak performance and avoid the general utility problem. Section 6 introduces and evaluates the Model-Based Adaptive Control (MBAC) system that fits a parabolic model to the performance response trend. In order to obtain increased model accuracy and avoid the need for performance response sampling, formal models of the performance response trend are necessary. Section 7 describes several such formal models. Section 8 concludes by considering the commonality of the performance response model in several learning paradigms and the possibility of unifying different paradigms based on this model.

## 2 PERFORMANCE RESPONSE

A useful tool for analyzing the general utility problem in machine learning is the performance response. The *performance response* is the performance of the learned knowledge measured during the course of learning. Figure 1 illustrates the typical performance response of a learning method that suffers from the general utility

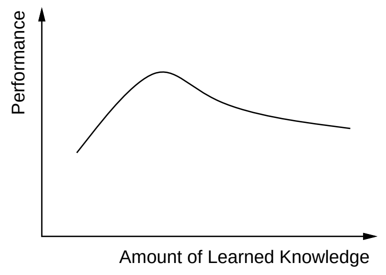


Figure 1: Performance response indicative of the general utility problem.

problem.

The horizontal axis of the performance response measures the amount of learned knowledge. The units along this axis represent the change in learned knowledge made by a knowledge transformation. A knowledge transformation is a decomposition of the learning method into less complex operations affecting the learned knowledge. For example, one decomposition of a splitting algorithm is a single split, and one decomposition of a neural network learning algorithm is a cycle. Since a knowledge transformation may not always increase the amount of learned knowledge in terms of the size of the set of knowledge, an increase along this axis more generally represents a refinement of existing knowledge.

The vertical axis of the performance response measures the performance of the learned knowledge after each transformation. The measure of performance depends upon the learning method. Different methods attempt to improve different dimensions of performance. For example, a neural network primarily attempts to improve classification accuracy, while an explanation-based learning method attempts to improve problem-solving speed. Other performance measures on the learned knowledge include the complexity and storage cost of the knowledge. The classification accuracy of empirical learners and the problem-solving speed of analytical learners are the focus of this work.

As an example, Figure 2 illustrates three performance responses obtained from the ID3 empirical learner [Quinlan, 1986] on the DNF2 domain [Pagallo and Haussler, 1990]. ID3 constructs a decision tree from the training data by splitting the data at a node. Splitting continues until satisfaction of a stopping criterion. Each performance response in Figure 2 represents a different traversal (node split order) of the decision tree. Performance is classification accuracy, and the amount of learned knowledge increases with the number of splits. Each performance response is an average

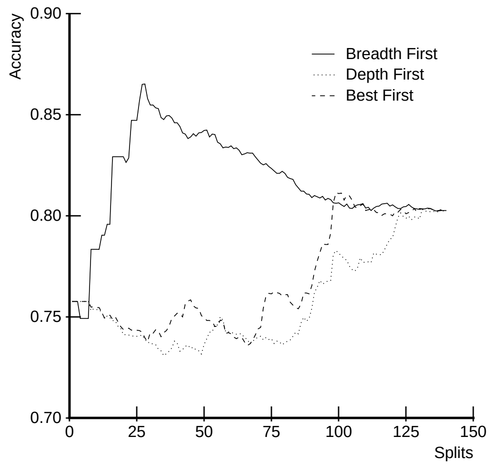


Figure 2: Performance responses for three traversals of a decision tree induced from the DNF2 domain.

over ten trials. Each trial consists of selecting random training and testing sets, generating the decision tree using the training set, and measuring accuracy after each split using the testing set.

As Figure 2 reveals, the order of the knowledge transformations is important for perceiving the desired performance response trend in Figure 1. The following sections discuss this issue in more depth. When learning proceeds from general to specific, the performance response follows the trend in Figure 1. Holder [1991a] plots the performance responses for several learning methods and shows that the trend is common to the methods. The next two sections provide a more formal understanding of the mechanisms that cause this trend in the performance response. Section 7.1 analyzes empirical learning methods, and Section 7.2 analyzes analytical methods. In both cases, the performance response trend results from two contributing forces and depends on a precise definition of the amount of learned knowledge in terms of the generality of this knowledge.

## 3 EMPIRICAL LEARNING RESPONSES

The general utility problem in empirical learning relates to the overfit problem. Overfit occurs when the learning method identifies errant patterns in the training data. Errant patterns may arise due to noise in the training data or inadequate stopping criteria of the method. As demonstrated below, splitting, setcovering and neural network learning methods suffer from overfit.

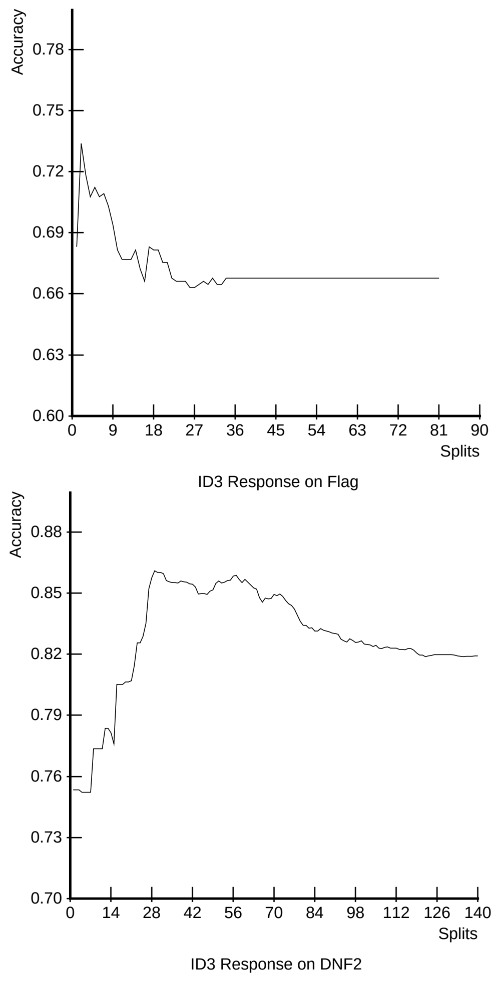


Figure 3: Performance response of ID3.

### 3.1 SPLITTING METHODS

Splitting methods recursively split the set of training data by choosing an appropriate feature or feature-value pair. The knowledge produced by a splitting method can be represented as a decision tree. The learned knowledge changes every time the method makes a split; therefore, one choice for the x-axis of the performance response is the number of splits. The y-axis (performance) measures the classification accuracy of the knowledge after each split, as measured using a separate set of test data.

ID3 [Quinlan, 1986] induces decision trees by recursively splitting the given set of training instances. The ID3 performance response in Figure 3 plots the accuracy of the decision tree on a separate set of testing instances after each split. Splits are performed in a breadth-first order, deferring overfit to the later splits. Figure 3 shows the performance response of ID3 on the Flag[1] and DNF2 domains using the node-purity stopping criterion. As the figure illustrates, this stopping criterion causes overfit, and the performance responses follow the trend of Figure 1.

Two tree-pruning techniques have been developed to combat overfit: pre-pruning and post-pruning. Pre-pruning constrains the stopping criterion to prevent splitting of impure nodes when no feature provides a significant increase in information resulting from a split. Post-pruning uses the pure-nodes stopping criterion to generate the decision tree, but then removes subtrees of the resulting tree to improve performance.

Quinlan [1986] developed a pre-pruning technique for ID3 based on the chi-square statistic ($\chi^2$). Although chi-square pre-pruning reduces the number of splits, overfit behavior is still evident. Increasing the confidence value may further reduce the number of splits, but does not eliminate overfit. Quinlan's [1987] reduced-error post-pruning technique removes subtrees from the original decision tree until accuracy decreases on a separate set of pruning instances. The reduced-error pruning alleviates most of the overfit, but on average the accuracy of the resulting tree is less than the peak accuracy of the performance response (see Table 1 in Section 5). Actual plots of these versions of ID3 can be found in [Holder, 1991a].

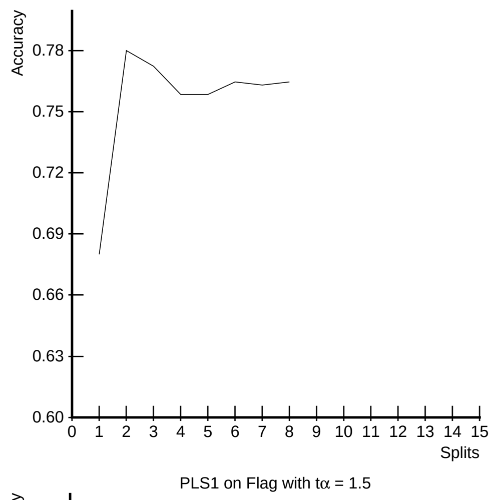
The PLS1 program [Rendell, 1983] is similar to ID3, but performs binary splits on feature-value pairs. At each iteration, PLS1 chooses the split that maximizes a probabilistic dissimilarity measure until the measure is no longer positive. The measure includes a factor $t_\alpha$ that reduces the dissimilarity according to error in the data. Typical values for $t_\alpha$ are between 1 and 2.

Since PLS1 chooses to split only if the maximum dissimilarity is positive, the $t_\alpha$ constant can be used to restrict the amount of splitting. Thus, $t_\alpha$ in PLS1 plays a role analogous to the confidence level in the chi-square pre-pruning technique for ID3. Figure 4 shows the performance responses of PLS1 on the same domains used for ID3 with $t_\alpha = 1.5$. Plots for other values of $t_\alpha$ are in [Holder, 1991a]. As with the chi-square pruning of ID3, increasing $t_\alpha$ reduces the number of splits made by PLS1, but the tendency to overfit is still evident (see Table 1 in Section 5).

### 3.2 SET-COVERING METHODS

Set-covering methods construct a hypothesis which describes a subset of the training instances, and then applies the same method on the remaining training

[1]The Flag domain is available from the UC Irvine machine learning databases.



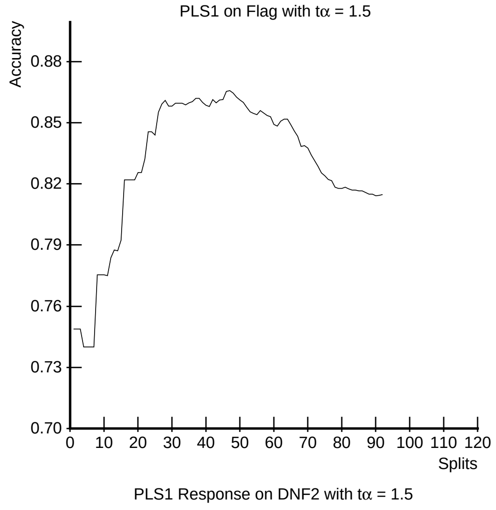


Figure 4: PLS1 performance response.

instances. Since set-covering methods typically learn disjunctive normal form (DNF) expressions for the hypotheses, the dimension used to measure the amount of learned knowledge is the number of disjuncts in the induced hypothesis.

During experimentation with the AQ system (specifically, AQ15), Michalski [1989] found that repetitive application of AQ can yield less accurate hypotheses than a more conservative application strategy combined with a more flexible inference mechanism than exact matching. Michalski compared the accuracy of the *complete* DNF hypothesis produced by AQ to truncated versions of the same hypothesis. The first truncated version of the hypothesis consists of the single disjunct covering the most examples (*best disjunct*). The second truncated version of the hypothesis consists of only those disjuncts covering more than one unique example (*unique > 1*). The truncated hypotheses use a simple matching procedure for classifying uncovered and multiply-covered examples (see [Michalski, 1989] for details).

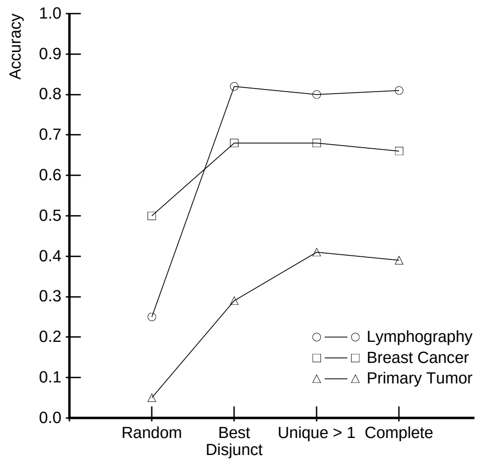


Figure 5: Performance response of AQ in three medical domains.

Although based on only four points, Figure 5 approximates the performance response of AQ in three medical domains (Lymphography, Breast Cancer and Primary Tumor) averaged over four trials. Figure 5 demonstrates that AQ also suffers from the general utility problem with increasing numbers of disjuncts, and the response curves indicate the same trend as in Figure 1.

The CN2 program [Clark and Niblett, 1989] is another set-covering empirical learning method. CN2 produces an ordered DNF hypothesis in the form of a decision list. Analyzing the hypotheses produced by CN2, Holte *et al.* [1989] reveal that the accuracy of the hypothesis degrades with the addition of small disjuncts. Because they are motivated from a small number of examples, small disjuncts are typically more error prone than large disjuncts. Therefore, CN2 suffers from the general utility problem due to the increasing amounts of low utility (small disjunct) knowledge.

## 3.3 NEURAL NETWORK METHODS

Neural networks learn from training data by presenting the feature values of an instance to the input layer, comparing the output layer's prediction to the instance's class, and updating the connection weights according to the difference. One method for updating the weights in such a network is *error back-propagation* [Rumelhart *et al.*, 1986]. Each error back-propagation pass through the training instances is called a *cycle*.

As the number of cycles increases, the network more accurately classifies the training instances. However, overfit eventually occurs as the network learns the

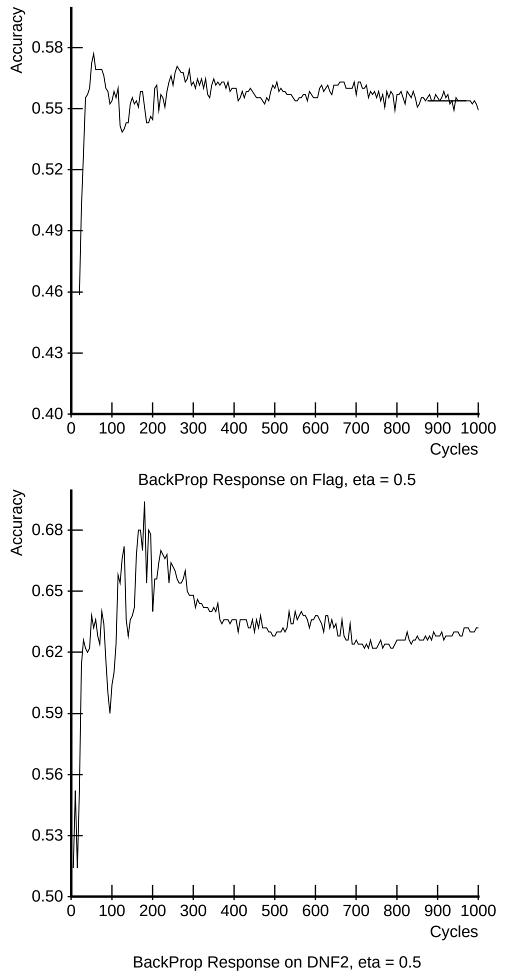


Figure 6: BackProp performance response.

training instances too precisely, degrading accuracy on the test data. To analyze the overfit of the back-propagation neural network, the performance response measures accuracy of the network after every five cycles.

Figure 6 shows the performance response of the error back-propagation neural network (BackProp) on the Flag and DNF2 domains. For these experiments, the learning rate $\eta$ was set at 0.5, and the network contained one hidden layer with four units. The choice of learning rate was arbitrary, because with a high enough learning rate, the number of hidden units determines the complexity of the function learnable by the network and, therefore, the extent of possible overfit [Karnin, 1990]. The output layer has two units, one for each of the two classes. The input layer has one unit for each feature-value pair in the domain. The network correctly classifies an instance if the output signals are within 0.1 of their desired values.

The BackProp response on the Flag and DNF2 domains follows the general utility problem trend as in Figure 1. Table 1 in Section 5 reveals that on average the network at the initial peak performs better than the final network.

## 4 ANALYTICAL LEARNING RESPONSES

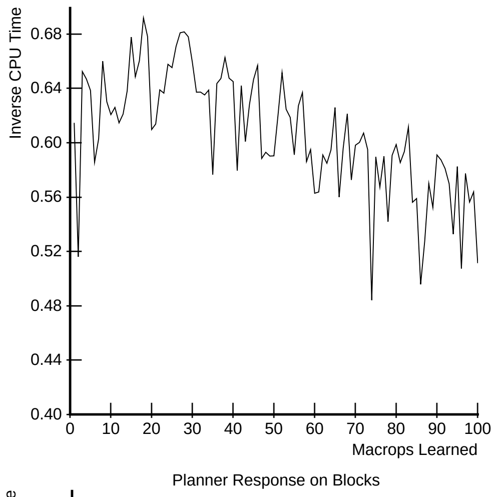
In experimentation with the MORRIS analytical learning system, Minton found that performance eventually degrades with increasing numbers of learned macro-operators [Minton, 1985]. Minton called this phenomenon the *utility problem* and offered the PRODIGY system as a solution [Minton, 1988]. PRODIGY maintains empirical estimates of rule utility and discards rules whose utility value becomes negative.

Experimentation with the SOAR [Tambe and Rosenbloom, 1989] and EGGS [Mooney, 1989] analytical learning systems indicates similar results. Tambe and Rosenbloom suggest restricting the expressiveness of the learned rules so that the complexity of the match is kept linear in the number of matching conditions. Mooney recommends limited use of the learned rules.

Although experimentation with the above analytical learning systems confirms the existence of the utility problem, the experiments typically do not show the performance response of the system. This section plots the performance response of a simple analytical learner in Figure 7. The analytical learner consists of a forward-chaining planner and a STRIPS-like plan generalizer [Fikes *et al.*, 1972]. Two domains are used in the experimentation: blocks and robot. The blocks domain consists of four operators for stacking and unstacking blocks. The robot domain consists of eight operators allowing the robot to move boxes within a layout of connected rooms.

The experiments proceed by solving a training problem in the domain, generalizing the resulting plan, adding the generalized plan to the set of available operators, and then measuring the amount of CPU time needed to solve a separate set of test problems using the augmented set of operators. The x-axis of the performance response is the number of learned macrops. The y-axis measures the inverse CPU time needed to solve the set of test problems. Inverse CPU time allows an increase along the y-axis to reflect an increase in planner performance.

Figure 7 plots the performance response of the planner while learning macrops in the blocks and robot domains. Although erratic in the blocks domain, both



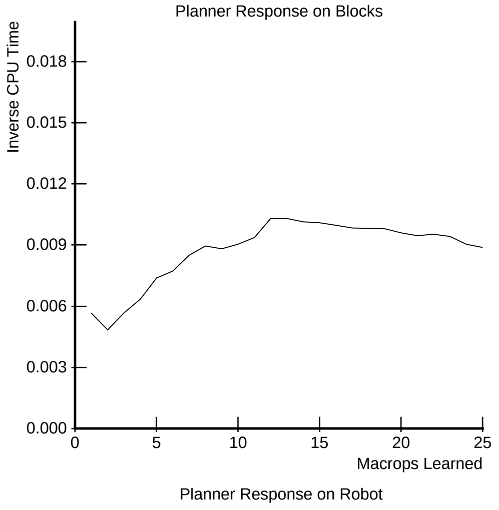


Figure 7: Planner performance response.

responses follow the trend of the general utility problem.

## 5 TRENDS

The previous sections verify the existence of the general utility problem in several machine learning methods. Furthermore, the performance responses of these methods follow the general trend illustrated in Figure 1. Adopting this trend as a model of the performance response permits the control of the general utility problem by constraining the amount of learned knowledge to reside at the point corresponding to the peak performance.

Tables 1 and 2 quantify the possible performance gains by using this model-based control of the amount of learned knowledge. Each entry in the tables is the percentage final performance of peak performance ($\frac{\text{final}}{\text{peak}} * 100$) averaged over ten performance response curves.

Table 1: Percentage final performance of peak for inductive learners.

| | Domain | | | | |
|---|---|---|---|---|---|
| Method | BC | Flag | Flare | Vote | DNF2 |
| ID3 | 91.2 | 88.2 | 95.0 | 97.6 | 93.6 |
| ID3 Chi 99.0 | 89.0 | 88.5 | 94.4 | 98.1 | 94.4 |
| ID3 Chi 99.9 | 90.8 | 89.9 | 96.1 | 97.0 | 97.2 |
| ID3 Red-Err | 98.6 | 95.4 | 98.7 | 99.7 | 100.3 |
| PLS1 $t_\alpha = 1.0$ | 87.9 | 96.3 | 97.7 | 98.1 | 92.8 |
| PLS1 $t_\alpha = 1.5$ | 92.4 | 97.6 | 98.5 | 98.9 | 92.8 |
| PLS1 $t_\alpha = 2.0$ | 94.6 | 98.4 | 98.5 | 99.3 | 95.6 |
| BP4 | 82.8 | 89.8 | 88.2 | 92.6 | 91.1 |

Table 2: Percentage final performance of peak for Planner.

| | Domain | |
|---|---|---|
| Method | Blocks | Robot |
| Planner | 67.4 | 76.1 |

Table 1 lists entries for several of the previously described empirical learning methods on five different domains[2]. Table 2 lists entries for the Planner analytical learner on two domains. Note that the entries in Table 2 can be arbitrarily deflated by allowing the analytical learner to acquire more macrops.

As shown in Tables 1 and 2, the final performance is less than the peak performance for all but one case. A majority of the values are statistically significant, and in the cases where the significance is low, the peak of the performance response is no worse than the final performance. Thus, the ability to constrain the amount of learned knowledge to the point corresponding to peak performance will improve the performance of the learner. Although individual methods exist for alleviating the general utility problem in each particular learning method, the performance response model offers a general method for avoiding the general utility problem in many machine learning methods.

## 6 EMPIRICAL MODELS

The performance response plots of previous sections show that several learning methods follow the behavior of the general utility problem in Figure 1. The results in the previous section show that learning methods will benefit from controlling the amount of knowledge learned. Maintaining a model of the trend would allow a control system to determine the necessary amount of

[2]The Breast Cancer (BC), Flare and Voting (Vote) domains are from the UC Irvine machine learning databases.

Table 3: Actual peak performance minus MBAC peak performance for empirical learners using the parabolic model. The standard deviation of the parabolic model appears in parentheses.

| | BC | Flag |
|---|---|---|
| ID3 | 0.000(0.042) | 0.003(0.063) |
| PLS1 | 0.012(0.037) | 0.002(0.041) |
| BP4 | 0.020(0.138) | 0.031(0.085) |
| BP8 | 0.003(0.077) | 0.043(0.096) |
| BP16 | 0.018(0.112) | 0.006(0.093) |
| | Flare | Vote |
| ID3 | 0.004(0.017) | 0.019(0.076) |
| PLS1 | 0.003(0.017) | 0.003(0.058) |
| BP4 | 0.028(0.189) | 0.004(0.028) |
| BP8 | 0.027(0.163) | 0.001(0.035) |
| BP16 | 0.028(0.148) | 0.004(0.020) |

learned knowledge and prevent overfit. Furthermore, the model can also predict the achievable performance of the learning method at the peak of the performance response. This value can be used to compare alternative learning methods for a learning task [Holder, 1991b]. The MBAC (Model-Based Adaptive Control) system uses a parabola to model the performance response trend. Using this model, MBAC is able to control the learner and achieve increased performance [Holder, 1991a].

Avoidance of the general utility problem requires that the amount of learned knowledge correspond to the peak of the performance response. The results in Table 3 evaluate MBAC's ability to converge to this peak. The experimental method first generates ten performance response curves. Each response curve is the result of learning on a randomly-selected set of training examples and testing on a randomly-selected set of test examples. Given the data points from the ten curves, MBAC computes the parabolic and determines the number of transformations (amount of learned knowledge) needed to reach the peak of the instantiated models.

Next, the experimental method generates ten testing response curves using the same technique described above on another ten randomly-selected training and testing sets. The method finds the actual peak $P_A$ of the average of the ten response curves. The method measures the performance $P_M$ along the testing response curve corresponding to the number of transformations suggested by the model. $P_M$ is the performance along the testing response corresponding to the peak of the instantiated parabola.

Table 3 lists the average difference $P_A - P_M$ for the empirical learners and indicates the standard deviation of the model in parentheses.[3]. Comparison of this difference with the standard deviation of the parabolic model indicates whether the error in $P_A - P_M$ is within the error (one standard deviation) of the model. The standard deviation of the parabolic model is the average absolute difference between each point used to fit the model and the estimated parabola. Table 3 indicates that the difference between actual peak performance and MBAC peak performance is within one standard deviation of the model. Thus, the parabolic model's predicted peak in the performance response is correct within the error of the model.

Although the parabolic model improves performance, it does not always accurately predict the peak of the performance response. A more formal model is necessary that includes parameters based on the current learning task, such as number of instances or size of the instance space. The commonality of the general utility problem trend suggests that a general formal model may encompass several learning methods.

## 7 FORMAL MODELS

Although most of the formal analysis has occurred on empirical learning, analysis shows that if the amount of learned knowledge corresponds to the complexity (specificity) of the induced hypothesis, then the performance response trend results from two components affecting accuracy: accuracy on the training data and accuracy on the testing data.

### 7.1 EMPIRICAL LEARNING

The CART program (Classification and Regression Trees) developed by Breiman *et al.* [1984] is another splitting method for inducing decision trees similar to ID3 [Quinlan, 1986] and PLS1 [Rendell, 1983]. The emphasis of this treatment of CART is not the details of the method, but a statistical analysis of the performance response (see appendix to Chapter 3 in [Breiman *et al.*, 1984]). Breiman *et al.* show that the shape of the performance (accuracy) response is the result of a tradeoff between bias and variance. *Bias* expresses the degree of fit of the decision tree to the classification surface (training instances). A low bias (many small hyper-rectangles) is preferred to a high bias (few large hyper-rectangles), because low bias allows a more precise fit to the data. However, a low bias increases the likelihood that hyper-rectangles produce classification errors due to a majority of the wrong

[3]BP$n$ stands for BackProp with $n$ hidden units

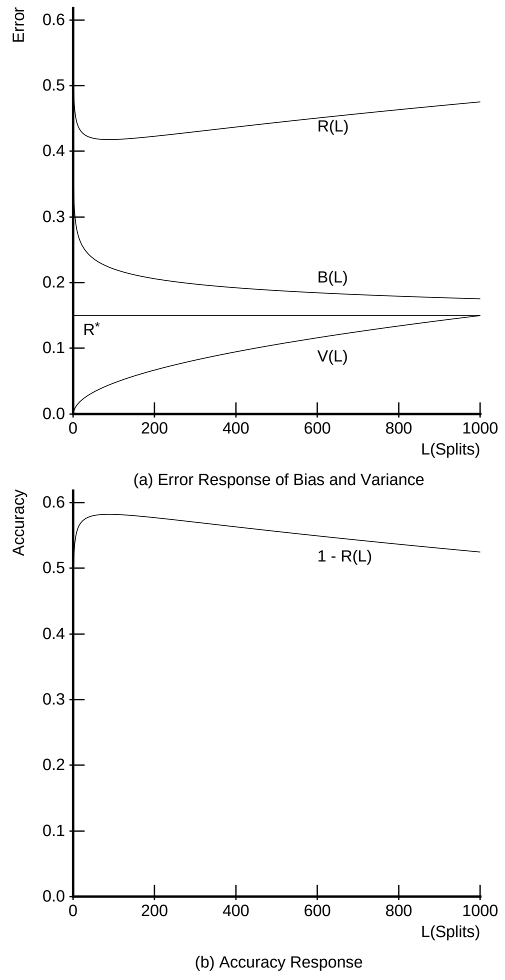


Figure 8: Performance response curve derived by Breiman et al. [1984] for decision tree induction.

class. Breiman *et al.* refer to this source of classification error as *variance*. This is not variance in the statistical sense of the expected value of the squared error, but an estimate of the discrepancy of the classification error from the Bayes error.

The analysis expresses the bias and variance in terms of the number of leaves $L$ in the decision tree. Assuming binary splits at each node of the tree, the number of splits is $L-1$. Therefore, the behavior of the bias and variance as the number of splits increase will be similar to the behavior as $L$ increases. The expression for the classification error $R(L)$ in terms of the bias $B(L)$ and the variance $V(L)$ is

$$R(L) = B(L) + V(L) + R^* \qquad (1)$$

where $R^*$ is the Bayes optimal classification error. Breiman *et al.* derive the following constraints on the bias $B(L)$ and the variance $V(L)$:

$$B(L) \leq \frac{C}{L^{2/M}}, \quad V(L) \leq \sqrt{\frac{L}{N}}, \quad V(L \simeq N) \leq R^*$$

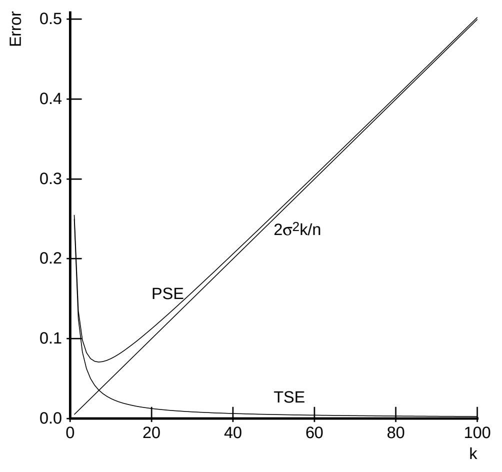
where $C$ is a constant, $M$ is the dimension of the instance space (i.e., number of features used to describe the training instances), and $N$ is the number of training instances. These expressions are for the classification error. As predicted, the bias decreases rapidly for small $L$ and more slowly as $L$ increases. The variance increases slowly as $L$ increases. When $L \simeq N$ and each hyper-rectangle contains one training instance, the variance is bounded by the Bayes error $R^*$.

Equation 1 is an expression of the classification error response curve. Figure 8a plots the bias $B(L)$, variance $V(L)$, Bayes error $R^*$, and estimated classification error $R(L)$ from Equation 1, where $C = 0.35$, $M = 20$, $N = 1000$ and $R^* = 0.15$.[4] The plot extends from $L = 0$ to $L = N = 1000$; however, the stopping criteria of actual decision tree induction programs would discontinue splitting at a point much less than $N$. For comparison to previous response curves, the error response curve is subtracted from one to yield the accuracy response curve in Figure 8b. The similarity of this performance response to that of Figure 1 supports the existence of a single peak and the inevitability of overfit in splitting algorithms without appropriate stopping criteria or post-pruning techniques. Maximizing performance while avoiding overfit requires the determination of the number of splits $L$ corresponding to the peak of the performance response.

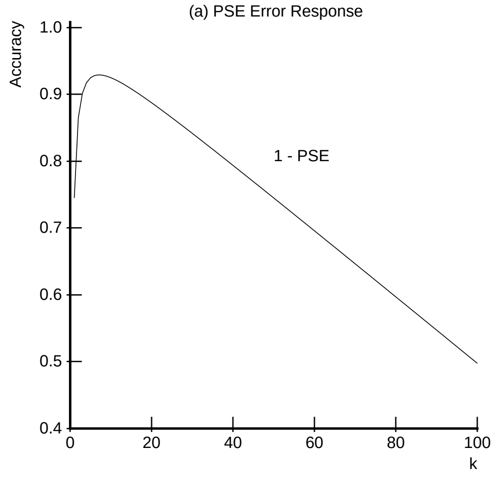
A similar analysis applies to agglomerative methods. Each time a splitting method makes a split in the decision tree, the resulting DNF expression of the hypothesis replaces a single disjunct with two, more specific disjuncts (assuming binary splits). Therefore, adding a disjunct to the DNF hypothesis in an agglomerative method is analogous to making a split in a splitting method. The above definitions of bias and variance apply directly to the agglomerative case. Decreasing the bias increases the number of disjuncts until each disjunct describes a single training instance. Variance, the error due to incorrect classifications made by the disjuncts on unseen testing instances, increases with decreasing bias. The corresponding expressions for bias and variance as a function of the number of disjuncts have a similar behavior as those depending on the number of splits, and the agglomerative performance response follows the behavior in Figure 8.

The performance response trend in neural networks is also the sum of the performance on training data

[4]For binary decision trees, $L \leq 2^M$.



(a) PSE Error Response



(a) PSE Accuracy Response

Figure 9: Performance response curve derived by Barron [1984] for a network as a function of the number of coefficients k in the network model.

and the performance on testing data. Before relating performance to the number of cycles, this analysis first considers the number of coefficients in the model represented by the network. Network models with increasing complexity (e.g., number of hidden units) have higher numbers of coefficients. If the complexity of the network is higher than the complexity of the problem, the complex network will use the overabundance of coefficients to overfit the training data.

Barron [1984] derives an expression for the predicted squared error (PSE) of the network that depends on the number of coefficients. The expression for PSE is

$$\text{PSE} = \text{TSE} + 2\sigma^2 \frac{k}{n}$$

TSE stands for the squared error of the network on the training examples, $\sigma^2$ is a prior estimate of the true error variance, $k$ is the number of coefficients in the network model, and $n$ is the number of training examples. One estimate of the true error variance $\sigma^2$ is the actual variance in the training data. The second term of PSE serves as an overfit penalty for excessively complex models. Assuming TSE has a similar behavior as the bias in Figure 8, Figure 9a plots the two components of PSE and their sum as a function of $k$ for $n = 100$. Figure 9b plots the same function subtracted from one to show the same orientation of previous performance responses. The resulting curve confirms the general utility problem trend in networks.

The above analysis uses the number of coefficients $k$ as a measure of the amount of learned knowledge; however, the performance responses for neural networks use the number of cycles as a measure of the amount of learned knowledge. One possibility for relating the number of cycles to the number of coefficients $k$ is to show that the higher numbers of coefficients in the network model are not used (negligibly small) until later cycles. In other words, earlier cycles use fewer coefficients to learn global patterns in the training data. As the cycles continue, the network attempts to reduce the error on noisy (or anomalous) training data by utilizing more coefficients to fit a higher-degree function to the training data.

The following argument derives from our observations of the error back-propagation method during the course of learning. The observations reveal that the network quickly learns to correctly classify a majority of the training data and uses the remaining cycles to learn a smaller subset of the training data. One cycle involves a single pass through the entire set of training data, where each incorrect classification initiates the error back-propagation procedure to update the weights toward correcting the error. Initially, a majority of the weight updates are due to errors on the training data representing the global patterns (the more prevalent data). After the network learns these global patterns, the majority of weight updates are due to errors on less prevalent patterns in the training data. One possible interpretation of this behavior is that later cycles attempt to fit higher degrees of the function represented by the training data. If this interpretation holds[5], then as the number of cycles increases, so does the degree (complexity) of the hypothesis learned by the network. Therefore, roughly similar behavior to that of Figure 9 will exist if the number of cycles replaces $k$ along the amount of learned knowledge axis.

[5]Observations by Mozer and Smolensky [1989] support a similar interpretation, but more experimentation is necessary to confirm the reason for this behavior.

### 7.2 ANALYTICAL LEARNING

An analytical learner is similar to an empirical learner in that both seek a concept that maximizes performance. The concept sought by an analytical learner is a set of macro-operators or control rules minimizing the time taken by the problem solver to solve problems from some domain. If the set of problems used to train the learner is not representative of the distribution of problems in the domain, then the performance obtained for the training examples may degrade performance on the testing examples for reasons similar to overfit in empirical learners. However, the factors underlying the performance degradation are different from those affecting empirical learners. Minton [1990] identifies three ways in which macro-learning affects the problem-solving performance of an analytical learner. A simple quantification of these three components in terms of branching factors behaves similarly to the general utility problem trend [Holder, 1991a], but the exact relationship between macro-operator (or control rule) learning and performance is not fully understood.

## 8 CONCLUSIONS

Both experimental and formal analysis verify the commonality of the performance response trend in Figure 1 for many learning paradigms. The analysis indicates that the trends result from two factors: one increasing performance and one decreasing performance. Furthermore, both analyses constrain the order of increasing the amount of learned knowledge to be from general to specific. The overall behavior of the performance response is a curve increasing rapidly to a single peak and then decreasing slowly after the peak.

Modeling the performance response trend allows the control of the general utility problem by using the model to predict the amount of learned knowledge corresponding to the peak of the performance response. Holder [1991a] describes the MBAC (Model-Based Adaptive Control) system that relies on one empirical model (parabola) for every learning method. Although MBAC improves performance by adaptively converging to the performance response peak, a formal would be more accurate due to its more direct dependence on properties of the learning task. As more formal models appear, they can replace the empirical model in MBAC to improve performance. Further analysis of the general utility problem will derive more general models encompassing multiple learning methods. These models will form the basis of a powerful control system for selecting from alternative learning methods and controlling the selected method to avoid the performance degradation underlying the general utility problem.

A utility-based model of learning based on the performance response curve is useful for predicting achievable performance and constraining the learning method to converge to the corresponding amount of knowledge. The existing models for empirical learning and the evolving relationship between empirical and explanation-based learning suggest the possibility of deriving general models of the performance response that unify the two learning paradigms. Furthermore, these models can be used to combine different learning methods into a single framework that selects the best learning method according to achievable performance on the learning task [Holder, 1991b].

**Acknowledgements**

I would like to thank Larry Rendell and the members of the Inductive Learning Group at University of Illinois for their helpful suggestions. Thanks also to Ray Mooney, Jude Shavlik and Carl Kadie for their implementations of the learning methods. This research was partially funded by the National Science Foundation under grant IRI 8822031.

## References

A. R. Barron. Predicted squared error: A criterion for automatic model selection. In S. J. Farlow, editor, *Self-Organizing Methods in Modeling*, chapter 4, pages 87–103. Marcel Dekker, 1984.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–284, 1989.

R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 4(3):189–208, 1972.

L. B. Holder. The general utility problem in machine learning. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 402–410, 1990.

L. B. Holder. *Maintaining the Utility of Learned Knowledge Using Model-Based Adaptive Control.* PhD thesis, Department of Computer Science, University of Illinois, October 1991.

L. B. Holder. Selection of learning methods using an adaptive model of knowledge utility. In *Proceedings* *of the First International Workshop on Multistrategy Learning*, pages 247–254, 1991.

R. C. Holte, L. E. Acker, and B. W. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 813–818, 1989.

E. D. Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

R. S. Michalski. How to learn imprecise concepts: A method based on two-tiered representation and the AQ15 program. In Y. Kodratoff and R. S. Michalski, editors, *Machine Learning: An Artificial Intelligence Approach, Vol III.* Morgan Kaufmann Publishers, 1989.

S. Minton. Selectively generalizing plans for problem-solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 596–599, 1985.

S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach.* Kluwer Academic Publishers, 1988.

S. Minton. Issues in the design of operator composition systems. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 304–312, 1990.

R. J. Mooney. The effect of rule use on the utility of explanation-based learning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 725–730, 1989.

M. C. Mozer and P. Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1(1):3–16, 1989.

G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–100, 1990.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

L. A. Rendell. A new basis for state-space learning systems and a successful implementation. *Artificial Intelligence*, 20(4):369–392, 1983.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing, Volume 1*, chapter 8, pages 318–362. MIT Press, 1986.

M. Tambe and P. Rosenbloom. Eliminating expensive chunks by restricting expressiveness. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 731–737, 1989.

J. Yoo and D. Fisher. Concept formation over problem-solving experience. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 630–636, 1991.