The Experimental Study of Machine Learning

Pat Langley (Langley@ptolemy.arc.nasa.gov)
AI Research Branch, Mail Stop 244-17, NASA Ames Research Center, Moffett Field, CA 94035 USA

Dennis Kibler (Kibler@ics.uci.edu)
Department of Information & Computer Science, University of California, Irvine, CA 92717 USA

1. Science and Observation

Machine learning is often characterized as a scientific discipline, and this suggests that we incorporate knowledge of science and its methods into the goals and techniques of the field. Research in AI and cognitive science further suggests that one can view science as a search through a space of theories that requires two active components – a *generator* and a *test*. The generator produces new theories or variants on existing theories, whereas the test yields information concerning the quality of theories. Science incorporates a variety of tests that guide the theory-generation process. These include evaluation metrics like elegance and internal consistency, which it shares with other intellectual endeavors such as mathematics and philosophy.

However, science diverges from philosophy in its emphasis on *observation*. No matter how elegant or consistent, a theory that disagrees with the data must be rejected or improved. Observation acts as the most important factor in the evaluation function that directs scientists' search through the space of theories. Hawking (1988) holds a similar view on the evaluation of scientific knowledge:

> A theory is a good theory if it satisfies two requirements: It must accurately describe a large class of observations ..., and it must make definite predictions about the results of future observations.

Thus, he distinguishes between two sorts of observations: those made before the theory is forwarded (which it must cover) and those made after its generation (which it must predict). This distinction reflects the two roles played by empirical results: the suggestion of new candidate theories and the evaluation of existing ones.

The success of physics, perhaps the most well-developed scientific discipline, should clarify the importance of observation. This field is primarily concerned with understanding the nature of the physical world – the structure and processes that govern matter and energy. This shared goal holds physics together as a field, but its progress derives primarily from its continued, repeated testing of theories against observation. Data play a central role in selecting among competing theories, and anomalies suggest improvements on incorrect theories. Over time, old theories are rejected and new ones emerge with higher predictive accuracy and greater generality.

2. The Role of Experiments in Machine Learning

Machine learning is another science, albeit a much younger one than physics. Our discipline is primarily concerned with understanding the computational mechanisms that underlie learning, and this shared purpose holds machine learning together as a coherent field. Like physics, the success of machine learning as a scientific discipline will rest on its ability to combine theory and observation, using data to drive theory selection and revision.

The field of machine learning focuses on intelligent artifacts – systems created by the researchers who study them. Thus, it constitutes what Simon (1969) has called a *science of the artificial*. As such, there is a temptation to emphasize formal analysis and theoretical approaches. Indeed, considerable progress has recently occurred on the theoretical front,[1] both in formalizing the nature of learning algorithms and in characterizing their behavior. In this view, machine learning is primarily a mathematical science.

Despite this progress, many learning algorithms are too complex for formal analysis, at least at the level of generality assumed by most theoretical treatments. As a result, empirical studies of the behavior of machine learning algorithms must retain a central role. Fortunately, the artificial nature of learning algorithms allows control over a wide range of factors, making it more akin to experimental disciplines such as physics and chemistry than to observational sciences such as astronomy or sociology. It is this view – machine learning as an experimental science – that we pursue in this paper.

The goal of scientific experimentation is to better understand a class of behaviors and the conditions under which they occur. Ideally, this will lead to empirical laws and theories, as well as to tests of those theories. In our field, the central behavior is learning. The conditions involve the algorithm employed, the domain knowledge, and the environment in which learning occurs. Lacking a formal analysis, an implemented learning algorithm is necessary but not sufficient for understanding – one should also attempt to specify when it operates well and the reasons for that behavior. Such generalizations provide the raw material for forming and testing theories of machine learning. Moreover, they can suggest improved algorithms that exhibit more desirable learning behaviors.

As normally defined, an *experiment* involves systematically varying one or more *independent* variables and examining their effect on some *dependent* variables. Thus, a machine learning experiment requires more than a single observation of a system's behavior; it requires a number of observations made under different conditions. In each case, one must measure some aspect of the system's behavior for comparison across the different conditions.

We have organized the remainder of the paper in these terms. We begin by examining some dependent variables that can be used in the experimental study of learning algorithms. After this, we address two broad classes of independent vari-

1. Kearns, Li, Pitt, and Valiant (1987), Dietterich (1990), and Haussler (1990) provide informative reviews of progress in the area of learnability theory.

ables – aspects of the algorithm and aspects of the environment. Finally, we consider some issues in the design and execution of experiments. Many of our suggestions are similar to the excellent points made by Cohen (1991) in his discussion of artificial intelligence, but they seem worth instantiating for the field of machine learning.

3. Dependent Measures of Learning

Most definitions of learning rely on some notion of improved *performance*. Thus, various performance measures are the natural dependent variables for machine learning experiments, just as they are for studies of human learning. Other measures, like 'understandability' of the acquired structures, may also be informative, but these are not relevant unless accompanied by performance improvement.[2] In some cases, intuitively plausible learning methods actually lead to *worse* performance (Minton, 1985), so performance measures are central to evaluating almost any learner's behavior.

3.1 Measures of Performance

Many measures of performance are possible. For supervised concept induction tasks, in which each instance has an associated class name, the most obvious metric is the percentage of correctly classified instances (Quinlan, 1986). One cannot use this dependent variable for unsupervised induction tasks like conceptual clustering, since no class name is available. However, one can replace it with a more general measure – the ability to predict a missing attribute's value, averaged across all attributes; Fisher (1987) refers to this performance task as *flexible prediction*.

More complex domains require more sophisticated measures of performance. For grammar-induction tasks, one can record the percentage of correctly parsed sentences and the percentage of correctly rejected non-sentences. For problem-solving domains, one can examine the percentage of problems solved or the quality of the resulting solution paths (Langley & Drummond, 1990). One can also measure the total CPU time or number of nodes considered during search (Minton, 1985). The last two metrics are concerned with *efficiency* rather than correctness, and thus seems appropriate for explanation-based approaches to learning, which have been largely concerned with the compilation of knowledge rather than its acquisition (Mitchell, Keller, & Kedar-Cabelli, 1986; DeJong & Mooney, 1986).

Given a particular performance criterion, one must implement this measure in some fashion. In nonincremental settings, one can present the learning system with a *training* set and then evaluate its performance on a separate *test* set. This is an important methodological point. The goal of learning is typically to use acquired knowledge to aid behavior in *novel* situations, not on problems that have been encountered in the past. Also, because any given set of instances may not be representative of the

2. As in psychology, we make a clear distinction between performance – an agent's behavior at a given instant in time – and learning – the change in an agent's performance over time. In this framework, the phrases *learning performance* and *performance of a learning system* are oxymorons.

domain, it is important to average over the results of runs on many sets of training and test problems that have been selected randomly from those available.[3]

One can use a similar scheme to study incremental systems, which process one experience at a time. In this case, one presents training instances one at a time and, after every nth instance, turns learning off and runs the system on a separate test set. Alternatively, one can treat each instance first as a test datum and then as a training datum, but this requires that one run the system more times. In either case, the result is a *learning curve* that shows change in performance as a function of the number of instances encountered. Although learning curves are informative, one can also condense this information into more succinct summary measures, such as the asymptotic performance and the number of instances needed to reach this asymptote.

When studying incremental methods, it is important not only to average over different training and test sets, but also over different orders of the training instances, since this can influence the course of learning in most incremental systems. However, in some contexts a researcher may be interested in examining order effects themselves, in which case he or she should systematically vary this factor like any other independent variable.

3.2 Performance in Classification Domains

Fisher (1987) has described COBWEB, an incremental unsupervised algorithm for inducing probabilistic concepts. The system organizes its acquired knowledge as a hierarchy of concepts, which it modifies with each training instance. However, our concern here is not with Fisher's algorithm but with experimentation, so let us consider a recent study of this system by McKusick and Langley (1991). Figure 1 presents a learning curve for COBWEB in a particular classification domain.

The data used in this study were collected by Schlimmer (1987) from the *Congressional Quarterly*. They describe votes of the 435 members of the 1984 U.S. House of Representatives on 16 issues, such as aid to El Salvador, funding for the MX missile, and duty-free exports. Thus, there are 435 instances, each consisting of 16 Boolean attributes that specifies whether a given House member voted 'yea' or 'nea'. Each instance also falls into one of two classes, of which 267 were Democrats and 168 were Republicans. Although Fisher's system was designed for unsupervised tasks, it can also learn from such supervised data, so the dependent measure here was predictive accuracy on the class label, rather than Fisher's measure of flexible prediction.

McKusick and Langley presented COBWEB with a random sample of 100 training instances from this domain and tested it on a separate set of 25 randomly selected instances. After each training case, learning was disabled and the system was asked to predict the class label for each test case. The percentage of correctly classified

3. One can achieve similar effects through *cross-validation* studies, in which one iterates through each of N available instances, in each case running the system on the other $N - 1$ instances and using the selected instance to test performance. One then averages the results for all N runs to estimate typical performance.

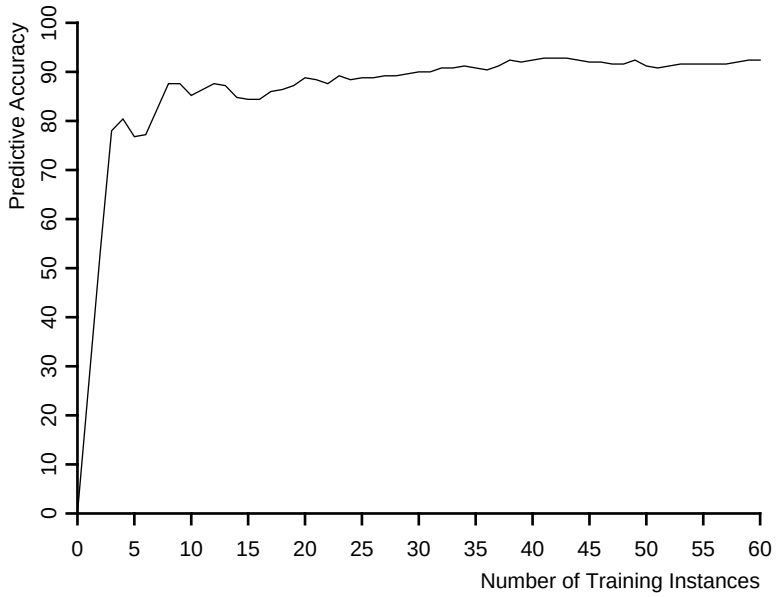

Figure 1. Learning curve for Fisher's COBWEB algorithm on Congressional voting records, as reported by McKusick and Langley (1991).

instances was recorded, learning was enabled, the system was presented with the next training instance, and the cycle continued. The learning curve shown in Figure 1 was averaged over ten runs based on different random orderings of the training data. Thus, each point on the curve shows the average percentage of the test set that COBWEB correctly classified.

The shape of the curve reveals that most learning occurs rather early. Asymptotic accuracy is approximately 92%, yet COBWEB reaches the 85% level after fewer than ten training instances. The system remains stable at this point for some time, then rises to above 88% around 20 instances. Slight improvements occur with additional instances, but the most important learning seems complete by this point. These results contradict some claims (e.g., Mitchell et al., 1986) that inductive instances require very many instances to acquire useful knowledge. But it also suggests that the behavior of House members may be quite regular and thus simple to induce. As a result, it can be dangerous to draw conclusions about the behavior of a learning algorithm from studies with a single domain. We will return to this issue later.

3.3 Performance in Problem-solving Domains

Learning curves can also be used to examine performance improvement in problem solving. Let us consider a study by Gratch (1991) using a reduced version of PRODIGY-EBL (Minton, 1990), a well-known and successful algorithm for acquiring search-control rules. The learning system employs an explanation-based method to transform problem-solving traces into rules for selecting operators, states, and goals during planning. Minton's system then uses these rules to constrain search on new problems.

Gratch's study uses CPU time as the measure of problem-solving performance. One could examine the number of search nodes considered in solving problems, but as Minton has shown, the amount of search is only one facet of problem-solving efficiency. His definition of the *utility* of acquired knowledge also includes the cost of applying that knowledge in controlling search, and CPU time takes this into account. Langley and Allen (in press) use a related measure, the total number of unifications required to solve a set of problems, which is less dependent on implementation and machine. They also examine both search nodes and match cost in an attempt to determine the source of power or difficulty.

The independent variable in Gratch's experiment is the number of training problems on which the system has practiced. He generated 220 problems from a special variant of the blocks world (described by Etzioni, 1990), dividing these into 100 training tasks, 20 'settling' problems, and 100 test cases. After every ten training problems, Gratch disabled the PRODIGY's explanation-based learning component and ran the system on the settling problems, which it used to gather statistics about the utility of individual control rules. During this stage, PRODIGY deleted rules that appeared to increase the overall cost of planning. After this, the experimenter disabled this facet of learning as well and measured the CPU time required to solve all the test problems.

Figure 2 shows the learning curve for this domain. The times are averaged over ten random orderings of the training and settling problems, since order effects can occur in problem-solving domains as easily as in classification tasks. The results are intriguing. The search-control knowledge that PRODIGY acquired actually *increases* its problem-solving time. This begins to decrease after the initial large rise, but it never quite returns to the level that existed before learning. Presumably, this effect occurs because the cost of matching their complex conditions more than offsets the savings due to reduced search, even though the settling phase was designed to avoid this problem.

For our purposes, these results demonstrate the clear need for experimental studies of learning's effect on performance. Without such evaluation, one cannot know whether learning is actually beneficial. With such experiments, one can identify the source of the degradation and modify the learning scheme to improve performance. Fortunately, this is not the complete story on PRODIGY; in fact, Etzioni carefully designed this particular variant of the blocks world to encourage PRODIGY to acquire expensive search-control rules. We will return to this point in Section 5.

Segre, Elkan, and Russell (1991) have noted an important complication in the experimental study learning in problem-solving domains. Most problem solvers include some computational limit and give up on a problem when they exceed it. Thus, reporting only efficiency results can be misleading; it is essential to include information about the percentage of test problems that the system has solved. Gratch was careful to use only problems that PRODIGY could solve within its computational limits, but this may not be practical for some real-world problems. Also, in some domains, the quality of problem solutions can also be important. Langley and Drummond (1990) suggest some ways in which to instantiate this dependent variable.

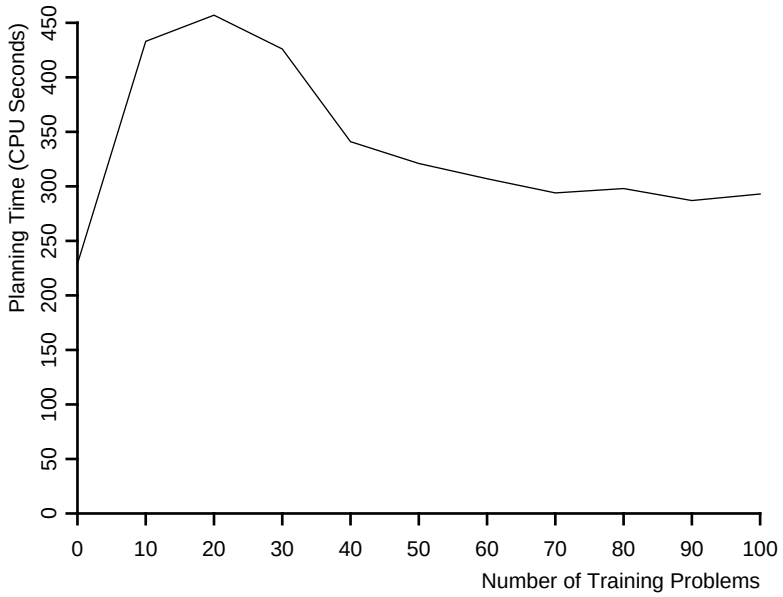

Figure 2. Learning curve for a reduced version of PRODIGY on problem-solving tasks from a variant of the blocks world, as reported by Gratch (1991).

4. Varying the Learning Method

One of the most difficult problems confronting psychology is teasing apart the relative effects of heredity and experience, of 'nature' and 'nurture'. Machine learning is more fortunate, in that it can experimentally control a learning system's 'innate' features (nature) and the training instances it encounters (nurture). Here we examine methods for evaluating the effect of system characteristics, delaying the role of experience until the following section.

The obvious way to examine the influence of system features on behavior is to compare different algorithms on the same task. That is, one runs two or more learning systems on a given domain, measures their performance on the same test cases, and compares the results. Until recently, such comparative studies were rare in the literature, but now they have become almost the default, and the availabilty of standard databases has provided a variety of domains to use in such experiments.[4]

4.1 Gross Comparisons of Learning Methods

Comparative studies come in a variety of forms. If one's goal is a computational model of human learning, then one should compare the algorithm's behavior with that of human learners. For example, children pass through a number of 'stages' in their acquisition of language, and one can compare the model's learning curves

4. In fact, most of domains we mention in this paper are available by `ftp` from `ics.uci.edu` using the account and password `anonymous`. The various data sets reside in the directory `pub/machine-learning-databases`.

Table 1. Percentage accuracies and training times for three induction algorithms in diagnosing soybean diseases (Shavlik, Mooney, & Towell, 1991).

ALGORITHM	ACCURACY (TRAINING)	ACCURACY (TEST)	CPU TIME (TRAINING)
PERCEPTRON	100.0 ± 0.0	92.9 ± 2.1	35.8 ± 5.2
ID3	100.0 ± 0.0	89.0 ± 2.0	161.0 ± 8.9
BACKPROP.	99.9 ± 0.2	94.1 ± 2.5	5260.0 ± 7390.0

with that of children (Langley, 1982). Similarly, a model of skill acquisition should account for the widely observed 'power law' of learning (Rosenbloom & Newell, 1987). Many factors affect human learning, making its experimental study difficult, but the psychological literature is filled with studies awaiting computational explanations.

More often, a machine learning researcher is interested in an algorithm's behavior for its own sake. However, even when studying an individual learning method, it is best to place that method's behavior in context. One can usually compare the system's performance to that of a 'straw algorithm' that uses a simple-minded strategy. For instance, in classification domains one can use an algorithm that predicts the most frequently occurring class. If this covers 90% of the instances, then a more sophisticated learner that achieves 91% accuracy is not impressive, and should be examined for ways to improve its learning ability. This approach is different from using a nonlearning performance system to establish a baseline, but the spirit is similar.

Shavlik, Mooney, and Towell (1991) provide an excellent example of comparing alternative learning methods on a diagnostic task originally reported by Reinke (1984). The goal is to classify soybean plants into one of 17 different disease categories based on 50 nominal (symbolic) attributes, such as weather, time of year, and characteristics of leaves and stems. There are 17 examples of each disease, giving a total of 289 cases. Shavlik et al. randomly selected two-thirds of these as training instances, reserving the remainder as test cases. They averaged their results over ten such partitions of the soybean data set.

The authors examined the behavior of three algorithms on this domain. The PERCEPTRON algorithm (Rosenblatt, 1962), one of the simplest forms of connectionist learning methods, represents knowledge as a single linear threshold unit. This results in the well-known limitation that it can only discriminate concepts that are linearly separable, that is, which can be separated by a single hyperplane drawn through the instance space. Thus, Shavlik et al. included this method as a straw algorithm. However, they also studied the behavior of BACKPROPAGATION (Rumelhart, Hinton, & Williams, 1986), a more popular connectionist technique that supports learning in multi-layer networks. Finally, they examined Quinlan's (1986) ID3, a widely-used algorithm for inducing decision trees.

Table 1 summarizes the behavior of these induction algorithms on soybean domain along three dimensions – classification accuracy on the training set, accuracy on the test set, and training time (measured in CPU seconds). All systems achieve perfect accuracy on the training set, but only the PERCEPTRON method, which has limited representational power, might have performed poorly on this front. A system that simply remembers all observed instances could fare as well; this is the reason accuracy on training cases is seldom useful. As expected, BACKPROPAGATION requires considerably more computation than either ID3 or the simpler connectionist scheme. This is *not* a performance measure but an indication of learning cost. Nevertheless, it can be an important factor on sizable domains, and may be worth reporting.

The surprise comes when we examine classification accuracy on the test cases. The PERCEPTRON method appears to perform slightly worse than BACKPROPAGATION, but it does much better than expected given the abuse it has taken over the years (e.g., Minksy & Papert, 1969). Even more unexpected, this technique actually has higher accuracy than ID3, which employs a much more sophisticated induction technique. The straw program refused to be blown over in this domain, suggesting it deserves more attention than it has traditionally received. However, it would be premature to conclude that one technique is superior to another based on their behavior in a single domain, as we will argue in Section 5.

Note that the table presents not only the means for each combination of algorithm and dependent measure, but the standard deviations as well. One can use this information, together with the number of runs, to determine the probability that the observed differences are due to chance. Shavlik et al. report that, using a t test, the difference between the PERCEPTRON and ID3 accuracies is significant at the 0.01 level. This means that the probability is greater than 99% that this difference did *not* occur by chance. However, they also report that the apparent difference between BACKPROPAGATION and PERCEPTRON is not significant at this level. In the absence of additional evidence, one must conclude that they are effectively equivalent on this domain. Significance tests are especially important in comparative studies and, although they are reported in few of the studies we will describe, we encourage researchers to use them whenever possible.

4.2 Parametric Studies of Learning Methods

Given the complexity of many learning algorithms, one may not be satisfied with comparisons between entire systems. The goal of experimentation is not to blindly label one method as superior to another, but to understand the *reasons* for behavioral differences. Finer-grained studies can be very useful in pursuit of this end.

An obvious approach is to examine the effect of parameters occurring in a system. In such cases, one can determine the importance of a parameter on algorithm behavior by systematically varying its settings and observing the results. Ideally, behavior will be 'acceptable' within a wide range of parameter values, with the system's behavior changing slowly as the parameter varies. Alternatively, one might identify an optimal setting that holds across different domains.

Clark and Niblett (1989) provide an example of a parametric study. They describe CN2, an algorithm that combines aspects of both Quinlan's (1986) ID3 and Michalski and Chilausky's (1980) AQ11. The system carries out a beam search through a space of rules, guided by an evaluation function based on information theory. Once it has decided on a rule to cover some training instances, it removes these and iterates to find additional rules. Because CN2 uses a statistical test to determine when to stop adding conditions and rules, it should be able to avoid overfitting the training data in noisy domains.

However, any statistical test requires one to specify some level of significance for making decisions. Clark and Niblett ran CN2 with different significance levels on a medical diagnosis task that involved classifying patients as either healthy or having some form of lymph cancer. They carried out five runs on this domain, in each one randomly selecting 70% of the 148 instances as training cases and the rest as test cases. Using 90% as the significance level, CN2's average accuracy on the test instances was 78%. In contrast, at the 95% and 99% levels it was 81% and 82%, respectively. Thus, the parameter setting appears to have some effect, but the difference is not a major one. Of course, one cannot tell the actual amount of noise in such a real-world domain, as we discuss further in Section 5.

Robertson and Riolo (1988) report another parametric study, in this case using a genetic algorithm. This class of methods retains a number of rules in memory, which compete for the chance to generate offspring through operations analogous to genetic mutation and crossover. The authors hypothesize that one factor in genetic learning is the number of copies retained of a given rule. Thus, they test their CFS system with different limits on this number, measuring its behavior on a task that involves learning to predict sequences of symbols. The results suggested a 'U-shaped' curve, in which performance increased with the number of copies allowed, but only up to a certain point, beyond which it dropped again. Detecting such regularities can let one fine tune a learning system to increase its learning rate or asymptotic performance.

4.3 Lesion Studies of Learning Components

Parametric studies are not the only means of exploring the sources of power in an intelligent system. One of the most common techniques in neuroscience involves the excision of a well-defined area of the brain to determine its role in behavior, and there is no difficulty in adapting this notion to the study of artifacts.

Many machine learning systems contain a number of independent components, and each component's usefulness can be studied through 'lesion' experiments.[5] In other words, one runs the system with and without a given component, measuring the difference on some performance dimension. If a component does not aid the overall learning process, then it can be safely omitted from the system.

For example, Schlimmer (1987) describes a lesion study using an artificial task that involves predicting the output of a 1×2 multiplexer. He designed this experiment

5. Cohen and Howe (1988) have referred to such experiments as *ablation* studies.

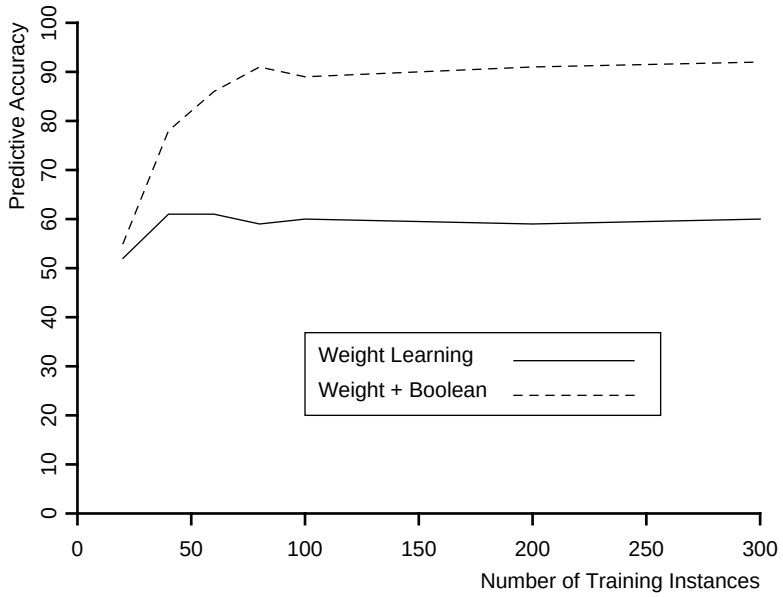

Figure 3. The effect of lesioning STAGGER's Boolean mechanism for creating new conceptual components (Schlimmer, 1987).

to evaluate the relative impact of two separate learning components in his STAGGER system. The first component assigns weights to the various components in a manner reminiscent of the PERCEPTRON algorithm, but using information about conditional probabilities to speed learning. The second component augments this process by forming logical combinations (conjuncts, disjuncts, and negations) of features that have high diagnostic power, which are then used by the weight-learning routine.

Figure 3 presents the learning curves for the two experimental conditions, which demonstrate the advantage of augmenting weight learning with a method for introducing Boolean features. In this case, the main difference lies in STAGGER's asymptotic accuracy, which is much greater for the combined method. However, the two-component version does take somewhat more training instances to reach its higher asymptote than does the weight-learning method to reach its lower one.

Although this experiment focuses on a knowledge-lean method, lesion studies also seem well suited to knowledge-intensive learning methods. One might 'lobotomize' a system by removing some of its knowledge or some of its mechanisms, then observe the effect on learning. For instance, in explanation-based approaches, overly specific domain theories would presumably lead to less transfer and thus to slower learning. We will briefly describe one study involving the impact of knowledge in Section 6.

Before closing our discussion of experiments with learning methods, we should emphasize that the goal of such studies is not to demonstrate superiority of one method over another, but to increase understanding. Experiments may indeed reveal limitations of particular methods or components, but this knowledge can in turn

suggest improved versions of the initial algorithm. For example, Aha, Kibler, and Albert (1991) describe a learning algorithm that simply stores training instances and uses a nearest-neighbor technique to classify new cases. Experiments reveal drawbacks of this method, which they attempt to remedy by placing constraints on the storage of instances. Lesion studies indicate the usefulness of this extension, but further experiments suggest other problems, which they mitigate with another addition to their instance-based algorithm. This process of incremental refinement relies on understanding the *reasons* for a learning method's behavior, and this should be the primary aim of experimentation.

5. Varying Characteristics of the Domain

As we mentioned earlier, innate biases or 'nature' are not the only influence on a learning system. One must also examine the effect of experience or 'nurture' on behavior, and this means systematically varying the environment or domain in which the learner acquires knowledge. This presents the machine learning experimenter with a choice. One can employ 'natural' domains like the diagnostic tasks we examined earlier. Alternatively, one use 'artificial' domains that have been designed with specific characteristics in mind. In this section we examine these two options. As we will see, each approach has its advantages and disadvantages, and we recommend both for the experimental study of machine learning.

5.1 Studies with Natural Domains

Natural domains, such as Reinke's (1984) soybean diagnosis task, are the most obvious testbeds because they show real-world relevance. Also, successful runs on a number of different natural domains provide evidence of generality. For example, let us return to Shavlik et al.'s study, from Section 4.1, and consider it in more depth.

Table 2 presents additional results for their three algorithms on four separate classification tasks. These include the soybean domain described earlier, a task that involves predicting the winner of chess end games based on 36 high-level features, an audiology domain that requires diagnosis of 24 hearing disorders based on 58 features, and a task that involves determining whether a patient has heart disease, given eight nominal attributes and six numeric ones. The table reports only accuracy on the test sets. For comparison, we have repeated the results for the soybean domain.

Recall that on the soybean data, the knowledge induced by both the BACKPROPAGATION and PERCEPTRON methods performed better than the ID3 algorithm. However, by examining behavior across domains, Shavlik et al. demonstrated that this result is misleading. Behavior in a single domain, even a real-world one, does not necessarily generalize to other domains. On both the chess and audiology testbeds, both ID3 and BACKPROPAGATION are significantly more accurate (at the 0.05 level) than the PERCEPTRON learning algorithm, but there is no significant difference between the two more sophisticated methods. BACKPROPAGATION does significantly better

Table 2. Percentage accuracies for three induction algorithms on four classification domains (Shavlik et al., 1991).

Algorithm	Soybean Disease	Chess End Game	Audiology Diagnosis	Heart Disease
Perceptron	92.9 ± 2.1	93.9 ± 2.2	73.5 ± 3.9	60.5 ± 7.9
ID3	89.0 ± 2.0	97.0 ± 1.6	75.5 ± 4.4	71.2 ± 5.2
Backprop.	94.1 ± 2.5	96.3 ± 1.0	77.7 ± 3.8	80.6 ± 3.1

than ID3 in diagnosing heart disease, but the induced decision trees outperform the learned perceptrons in turn (at the 0.01 level).

These results make one more confident in the non-naive approaches, but one would still like to understand the reasons for ID3's poor behavior on the soybean data. One possibility is that this domain is nearly linear separable, but that the hyperplane is not orthogonal to any of the axes in the instance space. Thus, the Perceptron technique can accurately classify instances using a linear unit, whereas ID3 is forced to approximate this with a highly disjunctive decision tree, in which each terminal node is based on a small sample.

In the midst of this discussion, we should not forget one of the main points of the Shavlik et al. study. Connectionist and 'symbolic' induction algorithms, although they rely on different representations of knowledge and use different methods to acquire that knowledge, are dealing with essentially the same problem, and this means that one can compare them on the same tasks. This form of comparative study is much healthier for the field than rhetorical arguments about the limitations of existing methods and the advantages of new approaches.

Experimental studies of problem-solving systems can also use multiple domains to evaluate learning algorithms. In Section 3.3 we reviewed results from Gratch's (1991) study of Prodigy on a single domain, but in fact he examined the system's behavior on others as well. Figure 4 incorporates the learning curves for an extended version of the Strips planning domain and for the original version of the blocks world used by Minton (1990). The results here are much more encouraging, with Prodigy showing clear improvement by the tenth training problem in both cases. After this point, the system seems to have stabilized, apparently having completed its acquisition of useful search-control knowledge.

This raises issues about the reasons Prodigy encounters difficulty in the original domain we examined. As mentioned earlier, Etzioni (1990) designed this variant of the blocks world, which includes a single additional operator that lets one move two blocks at a time, to produce just such a negative effect in Prodigy. He provides an interesting analysis of the causes for the system's divergent behaviors in these domains. This technique – altering an existing domain to elicit some effect – is a powerful experimental tool, and it leads naturally into our next topic.

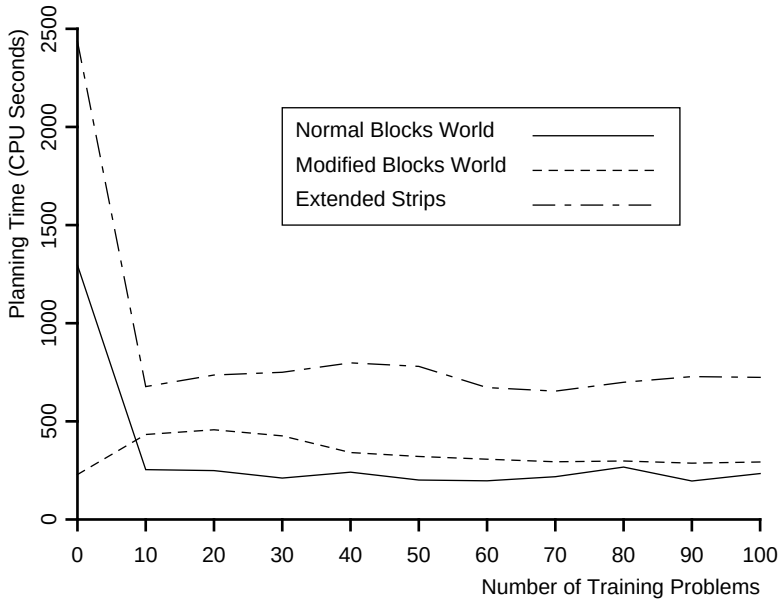

Figure 4. Learning curves for the Prodigy algorithm on three problem-solving domains (Gratch, 1991).

5.2 Noise in Artificial Domains

Studies with multiple natural domains are much more revealing than single-domain studies, in that they give evidence about the generality of learning phenomena. However, they provide little aid in understanding the effects of domain characteristics, since they do not let one independently vary different aspects of the environment. A given natural domain may be difficult along many dimensions, and one would like to know which factor is responsible for particular aspects of behavior. Artificial domains provide a way out of this dilemma by letting one control domain characteristics as independent variables. Instead of carrying out experiments with a real-world domains having unknown characteristics, one can design domain that have exactly the features one wants to study.

For instance, Breiman, Friedman, Olshen, and Stone (1984) report an artificial domain they designed to test the effectiveness of their Cart algorithm for decision-tree induction. The domain concerns a simulated LED display in which digits are described by seven Boolean features. The performance task involves classifying particular displays as one of the ten digits, which one must learn from classified training instances. However, to make the learning task difficult, they added random noise to features in the training instances, thus simulating a faulty display. To be specific, they introduced a ten percent noise level for each feature, by which they meant that each Boolean value was inverted with 0.1 probability.

Breiman et al. compared Cart's accuracy on this domain to a 'straw algorithm' that simply predicts the most frequent class. Moreover, using their knowledge about the probability of noise in features, they computed the predictive accuracy for an

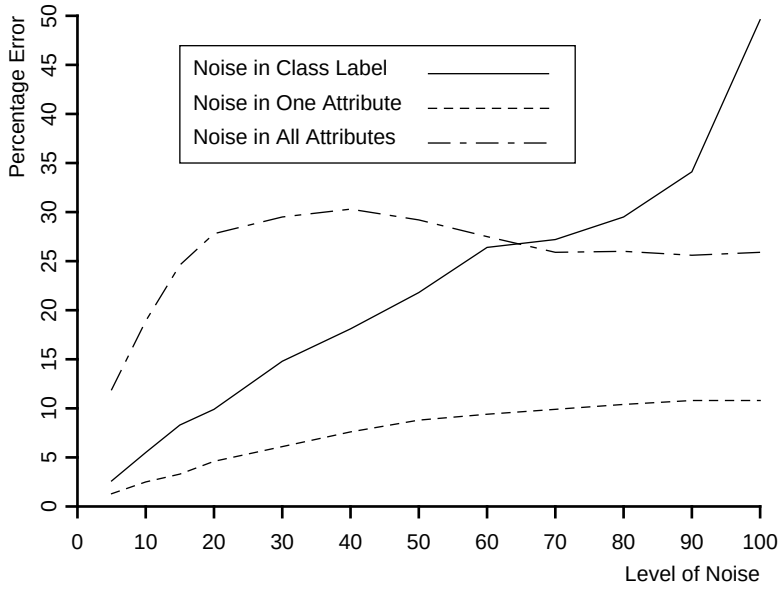

Figure 5. The effect of three types of noise on predictive accuracy in Quinlan's (1986) ID3.

optimal classifier. Thus, they established best-case learning behavior, which would be impossible for a real-world problem. For the LED domain with ten percent noise, the optimal accuracy is 74%. Because CART uses a statistical pruning technique to avoid overfitting the training data, they expected it would approach this level. Their experimental results backed this prediction, showing an accuracy of 70% for CART after 200 training instances, but only a 10% accuracy for the frequency method. Thus, their algorithm fares almost as well as possible on the LED task.

The Breiman et al. study used an artificial domain with controlled noise level, but it did not systematically vary this variable to determine the algorithms' behaviors across a range of noise levels. Quinlan (1986) provides an example of this type of experiment. He studied the classification accuracy of the trees induced by his ID3 algorithm when he varied the amount of noise in the training instances.

In particular, Quinlan examined the effect of noise when it occurred in a single (nominal) attribute, when it was present in all attributes, and when it occurred in the class label. The definition of noise in this study is somewhat different, referring to the probability of replacing the actual value with a randomly selected value (which might be still be correct). Thus, the maximum noise level is 100%, in which case the attribute or label contains no useful information. The ID3 algorithm differs from CART in its response to overfitting, halting construction of the decision tree when a statistical test indicates that the training data fail to justify further splits. However, Quinlan anticipated that this approach would let the system degrade gracefully for all three forms of noise.

Figure 5 shows the results when noise was added to training instances taken from a task involving chess end game, similar but not identical to that used in the Shavlik

et al. (1991) study. Noise in the class label degrades performance much more than noise in individual attributes, as one might expect. Also, the former changes in a roughly linear fashion, whereas the latter appears logarithmic. One might predict that noise in all attributes would make learning more difficult that noise in any single feature, including the class label. Indeed, the curve for this condition goes up rapidly,[6] but then actually decreases and levels off at a 26% error rate. Quinlan explains this surprising result by noting that, beyond a certain noise level, ID3's pruning technique leads to one-node trees that simply predict the most frequent class. The dip in the curve suggests the parameter setting for the statistical test is slightly high, allowing some overfitting to occur around the 40% noise level.

5.3 The Effect of Irrelevant Attributes

Artificial domains are also useful for examining the effect of irrelevant attributes on learning. In general, as the number of attributes increases, the number of possible concept descriptions grows exponentially (Haussler, 1987). Intuitively, learning should be more difficult in domains that contain more alternative hypotheses. If an algorithm has no way to identify relevant features early in training, increasing the number of attributes could drastically slow the rate of learning.

However, the effect of irrelevant features on any particular system is an empirical question, and many induction algorithms include techniques that should let them effectively ignore attributes that contain no useful information. For instance, Fisher's (1987) COBWEB system uses an information-theoretic evaluation function to classify instances through its probabilistic concept hierarchy. This function subtracts out the information that has already been summarized at a parent node, and thus emphasizes attributes that serve to distinguish concepts at the same level.

Gennari (1990) examined the effect of this factor on the behavior of CLASSIT, an extension of COBWEB that handles both symbolic and numeric attributes. He used a set of artificial domains that involved four separate classes, each differing in their values on four relevant numeric attributes. However, the domains varied in the number of irrelevant attributes – which have the same probability distribution independent of class – from zero to sixteen. All domains had small but definite amounts of attribute noise, and training instances were unclassified. The performance task involved predicting the numeric values of single relevant attributes omitted from test instances, and the dependent measure was the absolute error between the actual and predicted values.

Figure 6 presents the results, which are based on ten different orders of randomly generated training instances. The graph suggests that CLASSIT is robust with respect to irrelevant attributes, with an asymptote around 2.0, regardless of the number of irrelevant terms. This is close to the 'ideal' error of 0.47, which is the error for the best possible predictions that could be based on the observed training instances. CLASSIT's asymptote is also considerably less than that of a naive algorithm which

6. Note that the dependent variable reported in this case is percentage error, rather than the accuracy measure used in the previous studies we have examined.

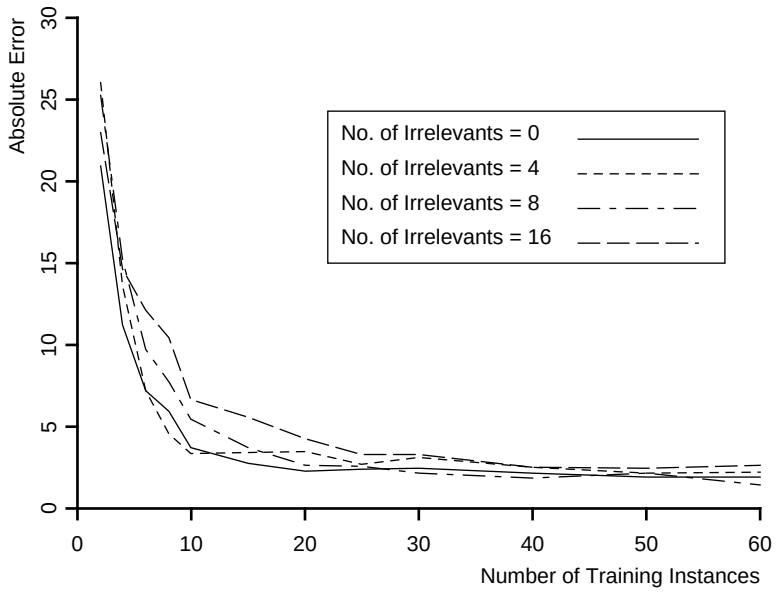

Figure 6. Learning curves for Gennari's (1990) CLASSIT on domains with varying numbers of irrelevant attributes.

simply predicts the mean value for each attribute, independent of its class. This provides another example of how one can use straw algorithms and optimal ones to calibrate learning behavior. The system's rate of learning does seem affected by the number of irrelevant attributes, but CLASSIT appears to scale well on this dimension, at least in the current domain.

Although the notion of irrelevancy has been most widely studied for inductive learning and for classification tasks, it has clear analogues in other approaches and different domains. For instance, Iba (1989) has demonstrated that promiscuous learning of macro-operators can degrade the performance of problem-solving systems. He shows that one can use statistical and other methods to eliminate such knowledge structures, retaining ones that actually reduce search on test problems. His studies focused on difficult but well-structured puzzles that had many aspects of artificial domains. Tambe, Newell, and Rosenbloom (1990) use an even more idealized search problem to study the effect of expensive rules on learning in problem solving.

Similarly, we suspect that irrelevant knowledge could slow the learning rate of analytic learning approaches by producing misleading explanations or making derivations intractable. Techniques for selecting among competing explanations and selecting likely search paths could play a similar role to the evaluation function that CLASSIT uses to ignore irrelevant attributes. Artificial domains, including both relevant and irrelevant background knowledge, are an obvious approach to testing this hypothesis. Elio and Watanabe (1991) describe one such study, in which they use carefully designed rules to study how the size and 'shape' of background knowledge affects constructive induction.

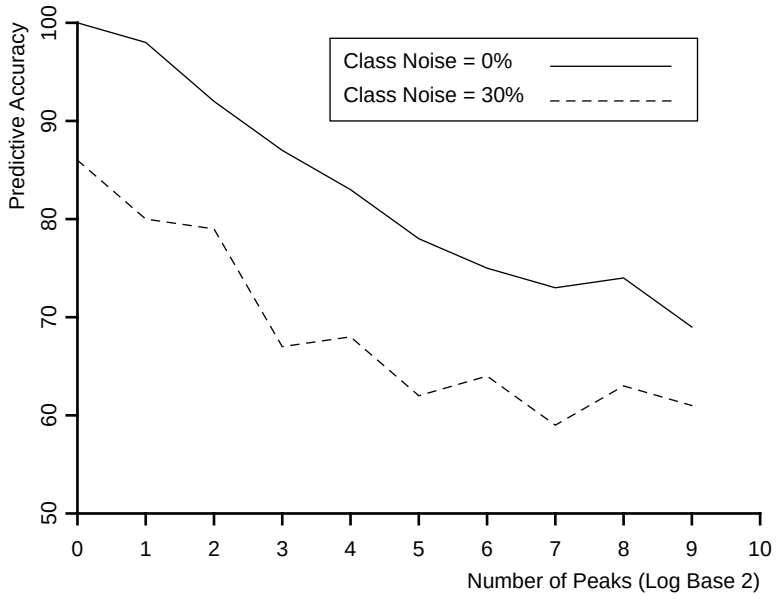

Figure 7. Predictive accuracy of the PLS system as a function of concept complexity, as reported by Rendell and Cho (1990).

5.4 The Effect of Concept Complexity

Another important characteristic of classification domains is the complexity of the concepts that describe their regularity, and one can use artificial domains to study the effect of concept complexity on learning. For instance, Langley (1987) systematically varied the number of conjunctive and disjunctive features in concepts, studying the impact of these factors on an incremental learning algorithm. Iba, Wogulis, and Langley (1998) report the results of a similar study with the Hillary system.

However, Rendell and Cho (1990) have argued that many real-world concepts are much more complex than those typically used in experimental studies. They view concepts as functions over the space of instances, measuring complexity as the number of 'peaks' or disjoint regions of classes in this space. There are now many algorithms that can acquire disjunctive concepts, but the authors hypothesized that existing techniques would break down on domains involving very many peaks.

To test this hypothesis, Rendell and Cho used an automated data generator to produce training and test sets for a variety of domains that had between one and 1000 peaks. Figure 7 shows the results for PLS1, a nonincremental induction algorithm that is similar to ID3, based on training sets with 2000 instances. The predictive accuracy of the induced concept decreases nearly linearly with the log of the domain complexity, even when the training data are free of noise. A similar but more ragged effect occurs when there is 30% noise in the class label, though this curve is lower overall. Also, Quinlan's ID3 algorithm produces a similar degradation as complex-

ity increases. Rendell and Cho suggest methods for representation change as one approach to grouping peaks and thus reducing effective complexity.

Issues of complexity are not limited to classification tasks. One can also vary the regularity of problem spaces, the structure of grammars, and the form of scientific laws. Artificial domains have a role to play in these domains as well, although clear definitions of complexity have not yet been forwarded for these more advanced data structures. Extending complexity measures to non-classification tasks is a prerequisite for understanding such domains, and thus should have priority in future work.

6. Stages of the Experimental Process

Before closing, it seems worth reviewing the basic steps involved in the experimental study of machine learning. The basic procedure differs little from that in other experimental sciences, except for the nature of the independent and dependent variables, which we have discussed in the previous sections. Many of our points will appear obvious to readers, but given the youth of our field, they are worth reiterating.

6.1 Formulating Hypotheses

In many situations, a researcher has clear expectations about the effects he will observe in an experiment. If so, it is important to state these hypotheses explicitly and to use them in focusing his/her experimental design. In many cases, these will be vague and qualitative. For instance, an experimenter will typically believe that an algorithm will lead to improved performance as the result of experience. Similarly, he/she may predict that an induction algorithm with pruning will produce more accurate decision trees in a noisy domain than one without pruning.

Some studies, particularly those involving natural domains, are so exploratory that no clear hypotheses suggest themselves. But many experiments are based on some analogy with previous studies, and in these situations, it seems worth stating predictions formally. In our own experience, predictions are often violated, and having stated them at the outset helps one focus attention on interesting phenomena, even when they are qualitative in nature.

In some cases, one has a clear model of both the algorithm and the learning environment, particularly when working with simple algorithms and artificial domains. If one is willing to make sufficient assumptions about the distribution of training data, one can make detailed predictions about the system's behavior, as Cohen (1991) has encouraged. For example, Pazzani and Sarrett (1990) present an average-case analysis of a conjunctive induction algorithm, which lets them predict detailed learning curves for domains with various characteristics.

In a similar vein, Thompson, Langley, and Iba (1991) describe an analysis that lets them predict the benefit their Labyrinth system receives from background knowledge in comparison to Fisher's Cobweb, which cannot use the same form of knowledge. To accomplish this, they make assumptions about the number of concepts in a

background IS-A hierarchy, the number of components associated with each concepts, and the number of possible types for each component. They also assume a regular structure for the background knowledge and a uniform distribution of instances.[7] From this they calculate the theoretical learning curves presented in Figure 8. Such detailed hypotheses are not required for progress in machine learning, but they have clear advantages over qualitative predictions.

6.2 Designing Experiments and Selecting Samples

Having decided on a set of hypotheses, the researcher must next design one or more experiments to test them. The obvious requirement here is to decide on the dependent and independent variables. Since we have spent many of the preceding pages examining the various options, we will not repeat them here. In most cases, the hypotheses themselves will suggest a small set of variables, and the experimenter need only decide which measures best suit his/her purpose. A complete design must also include decisions about the number of runs to average across, the range of each independent variable, and the step size for each such factor. If the independent variables are qualitative in nature, one must specify the set of values they take on. For example, one must enumerate the algorithms to be tested, the components to be lesioned, or the natural domains from which one will draw instances.

Another issue in experimental design involves sampling strategies. In the natural sciences, one can never control all possible variables. As a result, researchers must collect multiple observations for each cell in their experimental design and average the resulting values. As a science of the artificial, machine learning can avoid some but not all of these complications. One has control over the learning algorithm and the environment, but practical concerns still come into play. In particular, one cannot examine all possible training and test sets in a natural domain, so typically one randomly selects a number of such sets for use in an experiment, then averages over the results. Similarly, one cannot examine all possible training orders for incremental learning methods, so one must resort to a set of randomly selected orders.

Basic experimental method recommends varying the value of one independent term while holding others constant. However, one can apply this process iteratively to obtain *factorial* designs, in which one observes the dependent measures(s) under all combinations of independent values. This lets one move beyond isolated effects and look for *interactions* between independent variables. For instance, one might hypothesize that a decision-tree algorithm will fare better in one environment and that a perceptron method will fare better in another, as argued by Utgoff (1988). Factorial designs let one measure such interactions between independent variables. The results of Rendell and Cho's study, illustrated in Figure 7, revealed no interaction between complexity and noise; rather, their effects on accuracy appeared to be additive.

7. Most theoretical analyses of learning tasks and algorithms have aimed for distribution-independent results. However, this bias differs from those of more mature sciences like physics and chemistry, which are willing to make detailed assumptions to generate precise predictions, then to reconsider those assumptions if predictions are violated.

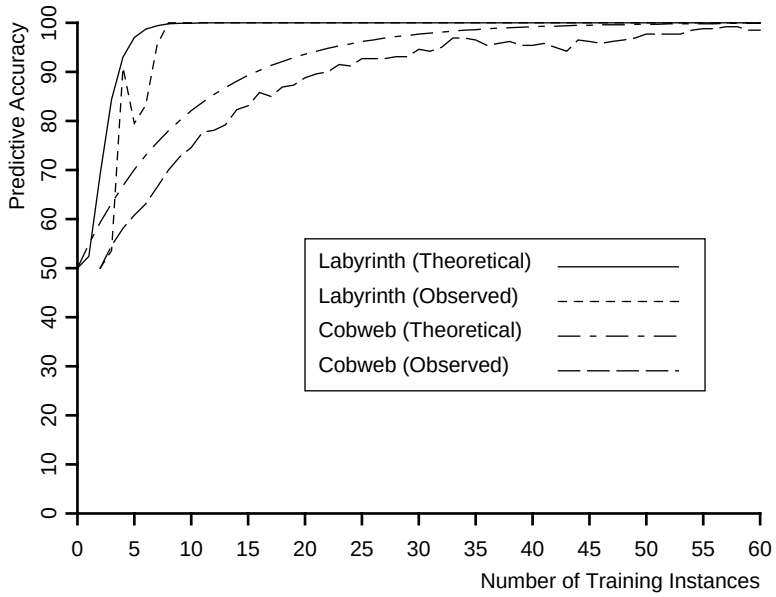

Figure 8. Theoretical and observed learning curves for LABYRINTH and COBWEB in the presence of background knowledge (Thompson, Langley, & Iba, 1991).

6.3 Running Experiments and Compiling Results

Given a clear experimental design, one can carry out the experiment that it specifies. For this one must gather the training instances or problems, implement or access the algorithms, run the algorithms on the training cases, and measure their performance for each sample in each experimental condition (i.e., combination of independent variables). One then averages across all samples in a condition and organizes the results in some readable format such as tables or graphs. This step is probably the least controversial activity in an experimental study.

We have seen many examples of experimental results in this paper, including learning curves, asymptotic accuracies in comparative studies, the effects of noise and other factors on asymptotes and learning rates. Such statistics are the most obvious product of scientific experimentation. Figure 8 presents another example, in this case the results of Thompson et al.'s comparative study of LABYRINTH and COBWEB in the presence of background knowledge.

6.4 Testing Hypotheses

Once the experimenter has collected and organized the data, they can be used to draw tentative conclusions. In an exploratory study, the results may suggest hypotheses that require additional experiments. In other cases, one will have hypotheses and use the observations to test them. Thus, one can examine learning curves to determine whether the acquired knowledge actually improves performance, or one can compare different experimental conditions to see whether the number of irrelevant variables

affect asymptotic accuracy. In some cases, regularities in the data may suggest detailed models that would explain them. For instance, both Quinlan's results on noise (Figure 5) and Rendell and Cho's findings (Figure 7) involved near-linear relations that call out for explanations.

As we saw in Section 4.1, one can use statistical methods to test some hypotheses, and these indicate the confidence with which one can believe apparent differences. This confidence level is affected by three factors – the observed differences between conditions, the number of samples in each condition, and the variances of those samples. Thus, even a large difference may not be robust if the sample is small or the variance is high, making it desirable to use significance tests whenever possible. Such tests make the most sense when comparing nominal conditions, such as alternative algorithms or different natural domains. Other statistical methods, such as correlation analysis, can be used for numeric variables.

Ironically, significance tests are least relevant when one has a detailed model that makes numeric predictions. Consider the theoretical and observed learning curves in Figure 8. The analysis specifies a clear difference in learning rate between the two algorithms, but predicts the same asymptote. These trends are clearly apparent in the experimental curves as well. Here the issue is the degree to which the predictions match the observations. One can use a technique like correlation analysis for this purpose. Alternatively, one can use the standard deviation of each point on the curve to draw 'error bars' around the curve, then see whether the theoretical curve falls within these ranges, as Pazzani and Sarrett (1990) have done. But in general, theory-laden sciences like physics have less need of statistical hypothesis testing than experiment-driven ones like psychology, and we hope that as machine learning matures, it will progress from the latter into the former.

6.5 Explaining Unexpected Results

Hypotheses in machine learning are based on some model of an algorithm and an environment, whether this is explicit or not. Results that agree with an hypothesis lend evidence to that model, though they do not 'confirm' it; science can never draw final conclusions about any situation. Results that diverge from one's expectations count as evidence against a model, and thus require additional explanation.

In some cases, explanations of rejected hypotheses may involve altering assumptions about the environment. Thus, one may posit that the PERCEPTRON algorithm did well on a particular domain because it was linearly separable, even though this was not anticipated at the outset. Other explanations concern the algorithms themselves. For instance, Thompson et al. suggest that LABYRINTH's and COBWEB's behaviors diverge slightly from the theoretical curves in Figure 8 because they cannot retrieve some instances due to poor indexing.

In either case, faulty predictions indicate that one's model needs improvement, often making them more significant than positive results. More important, they can indicate directions in which to make changes. The ensuing altered models, whether

formal or informal, suggest new hypotheses and predictions, which in turn suggest new experiments to test them. In other words, the iterative loop of hypothesize and test is as valid for machine learning as for any other experimental discipline.

6.6 Communicating Experimental Results

Like other sciences, machine learning is largely a communal activity, and this makes clear communication essential. Replication plays an important role in physics, chemistry, biology, and other mature sciences, since it ensures that results are robust and general before they become widely accepted. Such replication would aid our field as well, but it requires detailed enough descriptions to let researchers at other sites repeat the conditions of original studies. Machine learning has made an excellent start in using a standard set of natural domains and in providing pseudocode descriptions of algorithms, which allow reconstruction of learning systems even when the original code is unavailable.

However, replication also requires precise descriptions of the independent and dependent variables, the number of runs, the sampling strategy, and other details of the experimental design. Factors used to generate artificial data, such as one's definition of noise and irrelevant attributes, are also essential. Finally, one should include information about statistical tests used to evaluate hypotheses in communications of experimental results, since these depend on assumptions that others may question. Clear descriptions in a technical report or an archival journal constitute the final stage in an experimental study.

7. Conclusions

One can trace experimental approaches to machine learning back more than two decades (e.g., Hunt, Marin, & Stone, 1966), but the 'modern' era of experimentation began about five years ago. Since then, the number of experimental studies has grown at a rapid pace, with researchers identifying new dependent and independent variables, testing existing systems on new domains, and improving these systems when they encounter difficulties. Many experimental studies produce unexpected results, forcing the experimenter to think deeply about reasons for the observed learning behavior.

In general, the field of machine learning occupies a much healthier methodological state than a decade ago. However, the experimental method has been adapted more quickly to some areas than others. Early experimentation focused on inductive approaches to classification, as the current paper reflects in its examples, but recent years have seen many analogous studies of learning in problem-solving domains and experiments on explanation-based methods. Researchers have also started to measure the influence of background knowledge on inductive learning.

In summary, machine learning occupies a fortunate position that makes systematic experimentation easy and profitable. Some methodological questions remain unanswered, but researchers have made an excellent start and we expect the future holds

improved dependent measures, better independent variables, and more useful experimental designs. There remains room for improvement in all areas of machine learning, but the discipline seems well on its way to developing a sound experimental tradition.

However, these successes do not mean that empirical researchers should report gratuitous experiments any more than theoreticians should publish vacuous proofs. Whether they lead to positive or negative results, experiments are worthwhile only to the extent that they illuminate the nature of learning mechanisms and the reasons for their success or failure. Although experimental studies are not the only path to understanding, we feel they constitute one of machine learning's brightest hopes for rapid scientific progress, and we encourage other researchers to join in our field's evolution toward an experimental science.

Acknowledgements

We would like to thank David Aha, Wayne Iba, David Ruby, Jeff Schlimmer, Kevin Thompson, and Tom Dietterich for helpful comments on previous drafts. Earlier versions of this paper appeared in the journal *Machine Learning* and in the *Proceedings of the Third European Working Session on Learning*.

References

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37-66.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.

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

Cohen, P. R. (1991). A survey of the Eighth National Conference on Artificial Intelligence: Pulling together or pulling apart? *AI Magazine*, *12*, 16–41.

Cohen, P. R., & Howe, A. E. (1988). *The invisible hand: How evaluation guides AI research* (COINS Technical Report 88-21). Amherst: University of Massachusetts, Department of Computer and Information Science.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, *1*, 145–176.

Dietterich, T. G. (1990). Machine learning. *Annual Review of Computer Science*, *4*.

Elio, R., & Watanabe, L. (1991). An incremental deductive strategy for controlling constructive induction in learning from examples. *Machine Learning*, *7*, 7–44.

Etzioni, O. (1990). Why PRODIGY/EBL works. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 916–922). Boston, MA: AAAI Press.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Gennari, J. H. (1990). *An experimental study of concept formation*. Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

Gratch, J. (1991). *Utility generalization and composability problems in explanation-based learning* (Tech. Rep. No. UIUUCDCS-R-91-1681). Urbana: University of Illinois, Department of Computer Science.

Haussler, D. (1987). Learning conjunctive concepts in structural domains. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 466–470). Seattle, WA: AAAI Press.

Haussler, D. (1990). Probably approximately correct learning. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 1101–1108). Boston, MA: AAAI Press.

Hawking, S. (1988). *A brief history of time*. New York: Bantam Books.

Hunt, E. B., Marin, J., & Stone, P. J. (1966). *Experiments in induction*. New York: Academic Press.

Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, *3*, 285–317.

Iba, W., Wogulis, J., & Langley, P. (1988). Trading off simplicity and coverage in incremental concept learning. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 73–79). Ann Arbor, MI: Morgan Kaufmann.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987). Recent results on Boolean concept learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 337–352). Irvine, CA: Morgan Kaufmann.

Langley, P. (1982). Language acquisition through error recovery. *Cognition and Brain Theory*, *5*, 211–255.

Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.

Langley, P., & Drummond, M. (1990). Toward an experimental science of planning. *Proceedings of the 1990* DARPA *Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 109–114). San Diego, CA: Morgan Kaufmann.

McKusick, K. B., & Langley, P. (1991). Constraints on tree structure in concept formation. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*. Sydney: Morgan Kaufmann.

Michalski, R. S., & Chilausky, R. L. (1980). Knowledge acquisition by encoding expert rules versus computer induction from examples: A case study involving soybean pathology. *International Journal of Man-Machine Studies*, *12*, 63–87.

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. Cambridge, MA: MIT Press.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based learning: A unifying view. *Machine Learning*, *1*, 47–80.

Minton, S. N. (1985). Selectively generalizing plans for problem solving. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 596–599). Los Angeles: Morgan Kaufmann.

Minton, S. N. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*, 363–391.

Nilsson, N. J. (1965). *Learning machines*. New York: McGraw-Hill.

Pazzani, M. J., & Sarrett, W. (1990). Average case analysis of conjunctive learning algorithms. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 339–347). Austin, TX: Morgan Kaufmann.

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

Reinke, R. (1984). *Knowledge acquisition and refinement tools for the* ADVISE *meta-expert system*. Master's thesis, Department of Computer Science, University of Illinois, Urbana.

Rendell, L., & Cho, H. (1990). Empirical learning as a function of concept character. *Machine Learning*, *5*, 267–298.

Robertson, G. G., & Riolo, R. L. (1988). A tale of two classifier systems. *Machine Learning*, *3*, 139–159.

Rosenblatt, F. (1962). *Principles of neurodynamics*. New York: Spartan Books.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In J. L. McClelland & D. E. Rumelhart (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge, MA: MIT Press.

Rosenbloom, P., & Newell, A. (1987). Learning by chunking: A production system model of practice. In D. Klahr., P. Langley, & R. Neches (Eds.) *Production system models of learning and development*. Cambridge, MA: MIT Press.

Schlimmer, J. C. (1987). *Concept acquisition through representational adjustment*. Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

Segre, A., Elkan, C., & Russell, A. (1991). A critical look at experimental evaluations of EBL. *Machine Learning*, *6*, 183–195.

Shavlik, J. W., Mooney, R. J., & Towell, G. G. (1991). Symbolic and neural learning: An experimental comparison. *Machine Learning*, *6*, 111-143.

Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.

Tambe, M., Newell, A., & Rosenbloom, P. S. (1990). The problem of expensive chunks and its solution by restricting expressiveness. *Machine Learning*, *5*, 299–348.

Thompson, K., Langley, P., & Iba, W. F. (1991). Using background knowledge in concept formation. *Proceedings of the Eighth International Workshop on Machine Learning*. Evanston, IL: Morgan Kaufmann.

Utgoff, P. E. (1988). Perceptron trees: A case study in hybrid concept representations. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 601–606). St. Paul, MN: AAAI Press.