Learning Analytically and Inductively

Tom M. Mitchell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
E-mail: Tom.Mitchell@cmu.edu

Sebastian B. Thrun
University of Bonn
Institut für Informatik III
Römerstr. 164, 53117 Bonn, Germany
E-mail: thrun@uran.cs.bonn.edu

1 Learning

Learning is a fundamental component of intelligence, and a key consideration in designing cognitive architectures such as Soar [Laird *et al.*, 1986]. This chapter considers the question of what constitutes an appropriate general-purpose learning mechanism. We are interested in mechanisms that might explain and reproduce the rich variety of learning capabilities of humans, ranging from learning perceptual-motor skills such as how to ride a bicycle, to learning highly cognitive tasks such as how to play chess.

Research on learning in fields such as cognitive science, artificial intelligence, neurobiology, and statistics has led to the identification of two distinct classes of learning methods: inductive and analytic. Inductive methods, such as neural network Backpropagation, learn general laws by finding statistical correlations and regularities among a large set of training examples. In contrast, analytical methods, such as Explanation-Based Learning, acquire general laws from many fewer training examples. They rely instead on prior knowledge to analyze individual training examples in detail, then use this analysis to distinguish relevant example features from the irrelevant.

The question considered in this chapter is how to best combine inductive and analytical learning in an architecture that seeks to cover the range of learning exhibited by intelligent systems such as humans. We present a specific learning mechanism, Explanation Based Neural Network learning (EBNN), that blends these two types of learning, and present experimental results demonstrating its ability to learn control strategies for a mobile robot using vision, sonar, and laser range sensors. We then consider the analytical learning mechanism in Soar, called chunking, and recent attempts to complement chunking by including inductive mechanisms in Soar. Finally, we suggest a way in which EBNN could be introduced as a replacement for chunking in Soar, thereby incorporating inductive and analytical learning as architectural capabilities.

The following section provides an overview of inductive and analytic principles for learning, and argues that both are necessary for a general learning mechanism that scales up to handle a broad range of tasks. The subsequent section presents the EBNN learning mechanism, together with experimental results illustrating its capabilities. Finally, we consider the general learning mechanism for Soar, and the question of how to best incorporate both inductive and analytic learning within this architecture.

2 Why Combine Analysis and Induction?

At the heart of the learning problem is the question of how to successfully generalize from examples. For instance, when people learn to ride a bicycle or learn to play chess, they learn from specific experiences (e.g., riding a specific bicycle on a particular day in a particular place). Somehow they are able to generalize away from the myriad details of specific situations, to learn general strategies that they expect to apply in "similar" subsequent situations. In doing this, they must differentiate between the many irrelevant details of the situation (e.g., the angle of the sun on that particular day, the color of the bicyclist's shirt), and the few essential features (e.g., the velocity and tilt angle of the bicycle). This section provides brief overviews of analytical and inductive methods for generalizing from examples, then considers the complementary benefits of these two learning paradigms.

2.1 Analytical Learning

Analytical learning uses the learner's prior knowledge to analyze individual training examples, in order to discriminate the relevant features from the irrelevant. The most common analytical learning method is explanation-based learning [DeJong and Mooney, 1986], [Mitchell

Figure 1: A positive example of the concept "chess positions in which black can be forced to lose its queen within 2 moves." Note the white knight is attacking both the black king and queen. Black must therefore move its king, enabling white to capture the black queen.

---

et al., 1986]. To illustrate, consider the problem of learning to play chess. More specifically, consider the subtask of learning to recognize chess positions in which one's queen will be lost within the next few moves. A positive example of this class of chess positions is shown in Figure 1.

As explained in the figure caption, this is a positive example of a chess position in which black can be forced to lose its queen. To learn a general rule from this example, it is necessary to determine which board features are relevant to losing the queen, and which are irrelevant details to be ignored. Inductive learning methods generalize by collecting multiple training examples, then determining which features are common to the positive examples, but not to the negative examples. In the chess example above, there are many features that happen to be true (e.g., the feature “4 white pawns are still in their original positions”), but only a few that are relevant (e.g., “the black king and queen are simultaneously under attack”). In this chess example, inductive techniques will require hundreds or thousands of training example chess boards to statistically determine which features are relevant, and to generalize appropriately.

In analytical learning such as explanation-based learning, a justifiable generalization of the chess position can be derived from just this one training example. This is accomplished by the learner explaining how the training example satisfies the target concept, then identifying the features mentioned in the explanation as relevant. Given prior knowledge of the legal moves of chess, it is possible for the learner to construct an explanation of why the black queen is lost in the particular training example of Figure 1. The explanation is given in the figure caption. Notice this explanation mentions the feature that “the white knight is attacking both the black king and queen.” This feature is thus determined to be relevant in general, and forms a condition under which black will in general lose its queen. In contrast, other irrelevant features that would be considered in inductive learning (e.g., “4 white pawns are still in their original positions”) are not even considered, because they play no part in the explanation. Explanation-based learning generalizes through explaining and analyzing training instances in terms of prior knowledge. While our purpose here is to examine computer learning algorithms, it is interesting to note that research on human learning provides support for the conjecture that humans learn through such explanations (see, for example, [Chi and Bassok, 1989], [Qin *et al.*, 1992], [Ahn and Brewer, 1993]).

Analytical learning methods have been used successfully in a number of applications – notably for learning rules to control search. For example, Prodigy [Minton *et al.*, 1989] is a domain-independent framework for means-ends planning that uses explanation-based learning to acquire search control knowledge. Prodigy learns general rules that characterize concepts such as "situations in which pursuing subgoal ?x will lead to backtracking." Given a specific problem solving domain defined by a set of states, operators, and goals, Prodigy learns control rules that significantly reduce backtracking when solving problems in this domain. It has been demonstrated to learn search control rules comparable to hand-coded rules in a variety of task domains [Minton *et al.*, 1989].

The chunking mechanism in Soar [Laird *et al.*, 1986] also provides an example of analytical learning, as explained in [Rosenbloom and Laird, 1986]. In Soar, problem solving corresponds to search in problem spaces (a problem space is defined by problem states and operators). Whenever Soar has no control knowledge to choose its next search step, it reflects on the current situation by using a higher-level problem space, resulting in a choice of an appropriate search step. The trace of reasoning performed in this higher level space forms an explanation of why the selected search step is appropriate in this specific instance. The

Soar architecture automatically records this explanation whenever it reflects in this fashion, and the chunking mechanism forms a new control rule (called a production) by collecting the features mentioned in the explanation into the preconditions of the new rule. Soar's analytical chunking mechanism has been shown to learn successfully to speed up problem solving across a broad range of domains. For example, [Doorenbos, 1993] presents results in which over 100,000 productions are learned from such explanations within one particular domain.

2.2 Inductive Learning

Whereas analytical learning can produce appropriate generalizations by analyzing single training examples, it requires strong prior knowledge about its domain in order to construct appropriate explanations. Its determination of which features are relevant will only be as correct and complete as the prior knowledge from which explanations are formed. In many domains, such complete and correct prior knowledge is unavailable. For example, consider learning the concept "stocks that double in value over the next year." In this case, an explanation of a training example would require explaining which features were responsible for the increase in value. Unlike the chess domain, in which the effects of each possible move are known perfectly in advance, the stock market domain cannot be modeled so correctly and completely. In such cases, inductive techniques that identify empirical regularities over many examples are useful.

To illustrate inductive learning, consider the task of learning to drive a motor vehicle. [Pomerleau, 1989] describes the computer system ALVINN that learns to steer a vehicle driving at 55mph on public highways, based on input sensor data from a video camera. Notice this task involves learning control knowledge, much like the control knowledge learned by Prodigy and Soar. In this domain, however, a complete and correct model of the effects of different steering actions is not known a priori. Therefore an inductive learning method, neural network Backpropagation [Rumelhart *et al.*, 1986], is used to learn a mapping from the camera image to the appropriate steering direction. Thousands of training examples are collected by recording camera images and steering commands while a human drives the vehicle for approximately 10 minutes. These human-provided training examples of visual scenes and corresponding steering commands are generalized by the neural network to acquire a general mapping from scenes to steering commands. The resulting network has been demonstrated to drive unassisted for intervals up to 100 miles at speeds of 55 mph on public highways. The neural network learning technique is a method for fitting hundreds of numeric parameters in a predefined, but highly expressive non-linear functional form. These parameters are fit by performing a gradient descent search to minimize the error (i.e., difference) between the network output and the desired output for the training example set.

2.3 Why Combine Analytical and Inductive Learning?

Analytical and inductive learning offer complementary approaches to identifying correct hypotheses during learning. In domains where strong prior knowledge is available, such as chess, analytical methods such as explanation-based learning can generalize correctly from single examples. In domains where such strong prior knowledge is not available, inductive methods such as neural network Backpropagation offer a means of identifying empirical regularities over large sets of training data. In the chess example above, analytical learning offers a means of generalizing correctly by analyzing single examples, whereas inductive learning would probably require thousands of examples to find appropriate regularities given the large number of possible features. In the driving task, however, it is difficult to imagine how to program in sufficient prior knowledge to allow explaining why a particular steering direction is appropriate, in terms of the individual pixels of the camera image. In this case, inductive learning can sort through the large number of potential features by finding regularities among thousands of training examples.

Methods for combining inductive and analytical learning have been the subject of considerable recent research. For example, [Shavlik and Towell, 1989] describes a method called KBANN for using prior symbolic knowledge to initialize the structure and weights of a neural network, which is then inductively refined using the Backpropagation method. A similar method has been reported by [Fu, 1989]. [Pazzani *et al.*, 1991] describes a combined inductive/analytical method called FOCL for learning sets of horn clauses, demonstrating its ability to operate robustly given errors in the initial domain knowledge. [Ourston and Mooney, 1994] describes a method called EITHER for refining domain theories in the light of additional empirical data. While research in this area is very active, the question of how to best blend inductive and analytical learning is still open. Desired properties for a combined mechanism include the following:

- Given a perfect domain theory, learn as effectively as explanation-based learning
- Given an incorrect theory, or no theory, learn as effectively as the best inductive methods
- Operate robustly over entire spectrum of domain theory quality
- Accommodate noisy data
- Support sufficiently rich representations for hypotheses
- Refine the domain theory with experience, at the same time as using it to learn the target function

3 A Combined Inductive-Analytical Learning Mechanism

The Explanation-Based Neural Network (EBNN) learning mechanism integrates inductive and analytic learning. As the name suggests, EBNN is based on an artificial neural network representation of knowledge. Neural networks are used to draw inferences about the domain, just as rules are used to draw inferences in symbolic representations. By using a neural network representation, purely inductive learning algorithms such as the Backpropagation algorithm [Rumelhart *et al.*, 1986] become applicable. In addition, EBNN includes an analytic learning component, based on explaining and analyzing training instances in terms of other, previously learned networks.

In what follows we describe the EBNN learning mechanism. We also present some results obtained in the domain of mobile robot navigation. Based on these we discuss the role of inductive and analytical learning in EBNN.

3.1 Introduction to EBNN

To understand the EBNN learning mechanism, consider the example given in Fig. 2, adapted from [Winston *et al.*, 1983], [Towell and Shavlik, 1989]. Suppose we are facing the problem of learning to classify cups. More specifically, imagine we want to train a network, denoted by f, which can determine whether an object is a cup based on the features *is_light*, *has_handle*, *made_of_Styrofoam*, *color*, *upward_concave*, *open_vessel*, *flat_bottom*, and *is_expensive*. One way to learn the new concept is to collect training instances of cups and non-cups, and employ the Backpropagation procedure to iteratively refine the weights of the target network. Such a learning scheme is purely inductive. It allows learning functions from scratch, in the absence of any domain knowledge.

Now let assume one has already learned a *neural network domain theory*, which represents each individual inference step in the logical derivation of the target concept. In our example, such a domain theory may consist of three networks, the network f_1 which predicts whether an object is liftable, the network f_2 which determines if an object can hold a liquid, and

(a) Training examples

is light	*has handle*	*made of Styrofoam*	*upward concave*	*color*	*open vessel*	*flat bottom*	*is expensive*	*is_cup?*
yes	yes	no	no	blue	yes	yes	yes	yes
no	no	yes	yes	red	no	yes	no	no
yes	no	yes	yes	red	yes	no	no	no
no	no	yes	yes	green	yes	yes	no	yes
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

(b) Target concept

is_cup?	:-	*is_liftable, holds_liquid*
is_liftable	:-	*is_light, has_handle*
is_liftable	:-	*made_of_Styrofoam, upward_concave*
holds_liquid	:-	*open_vessel, flat_bottom*

Figure 2: The cup example.

---

a third network f_3 that predicts if an object is a cup as a function of the two intermediate concepts *is_liftable* and *holds_liquid* (*cf.* Fig. 3). This set of networks forms a complete domain theory for the classification of a cup, as it allows classifying any object as a cup or not. However, it is not necessarily correct, as the domain theory networks themselves may have been constructed from examples.

How can this neural network domain theory be employed to refine the target network f? EBNN learns analytically by explaining and analyzing each training example using the following three step procedure:

1. **Explain.** The domain theory is used to *explain* training instances by chaining together multiple neural network inferences. In our example, the domain theory network f_1 is used to predict the degree to which the object is liftable, network f_2 is employed to predict whether the object can hold a liquid, and finally network f_3 uses these two results to infer whether the object is a cup. This collection of neural network inferences, which we will refer to as the *explanation*, explains why a training instance is a member of its class in terms of the domain theory. The explanation sets up the inference structure necessary for analyzing and generalizing this training instance.

2. **Analyze.** The above explanation is analyzed in order to generalize the training instance in feature space. Unlike symbolic approaches to EBL, which extract the *weakest precondition* of the explanation, EBNN extracts *slopes* of the target function. More specifically, EBNN extracts the output-input slopes of the target concept by computing the first derivative of the neural network explanation. These slopes measure, according to the domain theory, how infinitesimal changes in the instance feature values will change the output of the target function.

Figure 3: A neural network domain theory for the classification of a cup.

In the cup example, EBNN extracts slopes from the explanation composed of the three domain theory networks f_1, f_2, and f_3. The output-input derivative of f_1 predicts, how infinitesimal changes in the input space of f_1 will change the degree as to which f_1 predicts an object to be liftable. Likewise, the derivatives of f_2 and f_3 predict the effect of small changes in their input spaces on their vote. Chaining these derivatives together results in slopes which measure how infinitesimally small changes of the individual instance features will change the final prediction of cupness.

Slopes guide the generalization of training instances in feature space. For example, irrelevant features, whose values play no role in determining whether the object is a cup (*e.g.*, the features *color* and *is_expensive*) will have approximately zero slopes. On the other hand, large slopes indicate important features, since small changes in the feature value will have a large effect on the target concept according to the neural network domain theory. Notice that the extraction of slopes relies on the fact that artificial neural networks are differentiable, real-valued functions.

3. **Refine.** Finally, EBNN refines the weights and biases of the target network both inductively and analytically. Fig. 4 summarizes the information obtained by the inductive and the analytical component. Inductive learning is based on the target *value* for each individual training instance, whereas analytical learning is based on the target *slopes* extracted from the explanation. When updating the weights of the network, EBNN minimizes a combined error function which takes both value error and slope error into account.

$$E = E_{\text{values}} + \alpha E_{\text{slopes}} \tag{1}$$

Here α is a parameter that trades off value fit versus slope fit, and that therefore determines the relative impact of inductive versus analytical components of learning. Gradient descent is employed to iteratively minimize E. Notice that in our implementation we used a modified version of the Tangent-Prop algorithm [Simard *et al.*, 1992] to refine the weights and biases of the target network [Masuoka, 1993].

What is a reasonable method for weighting the contributions of the inductive versus analytical components of learning; that is, for selecting a value for α? Because the domain theory might be incorrect, the analytically extracted slopes for a particular training example might be misleading. In such cases, one would like to suppress

Figure 4: **Fitting values and slopes in EBNN:** Let f be the target function for which three examples $\langle x_1, f(x_1)\rangle$, $\langle x_2, f(x_2)\rangle$, and $\langle x_3, f(x_3)\rangle$ are known. Based on these points the learner might generate the hypothesis g. If the slopes are also known, the learner can do much better: h.

---

the analytical component (α=0), relying only on the inductive component. For other training examples the domain theory and explanation might be correct, and α=1 might be more useful. EBNN dynamically adjusts the value of α for each individual training example, based on the observed accuracy of the explanation: The more accurately the domain theory predicts the known target value of the training instance, the higher the value of α for this training example.

The accuracy, denoted by δ, is measured as the root-mean square difference between the true target value and the prediction by the domain theory. When refining the weights and biases of the target network, δ determines the weight of the target slopes α according to the following formula.

$$\alpha = 1 - \frac{\delta}{\delta_{\max}} \qquad (2)$$

Here $\delta_{\max}$ denotes the maximum prediction error, which is used for normalization.

> This weighting scheme attempts to give accurate slopes a large weight in training, while ignoring inaccurate slopes. This heuristic weighting scheme, called LOB* [Mitchell and Thrun, 1993], is based on the heuristic assumption that the accuracy of the explanation's *slopes* are correlated to the accuracy of the explanation's *predictions*.

This completes the description of the EBNN learning mechanism. To summarize, EBNN refines the target network using a combined inductive-analytical mechanism. Inductive training information is obtained through observation, and analytical training information is obtained by explaining and analyzing these observations in terms of the learner's prior knowledge. Inductive and analytical learning are dynamically weighted by a scheme that estimates the accuracy of the domain theory for explaining individual examples.

Both components of learning, inductive and analytical, play an important role in EBNN. Once a reasonable domain theory is available, EBNN benefits from its analytical component, since it gradually replaces the pure syntactical bias in neural networks by a bias that captures domain-specific knowledge. The result is improved generalization from less training data. Of course, analytic learning still requires the availability of a domain theory that allows explaining the target function. If such a domain theory is weak or not available, EBNN degrades to a purely inductive neural network learner. It is the inductive component of EBNN which ensures that learning is possible even in the total absence of domain knowledge.

3.2 Experimental Results

EBNN has been applied to various domains, including classification, prediction, game playing, perception and robot control. Here we report an application of EBNN to learning mobile robot navigation using Q-Learning [Watkins, 1989].

Xavier [O'Sullivan, 1994], the robot at hand, is shown in Fig. 5. Xavier is equipped with a ring of 24 sonar sensors, a laser range finder, and a color camera mounted on a pan/tilt head. Sonar sensors return approximate echo distances along with noise. The laser range finder measures distances more accurately, but its perceptual field is limited to a small range in front of the robot. The task of the robot was to learn to navigate to a specifically marked target location in a laboratory environment. In some experiments, the location of the marker was fixed throughout the course of learning, in others it was moved across the laboratory and only kept fixed for the duration of a single training episode. Sometimes parts of the environment were blocked by obstacles. The marker was detected using a visual object tracking routine that recognized and tracked the marker in real-time using the pan/tilt head. The robot was rewarded if it managed to stop in front of the marker. It was penalized for losing the sight of the marker.

Sensor input was encoded by a 46-dimensional vector, consisting of 24 sonar distance measurements, 10 distance measurements by the laser range finder, and an array of 12 values coding the angle of the marker position relative to the robot (*cf.* Fig. 6). Every 3 seconds the robot could chose one of 7 possible actions, ranging from sharp turns to straight motion. In

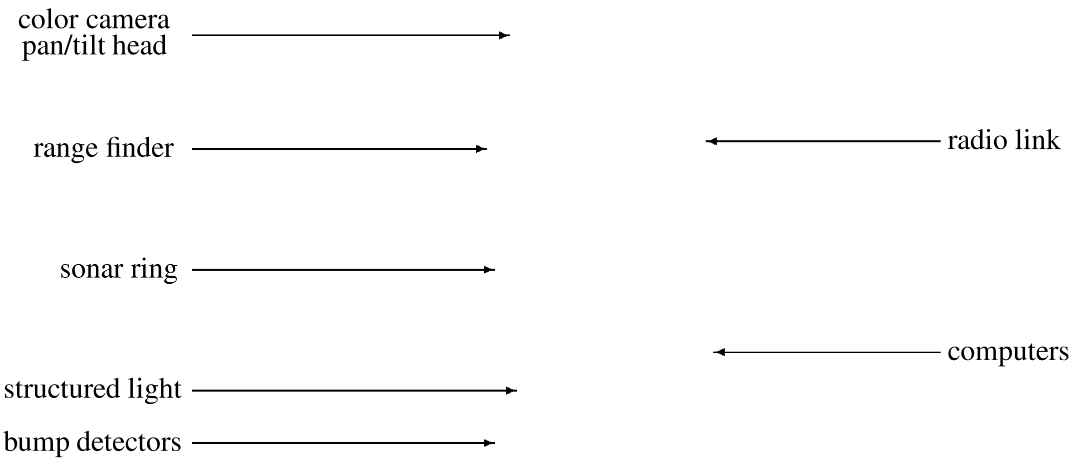

Figure 5: The Xavier robot.

order to avoid collisions, the robot employed a pre-coded obstacle avoidance routine based on potential field navigation [Khatib, 1986]. Whenever the projected path of the robot was blocked by an obstacle, the robot decelerated and, if necessary, changed its motion direction (regardless of the commanded action). Xavier was operated continuously in real-time. Each learning episode corresponds to a sequence of actions which starts at a random initial position and ends either when the robot loses sight of the target object, for which it is penalized, or when it halts in front of the marker, in which case it is rewarded.

In order to learn to generate actions from delayed reward, we applied EBNN in the context of $\mathcal{Q}$-Learning [Watkins, 1989]. In essence, $\mathcal{Q}$-Learning constructs utility functions $\mathcal{Q}(s, a)$ that map sensations s and actions a to task-specific utility values. $\mathcal{Q}$-values will be positive for final successes and negative for final failures. In between, utilities are calculated recursively using an asynchronous dynamic programming technique [Barto *et al.*, 1991]. More specifically, the utility $\mathcal{Q}(s_t, a_t)$ at time t is estimated through a mixture of the utilities of subsequent observation-action pairs, up to the final utility at the end of the episode. The exact update equation used in the experiments, combined with methods of temporal differencing [Sutton, 1988], is:

$$\mathcal{Q}^{\text{target}}(s_t, a_t) \longleftarrow \begin{cases} +100 & \text{if } a_t \text{ final action, robot reached goal} \\ -100 & \text{if } a_t \text{ final action, robot lost sight of the marker} \\ \gamma \cdot \left[(1-\lambda) \cdot \max\limits_{a \text{ action}} \mathcal{Q}(s_{t+1}, a) + \lambda \cdot \mathcal{Q}^{\text{target}}(s_{t+1}, a_{t+1}) \right] & \text{otherwise} \end{cases} \tag{3}$$

Here γ ($0 \leq \gamma \leq 1$) is a discount factor that, if $\gamma < 1$, favors rewards reaped earlier in time. λ ($0 \leq \lambda \leq 1$) is a gain parameter trading off the recursive component and the non-recursive component in the update equation. Once the function $\mathcal{Q}$ has been learned, steepest descent in the utility space results in optimal paths to the goal. Hence, learning control amounts to learning appropriate $\mathcal{Q}$-functions. In our implementation, the $\mathcal{Q}$ function was represented by a collection of artificial neural networks which mapped sensations s to utility values, one for each action a.

Recently, $\mathcal{Q}$-Learning and other related dynamic programming algorithms have been applied successfully to game playing domains [Tesauro, 1992] and robotics [Gullapalli, 1992]. In these previous approaches, the update of the target networks was purely inductive. EBNN, applied to Q-Learning, extends inductive learning by an analytical component. Just as for the classification example given in the previous section, EBNN requires the availability of an appropriate neural network domain theory. Here the learner is given a collection of *predictive action models* f_a, one for each action a, that allow predicting the sensation and reward at time $t+1$ from the sensations of time t. Such a neural network domain theory allows explaining (post-facto predicting) sensations and final outcomes of each individual learning episode. In Xavier's case, each action model maps 46 input values to 47 output values. The models were learned beforehand using the Backpropagation training procedure, employing a cross-validation scheme to prevent overfitting the data. Initially, we used a training corpus of approximately 800 randomly generated actions, which was gradually increased through the course of this research to 3,000 actions, taken from some 700 episodes. These training examples were distributed roughly equally among the 7 action model networks.

Xavier's predictive action models face a highly stochastic situation. There are many unknown factors which influence the actual sensations. First, sensors are generally noisy, *i.e.*, there is a certain likelihood that the sensor returns corrupted values. Second, the obstacle avoidance routine is very sensitive to small and subtle details in the real world. For example, if the robot faces an obstacle, it is often very hard to predict whether its obstacle avoidance behavior will make it turn left or right. Third, the delay in communication, imposed by the radio Ethernet link, turned out to be rather unpredictable. These delays, which influenced the duration of actions, were anywhere in the range of 0.1 up to 3 seconds. For all those reasons, the domain theory functions f_a captured only *typical* aspects of the world by modeling the average outcome of actions, but were clearly unable to predict accurately. Empirically we found, however, that they were well-suited for extracting useful slopes. Fig. 6 gives an example of a slope array extracted from a domain theory network applied to a typical world state.

Having trained the domain theory networks, we finally attacked the primary goal of learning, namely learning Q. The explanation and analysis of episodes in Q-Learning is analogous to the analysis of training instances in the cup example. EBNN explains episodes by chaining together predictive action models. For each state-action pair in the episode, EBNN extracts target slopes of the utility function via the first derivative of the explanation of the episode:

$$\nabla_{s_t} Q^{\text{target}}(s_t, a_t) \quad \longleftarrow \quad \begin{cases} \left.\frac{\partial f_{a_t}}{\partial s}\right|_{s_t} & \text{if } a_t \text{ final action} \\ \gamma \cdot \left[(1-\lambda) \cdot \left.\frac{\partial Q_a}{\partial s}\right|_{s_{t+1}} + \lambda \cdot \nabla_{s_{t+1}} Q^{\text{target}}(s_{t+1}, a_{t+1}) \right] \cdot \left.\frac{\partial f_{a_t}}{\partial s}\right|_{s_t} & \text{otherwise} \end{cases} \tag{4}$$

We performed five complete learning experiments, each consisting of 30-40 episodes. When training the Q networks, we explicitly memorized all training data, and used a recursive replay technique similar to "experience replay" described in [Lin, 1992]. The update parameter γ was set to 0.9, and λ was set to 0.7.

In all cases Xavier learned to navigate to a static target location in less than 19 episodes, each of which was between 2 and 11 actions in length. It consistently learned to navigate to

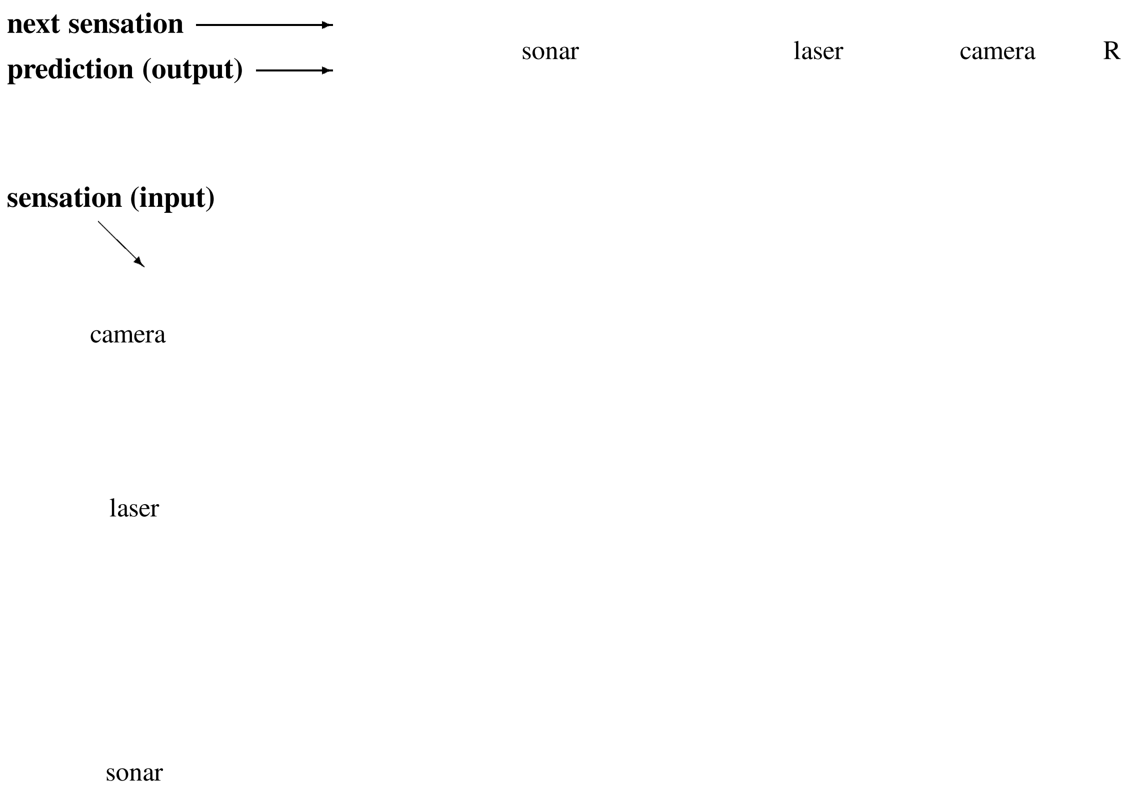

Figure 6: **Prediction and slopes.** A neural network action model predicts sensations and reward for the next time step. The large matrix displays the output-input slopes of the network. White boxes refer to positive and black boxes to negative slope values. Box sizes indicate absolute magnitudes. Notice the positive slopes along the main diagonal.

arbitrary target locations (which was required in 4 out of 5 experiments) always in less than 25 episodes. Although the robot faced a high-dimensional sensation space, it always managed to learn the task in less than 25 episodes, which is less than 10 minutes of robot operation,

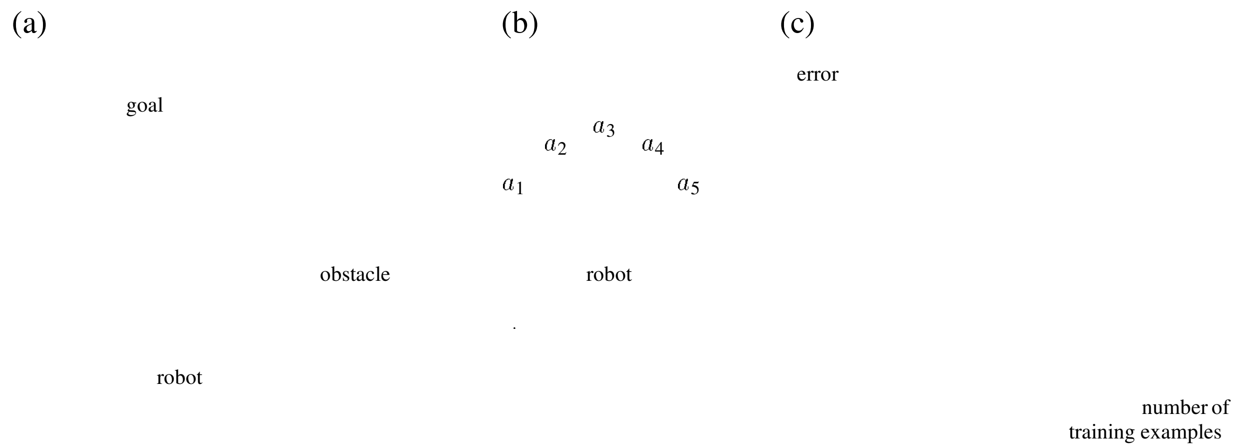

Figure 7: a. The simulated robot world. b. Actions. c. The squared generalization error of the domain theory networks decreases monotonically as the amount of training data increases. These nine alternative domain theories were used in the experiments.

---

and, on average, less than 20 training examples per Q-network. This training time does not include the time for collecting the training data to learn the domain theory action models. In general, we assume the cost of learning such domain-specific but task-independent action models can be amortized over many control learning tasks faced by the robot within its environment. Hence, only a small fraction of this cost should be considered as relevant to this task.

When testing the robot, we also confronted it with situations which were not part of its training experience. In one case, we kept the location of the marker fixed and moved it only in the testing phase. In a second experiment, we blocked the robot's path by large obstacles, even though it had not experienced obstacles during training. It was here that the presence of an appropriate domain theory was most important. While without domain theory the robot almost always failed to approach the marker under these new conditions, it reliably ($> 90\%$) managed this task when it was trained with the help of the domain theory networks. This is because the domain theory provides a knowledgeable bias for generalization to unseen situations.

In order to investigate and characterize EBNN more thoroughly, we applied EBNN to a simulated robot navigation task, which is depicted in Fig. 7a. The robot, indicated by the small circle, has to navigate to the fixed goal location (large circle) while avoiding a collision with the obstacle and the surrounding walls. Its sensors measure the distances and the angles to both the center of the obstacle and the center of the goal, relative to the robot's view. At any time, five different actions are available to the robot, depicted in Fig. 7b. Note that this learning task is completely deterministic.

The learning setup was analogous to the robot experiments described above. Before learning an evaluation function, we trained 5 neural network action models, one for each individual action. Subsequently, 5 Q-functions were trained to evaluate the utility of each individual action.[1] In a first experiment, we were interested in the merit of the analytic learning component of EBNN. We trained the model networks using a large training corpus of 8,192 training instances each. EBNN was then applied to the task of learning control. Fig. 8 displays the performance as a function of the number of training episodes. As can easily be seen,

[1] In this implementation we used a real-valued approximation scheme using nearest-neighbor generalization for the Q-functions. This scheme was empirically found to outperform Backpropagation.

standard inductive learning learns slower than the combined inductive and analytic learning in EBNN. This is because the analytic component of EBNN provides a knowledgeable, domain-dependent bias, which partially replaces the purely syntactical, inductive bias. Of course, asymptotically, with an unlimited amount of training data, both approaches might be expected to exhibit equivalent performance.

Clearly, EBNN outperformed pure inductive learning because it was given a domain theory trained with a large number of training examples. In order to test the impact of weaker domain theories on EBNN, we conducted a series of experiments in which we trained different sets of domain theory networks using 5, 10, 20, 35, 50, 75, 100, and 150 training examples per action network. As shown in Fig. 7c, the number of training examples determines the accuracy of the resulting domain theory. Fig. 9 displays the performance graphs resulting from using these different domain theories. As can be seen, EBNN degrades gracefully to the performance of a pure inductive learner as the accuracy of the domain theory decreases.

The graceful degradation of EBNN with decreasing accuracy of the domain theory is due to the fact that misleading slopes are identified and their influence weakened (*cf.* Eq. (2)). In other experiments reported elsewhere [Thrun and Mitchell, 1993] it was demonstrated that EBNN will fail to learn control if the domain theory is poor and α is kept fixed. These results also indicate that in cases where the domain theory is poor a pure analytical learner would be hopelessly lost. In the experiments reported here EBNN recovered from poor domain theories because of its inductive component, which enabled it to overturn misleading slopes

Figure 8: **EBNN:** Performance curves for EBNN with (black) and without (gray) analytical training information (slope fitting) for three examples each, measured on an independent set of problem instances. The dashed lines indicate average performance. In this experiment, the agent used well-trained predictive action models as its domain theory.

---

extracted from inaccurate prior knowledge.

3.3 Why are Both Induction and Analysis Needed?

The inductive component of learning in EBNN is the standard Backpropagation algorithm for updating network weights to fit observed training example values. The analytical component of learning in EBNN is the use of prior knowledge to explain observed training examples, and the use of slopes extracted from these explanations to further update the weights of the target network. EBNN blends these two mechanisms, weighting the contribution of analytical

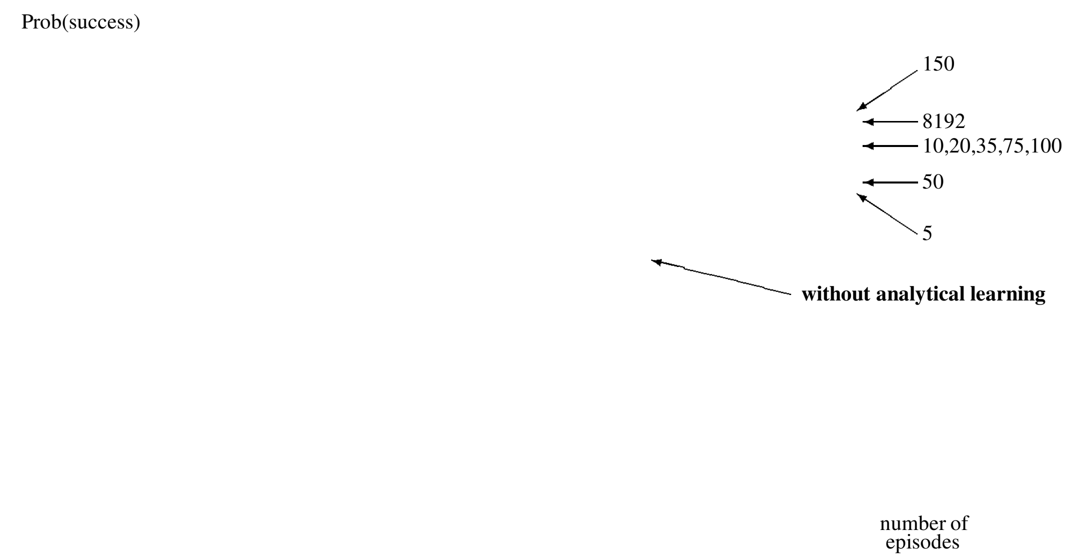

Figure 9: How does domain knowledge improve generalization accuracy? Averaged results for EBNN domain theories of differing accuracies, pre-trained with 5 to 8,192 training examples. In contrast, the bold gray line reflects the learning curve for pure inductive Q-Learning. All curves are averaged over 3 runs and are also locally window-averaged. The performance (vertical axis) is measured on an independent test set of starting positions.

---

learning on an example by example basis, depending on the accuracy of the domain theory in explaining the particular observed training example.

The importance of analytical learning lies in its ability to generalize more correctly from fewer examples. As demonstrated by the experimental results from EBNN, prior knowledge leads to a significantly steeper learning curve. This in turn allows scaling up the learning system to more complex learning tasks. We conjecture that with increasing task complexity, the analytic component of learning will gain in importance. This is because for each inductively derived target value, the analytic component of EBNN will generate n target slopes, where n is the input dimension of the target network. In the ideal case, a single training example in EBNN might therefore be as effective as $n + 1$ training examples in pure inductive learning. These numbers match very roughly our empirical findings.

The importance of inductive learning lies in its ability to extend the system's knowledge beyond that which can be inferred from its starting knowledge. It is obvious in domains such as robot control and speech recognition that manually developing a perfect domain theory is unrealistic. In the case of Xavier, for example, one would have to apply an incredible effort to manually input action models that would meet the requirements of purely analytical learning. An appropriate domain theory would have to include accurate descriptions of the particular sensor noise of ultrasonic and visual sensors, as well as the physics of the robot and its environment. Even the communication delay caused by heavy network traffic would have to be explainable, since it matters crucially for the outcome of the robot's actions. While past research has illustrated the feasibility of hand-coding models for idealized and usually noise-free learning scenarios [Laird and Rosembloom, 1990], [Mitchell, 1990], in a domain as complex and as stochastic as Xavier's world this will be an unreasonably difficult undertaking. The importance of induction is that it enables learning observed empirical regularities that can contradict prior knowledge.

4 Induction and Analysis in Soar

Since Soar is intended as a general architecture for intelligent systems, the question of how to best incorporate both learning mechanisms within Soar seems central to its success. The first subsection below briefly summarizes current approaches, and the following subsection considers the possibility of incorporating EBNN within Soar.

4.1 Current Learning Mechanisms in Soar

Soar has a single, analytical learning mechanism embedded in the architecture: chunking. Chunking is a variant of explanation-based learning, as discussed in [Rosenbloom and Laird, 1986]. For the current discussion, the key points regarding learning in Soar are:

- Chunking is the only architectural learning mechanism in Soar. It has been claimed [Laird *et al.*, 1986] that chunking is a sufficient architectural learning mechanism.
- The chunking mechanism preserves the correctness of Soar's prior knowledge. More precisely, each production rule created by chunking follows *deductively* from previous rules in the system. If the prior knowledge is correct, any rule learned by chunking will also be correct. If prior knowledge is incorrect, learned rules may be incorrect.
- In cases where Soar's initial knowledge is incorrect *and internally inconsistent*, chunking may learn contradictory rules when given multiple examples. In this case, the ill effects of one incorrect rule may be screened by a second (contradictory) rule that

applies to some of the same situations (e.g., see [Rosenbloom *et al.*, 1987]). Whether this beneficial screening will occur depends on the form of the initial knowledge, and on the sequence in which training examples are presented. Thus, while learning of individual rules is a truth-preserving step, learning sets of rules from inconsistent initial knowledge can lead to behavior that improves over the behavior of the initial inconsistent knowledge. This is possible because of the screening of incorrect rules. Because this effect depends on the observed training examples, and can result in behavior not produced by the initial knowledge, we will consider this an inductive effect.

- Explicit inductive learning methods have also been implemented on top of the Soar architecture by providing problem spaces that perform induction (e.g., [Rosenbloom and Aasman, 1990], [Miller and Laird, 1991]). The detailed implementation of this inductive mechanism relies on the above effect of screening incorrect rules by learning new rules that better fit the observed data. The explicit programming of induction in problem spaces is accomplished by providing a problem space in which reasoning traces follow the general-to-specific ordering of possible concepts, and in which chunking provides the mechanism for creating new rules from reasoning traces in this problem space. The initial knowledge in this problem space is general enough to "explain" any example observed by the system, so that the particular sequence of examples encountered determines which reasoning traces are produced, and hence which productions are learned.

As the above summary indicates, Soar's chunking mechanism provides a form of analytical learning, and the screening effect of new learned rules on inconsistent prior knowledge allows for an inductive component to Soar's learning. Whereas the capabilities of Soar's analytical learning mechanism have been convincingly demonstrated in a variety of domains (e.g., [Tambe *et al.*, 1992]), its inductive learning capabilities have yet to be demonstrated on the same scale.

Our assessment is that while combining inductive and analytical learning will be important to the goal of Soar as a general model of learning, we foresee two difficulties with the current strategy of combining an architectural chunking mechanism with explicit programming of inductive mechanisms in problem spaces. First, it is not clear whether an inductive process that produces new rules but does not refine old rules can efficiently support statistical induction in which each example contributes a small change to a continuously evolving hypothesis. For example, consider the problem of learning to drive Pomerleau's ALVINN system. To succeed, Soar would have to acquire control knowledge from training data involving many hundreds of noisy, complex images, for which inductive regularities emerge only from the statistics of large numbers of training examples. In this kind of application, initial knowledge is difficult to derive, so that analytical learning is inapplicable and the burden falls on inductive mechanisms. Here, it appears that Soar's inductive mechanism would require producing a new production each time a new training example was encountered that was inconsistent with the current hypothesis, leading to an enormous number of generated rules that are later screened. Second, it is not clear how Soar's current inductive processes would behave in the face of noise in the training data. Because new rules screen old, one can imagine a scenario in which noisy data leads the system to learn rule A, then to screen it by rule B (which is consistent with the new example), then to relearn A, etc. While it has not been demonstrated that such difficulties are insurmountable by the proposed Soar inductive methods, neither has it been demonstrated that they can successfully deal with these difficulties.

Because EBNN has been demonstrated to successfully learn statistical regularities from high dimensional, noisy data, guided by approximate domain knowledge, it is worthwhile to consider the feasibility of incorporating it within a Soar-like architecture for problem solving and learning. This possibility is considered in the following section.

4.2 Embedding EBNN in Soar

Here we consider the possible incorporation of the EBNN learning mechanism within a Soar-like architecture. The benefit of such a union would be to combine the inductive-analytical learning capabilities of EBNN with the architectural capabilities of Soar to organize learning and problem solving. One way to incorporate EBNN within Soar would be to substitute EBNN for chunking, with the following correspondences:

- The problem solving traces that provide dependencies used by chunking would become the explanations used by EBNN.

- As with chunking, EBNN would be invoked each time a problem-solving impasse occurs (i.e., whenever the next search action is non-obvious). The types of information learned (to preempt subsequent impasses) would be the same as for chunking.

- Unlike chunking, in which a distinct rule is learned from each explanation, in EBNN a distinct network would be learned only for each distinct type of impasse for each distinct problem space. When future impasses of the same type occurred for the same problem space, the existing network would be refined (both analytically and inductively) using the EBNN algorithm.

- In order for the explanations to provide slope information for EBNN, the production rules on which they were based would be modeled by separate neural nets – one for each type of rule. In general, for each Soar production rule concluding some value for attribute A, a corresponding network for concluding A would be created, and used to provide slope information whenever the corresponding rule was involved in some explanation.

- In Soar, an impasse is resolved once deliberation chooses how to continue beyond the search impasse (e.g., by selecting which operator to apply next in the search). At this point, chunking is invoked. In order to support the inductive component of EBNN learning, and to allow for the possibility that deliberation was based on incorrect knowledge, a separate empirical evaluation of the impasse resolution will be required (e.g., to determine whether the selected operator truly succeeds). This empirical evaluation might be obtained, for example, by monitoring the final outcome of the search, or by estimating its likelihood of success based on independent knowledge (e.g., as in reinforcement learning).

Let us refer to such a system as EBNN-Soar, and to the current version of Soar as Chunking-Soar. Key differences between these two systems would include:

- Learning in EBNN-Soar would support pure induction at the architecture level, as well as analytical learning and blends of these two. We expect this would provide more robust learning in the face of approximate prior knowledge and noise in the training data.

- Analytical learning in EBNN-Soar will be slower than in Chunking-Soar. More precisely, analytical learning in EBNN-Soar will incrementally refine the neural network that encodes knowledge for a particular impasse/problem-space pair, mediated by the companion inductive learning component of EBNN. This will lead to less abrupt changes to system knowledge than in Chunking-Soar, where a complete rule is formulated based solely on the analysis of a single example. Others have noted [Rosenbloom, 1983] that learning in Chunking-Soar may be too abrupt to be a correct model of human learning.

- EBNN-Soar may avoid the average growth effect encountered by Chunking-Soar [Tambe *et al.*, 1992], in which the large number of learned chunks can lead to significant slowdown in processing. By representing knowledge using a bounded collection of neural nets (one per <impasse, problem-space> pair), the cost of applying such knowledge cannot grow indefinitely.

- Because it would use a bounded set of networks for representing learned knowledge, EBNN-Soar may encounter difficulties not present in Chunking-Soar. In particular, its bounded-space representation of control knowledge may have too little capacity to accurately represent complex control knowledge. As a result, it may be necessary to develop a more elaborate memory organization than the simple approach sketched here.

5 Summary and Conclusions

We began this chapter with the question of how to best combine inductive and analytical learning in an architecture that seeks to cover the range of learning exhibited by intelligent systems such as humans. While this question is far from answered, the EBNN algorithm described here provides a general learning method for robustly combining these two types of learning. We have presented experimental results showing the use of EBNN for learning mobile robot control strategies, guided by approximate, previously learned knowledge. We have also considered briefly the way in which Soar achieves analytical and inductive learning, and the possibility of incorporating EBNN within a Soar-like architecture. The main points of this paper include:

- Both inductive and analytical learning mechanisms will be needed to cover the range of learning exhibited by humans and other intelligent systems. Analytical mechanisms are required in order to scale up to learning complex concepts, and to handle situations in which available training data is limited. Inductive mechanisms are required in order to learn in situations where prior knowledge is incomplete or incorrect.

- Explanation-based neural network (EBNN) learning provides a robust combination of inductive and analytical learning. Experimental results demonstrate that EBNN can learn to control a mobile robot, from noisy data including vision, sonar, and laser range sensors, and based on approximate knowledge that was previously learned by the robot itself. Given strong prior knowledge, EBNN learns from considerably less data than pure induction (exemplified by the neural network Backpropagation algorithm). As the accuracy of this prior knowledge decreases, EBNN's ability to generalize degrades gracefully until it reaches the same level of performance as pure induction.

- Soar's architectural learning mechanism, chunking, is a purely analytical mechanism. Recent research on Soar has explored ways of incorporating inductive learning mechanisms by including problem spaces that, when coupled with chunking, result in inductive learning within Soar. We have suggested an alternative way to combine analytical and inductive learning within a Soar-like architecture: replace Soar's chunking mechanism by EBNN. We believe such an architecture would have significant advantages in handling statistical learning from noisy data. Such a change would introduce changes to the architecture's representation (neural network versus symbolic productions), memory organization (one neural network replaces multiple productions), and scaling properties (reducing the average growth effect problem, but raising new problems arising from the bounded expressive capability of fixed-size networks). While the details of such an architecture remain to be designed, we believe the demonstrated abilities of EBNN warrant further exploration of this alternative.

6 Acknowledgments

This paper has benefited from a number of discussions over the years with Allen Newell, John Laird, and Paul Rosenbloom regarding the sufficiency of chunking and the potential role of inductive learning in Soar.

This research is sponsored in part by the National Science Foundation under award IRI-9313367, and by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of NSF, Wright Laboratory or the United States Government.

References

[Ahn and Brewer, 1993] Woo-Kyoung Ahn and William F. Brewer. Psychological studies of explanation-based learning. In Gerald DeJong, editor, *Investigating Explanation-Based Learning*. Kluwer Academic Publishers, Boston/Dordrecht/London, 1993.

[Barto *et al.*, 1991] Andy G. Barto, Steven J. Bradtke, and Satinder P. Singh. Real-time learning and control using asynchronous dynamic programming. Technical Report COINS 91-57, Department of Computer Science, University of Massachusetts, MA, August 1991.

[Chi and Bassok, 1989] Michelene T.H. Chi and Miriam Bassok. Learning from examples via self-explanations. In Lauren B. Resnick, editor, *Knowing, learning, and instruction : essays in honor of Robert Glaser*. L. Erlbaum Associates, Hillsdale, N.J., 1989.

[DeJong and Mooney, 1986] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[Doorenbos, 1993] Robert E. Doorenbos. Matching 100,000 learned rules. In *Proceeding of the Eleventh National Conference on Artificial Intelligence AAAI-93*, pages 290–296, Menlo Park, CA, 1993. AAAI, AAAI Press/The MIT Press.

[Fu, 1989] Li-Min Fu. Integration of neural heuristics into knowledge-based inference. *Connection Science*, 1(3):325–339, 1989.

[Gullapalli, 1992] Vijaykumar Gullapalli. *Reinforcement Learning and its Application to Control*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1992.

[Khatib, 1986] Oussama Khatib. Real-time obstacle avoidance for robot manipulator and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[Laird and Rosembloom, 1990] John E. Laird and Paul S. Rosembloom. Integrating execution, planning, and learning in soar for external environments. In *Proceeding of the Eighth National Conference on Artificial Intelligence AAAI-90*, pages 1022–1029, Menlo Park, CA, 1990. AAAI, AAAI Press/The MIT Press.

[Laird *et al.*, 1986] John Laird, Paul Rosenbloom, and Allen Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

[Lin, 1992] Long-Ji Lin. *Self-supervised Learning by Reinforcement and Artificial Neural Networks*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1992.

[Masuoka, 1993] Ryusuke Masuoka. Noise robustness of EBNN learning. In *Proceedings of the International Joint Conference on Neural Networks*, October 1993.

[Miller and Laird, 1991] Craig S. Miller and John E. Laird. A constraint-motivated lexical acquisition model. In *Proceedings of the Thirteenth Annual Meeting of the Cognitive science Society*, pages 827–831, Hillsdale, NJ, 1991. Erlbaum.

[Minton *et al.*, 1989] Steve Minton, Jaime Carbonell, Craig A. Knoblock, Dan R. Kuoka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40:63–118, 1989.

[Mitchell and Thrun, 1993] Tom M. Mitchell and Sebastian B. Thrun. Explanation-based neural network learning for robot control. In S. J. Hanson, J. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 287–294, San Mateo, CA, 1993. Morgan Kaufmann.

[Mitchell *et al.*, 1986] Tom M. Mitchell, Rich Keller, and Smadar Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Mitchell, 1990] Tom M. Mitchell. Becoming increasingly reactive. In *Proceedings of 1990 AAAI Conference*, Menlo Park, CA, August 1990. AAAI, AAAI Press / The MIT Press.

[O'Sullivan, 1994] Joseph O'Sullivan. Xavier manual. Carnegie Mellon University, Learning Robot Lab Internal Document - contact josullvn@cs.cmu.edu, January 1994.

[Ourston and Mooney, 1994] Dirk Ourston and Raymond Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:311–344, 1994.

[Pazzani *et al.*, 1991] Michael J. Pazzani, Clifford A. Brunk, and Glenn Silverstein. A knowledge-intensive approach to learning relational concepts. In *Proceedings of the*

Eighth International Workshop on Machine Learning, pages 432–436, Evanston, IL, June 1991.

[Pomerleau, 1989] D. A. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. Technical Report CMU-CS-89-107, Computer Science Dept. Carnegie Mellon University, Pittsburgh PA, 1989.

[Qin *et al.*, 1992] Yulin Qin, Tom M. Mitchell, , and Simon Herbert. Using EBG to simulate human learning from examples and learning by doing. In *Proceedings of the Florida AI Research Symposium*, pages 235–239, April 1992.

[Rosenbloom and Aasman, 1990] Paul S. Rosenbloom and Jans Aasman. Knowledge level and inductive uses of chunking (ebl). In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 821–827, Boston, 1990. AAAI, MIT Press.

[Rosenbloom and Laird, 1986] Paul S. Rosenbloom and John E. Laird. Mapping explanation-based generalization onto soar. Technical Report 1111, Stanford University, Dept. of Computer Science, Stanford, CA, 1986.

[Rosenbloom *et al.*, 1987] Paul S. Rosenbloom, John Laird, and Allen Newell. Knowledge level learning in soar. Technical Report AIP-8 (Artificial Intelligence and Psychology Project), Carnegie Mellon University, Pittsburgh, PA 15213, 1987.

[Rosenbloom, 1983] Paul Rosenbloom. *The Chunking of Goal Hierarchies: A model of Practice and Stimulus-Response compatibility*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1983. Technical Report CMU-CS-83-148.

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol. I + II*. MIT Press, 1986.

[Shavlik and Towell, 1989] Jude W. Shavlik and G.G. Towell. An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1(3):231–253, 1989.

[Simard *et al.*, 1992] Patrice Simard, Bernard Victorri, Yann LeCun, and John Denker. Tangent prop – a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 895–903, San Mateo, CA, 1992. Morgan Kaufmann.

[Sutton, 1988] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 1988.

[Tambe *et al.*, 1992] Milind Tambe, Robert Doorenbos, and Allen Newell. The match cost of adding a new rule : a clash of views. Technical Report CMU-CS-92-158, Carnegie Mellon University, Pittsburgh, PA 15213, 1992.

[Tesauro, 1992] Gerald J. Tesauro. Practical issues in temporal difference learning. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 259–266, San Mateo, CA, 1992. Morgan Kaufmann.

[Thrun and Mitchell, 1993] Sebastian B. Thrun and Tom M. Mitchell. Integrating inductive neural network learning and explanation-based learning. In *Proceedings of IJCAI-93*, Chamberry, France, July 1993. IJCAI, Inc.

[Towell and Shavlik, 1989] Geoffrey G. Towell and Jude W. Shavlik. Combining explanation-based learning and neural networks: an algorithm and empirical results. Technical Report 859, University of Wisconsin-Madison, Computer Science, 1989.

[Watkins, 1989] Chris J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, England, 1989.

[Winston *et al.*, 1983] P.H. Winston, T.O. Binford, B. Katz, and M. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 433–439, Washington D.C., 1983. Morgan Kaufmann.