Figure 1: **Combining inductive and analytical learning:** In the ideal case, a learning system deals with all levels of domain theories, i.e., it is *robust* with respect to severe errors therein. It operates purely inductively if no domain theory is available or the domain theory is random, and purely analytically if the domain theory is perfect.

form they require correct and complete prior knowledge of the domain. In contrast, *inductive learning methods* require no such prior knowledge, but rely instead on many more training examples to guide generalization, together with some syntactic inductive bias. One of the major open problems in machine learning is to combine analytical and inductive learning in order to gain the benefits of both approaches: reduced requirement for training data, and robustness with respect to poor prior knowledge.

Figure 1 illustrates the spectrum of domain theories over which a general learning system should be able to operate. At present, we have inductive learning methods that operate well at the leftmost point on the spectrum, in which no domain theory is available. We also have explanation-based methods that operate well on the right (under certain assumptions about the character of potential errors in the domain knowledge). We seek a single unified method, which is

- **Robust** with respect to severe errors in the domain theory, i.e., it should operate across the entire spectrum. In particular, if no domain theory is available (or one that is even worse than random), we desire that the system learns as well as a purely inductive system. At the other extreme, if perfect knowledge is available, the system should perform comparably to current explanation-based methods.
- **General**, i.e., it should be able to employ background domain knowledge that it has previously learned from scratch, as well as knowledge provided by the designer. In particular, we are interested in methods that can operate under a broad variety of domain theory errors, such as those typical of inductively learned do-

Figure 2: **Episode:** Starting with the initial state s_1, the action sequence $a_1, a_2, \ldots, a_{n-1}$ was observed to produce the final state s_n, a goal state. The domain knowledge represented by neural networks can be used to *explain* how the observed state-action sequence resulted in achieving the goal. EBNN extracts slopes of the target function (i.e., the partial derivatives of the goal feature of the final state with respect to all features of the initial state) from this explanation.

main knowledge. We are also interested in interleaving learning of the domain theory and the target concept.

- **Noise tolerant**, i.e., it should be able to learn from noisy data. Noise may be present both in the features that describe instances, and in the given training classifications.

2 The EBNN Learning Algorithm

EBNN is an explanation-based learning method utilizing neural network representations that seeks to achieve the above three properties. In EBNN, the domain theory is represented by a collection of artificial neural networks. The target function to be learned is represented separately, either by an additional neural network or by an alternative representation for real-valued functions (e.g., a nearest neighbor scheme). As in symbolic EBL, EBNN uses its domain theory to guide learning of the target function by explaining and analyzing each observed training example of the target function in terms of the domain theory. The domain theory itself may be learned from scratch using Backpropagation [Rumelhart *et al.*, 1986] or some other neural network learning procedure, either before or during learning of the target function.

2.1 Neural Network Domain Theories

To illustrate EBNN, consider an agent (perhaps a robot) which must learn a strategy for choosing which of its actions to apply in any given state in order to achieve its goal. Consider, for example, the episode shown in Figure 2. Starting with an initial state s_1, the sequence of actions $a_1, a_2, \ldots, a_{n-1}$ is observed to produce the goal state s_n. The learning task in this case is to acquire the concept "the class of states, s, for which the action a will lead eventually to a goal state." The target function in this case is a function from states and actions to $[0, 1]$ (i.e., a 1 indicates that executing this action in this state leads to the goal). Once learned, this evaluation function allows the agent to select actions that achieve its goal, as in [Watkins, 1989], [Barto *et al.*, 1991].

2.2 Explaining and Analyzing Observed Episodes

One could apply standard explanation-based learning methods to this problem, provided the agent initially possessed a perfect domain theory describing the effects of its actions on the world state. Instead, we consider the case where the robot has only an approximate, previously learned theory of the effects of its actions. This domain theory is represented by a collection of neural networks, one for each action. The network characterizing action a_i takes as input the description of an arbitrary state, and produces as output a description of the predicted resulting state (i.e., each network represents the same information typically represented by symbolic precondition-postcondition action descriptions). EBNN applies these action model networks to explain and learn from each observed episode in which it achieves its goal. More precisely, EBNN applies the following three-step process to each observed episode:

1. **Explain:** An *explanation* is a post-facto prediction of the observed episode using the domain knowledge. Explanations are constructed by using the neural network domain theory to post-facto predict, and thus explain, how action a_1 applied at state s_1 led to the observed state s_2, how a_2 led to s_3, and so on. Note that predicted states usually deviate from the observed ones since inductively learned domain theories are only approximately correct.
2. **Analyze:** The role of the explanation is to elucidate how achieving the final goal *depends on* the various features of the observed initial state, s_1. In symbolic EBL, this dependence is used to extract the *weakest precondition* under which the same explanation would have produced the same outcome. Since EBNN represents its domain theory by neural networks, it is difficult to extract weakest preconditions. However, since neural networks are real-valued differentiable functions, EBNN uses the dependencies in the explanation to extract the *derivatives (i.e., slopes)* of the final goal feature with respect to each feature of the initial state, s_1. More specifically, EBNN examines the specific chain of neural net activations and weights in the explanation to analytically extract these derivatives.

   To see how this is done, consider the last state-action pair $\langle s_{n-1}, a_{n-1} \rangle$ shown in Figure 2, which led to the goal state s_n. Neural networks represent differentiable functions. Using the last step of the explanation for this episode, the slopes of the goal features of the predicted final state s_n with respect to s_{n-1} and a_{n-1} can be extracted by computing the derivative of the neural network function. These slopes describe the dependence of the final state s_n on the previous state s_{n-1}, and action a_{n-1}. In particular, they measure how infinitesimally small changes applied to s_{n-1} or a_{n-1} will change the final state s_n. The extraction of slopes can be chained back through the entire episode by applying the chain rule of differentiation to the multiple explanation steps. The result of this analysis is the set of derivatives (slopes) of the target concept (goal state) with respect to each state-action pair in the observed episode. As stated above, these slopes measure the *dependence* of the target concept on each of the features of the states and actions in the observed episode. State features believed (by the domain theory) to be

Figure 3: **Fitting slopes:** Let f be a 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.

irrelevant to achieving the final goal will have partial derivatives of zero, whereas large slope values indicate the presence of strongly relevant features.

3. **Refine:** The slopes extracted from the explanation, along with the observed training example itself, are used to refine the learner's description of the target function. The target function in EBNN is represented by a separate neural network (or any other representation appropriate for approximating real-valued functions from sample values and sample slopes). This target function is incrementally updated with each new training example, both inductively and analytically, to iteratively approximate the true target function. In the episode from our example, each state-action pair in the episode, $\langle s_i, a_i\rangle$, is observed to lead to the goal state and thus becomes a training example for inductively and analytically refining the target network. The **inductive component of learning** corresponds to updating the target network to produce the target output *value* (e.g., 1 if the example leads to achieving the goal). Inductive learning is crucial for compensating for errors in the domain theory. The **analytical component of learning** corresponds to updating the network to fit the target output *slopes*, extracted analytically from the explanation. As shown in Figure 3, these slopes influence the learned network by overriding the default bias of interpolating between observed points. Therefore the analytical component in EBNN enables more correct generalization from less training data, if slopes are sufficiently accurate. In the case that the target function is represented by a neural network, the Backpropagation algorithm can be extended to fit slopes as well as values, as may be found in [Simard *et al.*, 1992].

To summarize, the target function is iteratively approximated by updating it (a) inductively, to fit the empirically observed *training values* of the target function, and (b) analytically, to fit the analytically derived *training slopes* obtained by explaining the observed example in terms of a previously learned domain theory.

2.3 Accommodating Imperfect Domain Theories

Since the domain theory is learned inductively from training instances[3], its accuracy might be arbitrarily

[3]This process of inductively learning the domain theory is not to be confused with learning the target function.

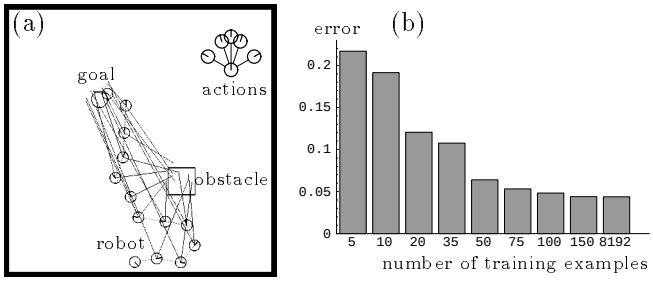

Figure 4: a. The simulated robot world. b. The squared generalization error of the domain theory networks decreases monotonically as the amount of training data increases. These nine alternative domain theories were used in the experiments.

poor, resulting in arbitrarily poor explanations and extracted slopes. How can the learner avoid the damaging effects of such incorrect slopes arising from a poor domain theory?

EBNN reduces the undesired influence of incorrect domain theory predictions by estimating the *accuracy* of the extracted slopes, based on the fit between the observed sequence of states and those predicted by the explanation (the underlying assumption *"prediction errors measure slope errors"* is called LOB*). More specifically, each time the domain theory is used for post-facto predicting a state s_{k+1}, its prediction $s_{k+1}^{\text{predicted}}$ may deviate from the observed state $s_{k+1}^{\text{observed}}$. We define the 1-step prediction accuracy at state s_k, denoted by $c_1(i)$, as 1 minus the normalized prediction error:

$$c_1(i) \quad := \quad 1 \; - \; \frac{||s_{i+1}^{\text{predicted}} - s_{i+1}^{\text{observed}}||}{\text{max\_prediction\_error}}$$

For a given episode we define the n-step accuracy $c_n(i)$ as the product of the 1-step accuracies in the next n steps. The n-step accuracy, which measures the accuracy of the extracted slopes n steps away from the end of the episode, possesses three desirable properties: a. It is 1 if the learned domain theory is perfectly correct, b. it decreases monotonically as the length of the chain of inferences increases, and c. it is bounded below by 0. The n-step accuracy is used to determine the ratio by which the analytical and inductive components are weighted when learning the target function. If an observation is n steps away from the end of the episode, the analytically derived training information (slopes) is weighted by the n-step accuracy times the weight of the inductive component (values). Although the experimental results reported in Section 2.4 are promising, the generality of this approach is an open question, due to the assumption LOB*.

2.4 Experimental Results

EBNN was evaluated in a simulated robot navigation domain. The world and the action space are depicted in Figure 4a. The learning task is to find an evaluation function Q for which the greedy policy navigates the agent to its goal location (circle) from arbitrary starting locations, while avoiding collisions with the walls or the

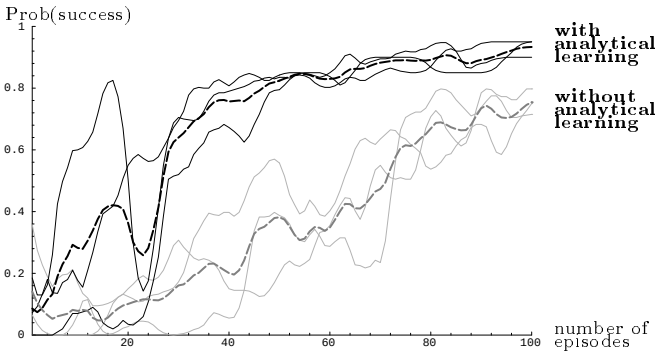

Figure 5: **EBNN:** Performance curves for EBNN with (black) and without (gray) analytical training information (slope fitting) for three examples each, measured on an independent set of problem instances. The dashed lines indicate average performance. In this experiment, the agent used well-trained predictive action models as its domain theory.

obstacle (square). States are described by the local view of the agent, in terms of distances and angles to the center of the goal and to the center of the obstacle. Note that the world is deterministic in these experiments, and there is no sensor noise.

In order to allow exploration of the robot environment and to compensate for the necessary non-optimal action choices, we applied EBNN to Watkins' $\mathcal{Q}$-Learning [Watkins, 1989] together with Sutton's temporal difference learning $TD(\lambda)$ [Sutton, 1988] (with $\lambda = 0.7$ and a reward *discount* $\gamma = 0.8$)[4]. Each discrete action was modeled in the domain theory by a separate neural network. We used neural network Backpropagation learning for learning action models. The evaluation functions $\mathcal{Q}$ were approximated by an instance-based local approximation technique, modeling $\mathcal{Q}$ separately for each action. In this instance-based technique, each training instance together with its slopes was explicitly memorized. Given a new point as a query, generalization was achieved by fitting a local second order polynomial over the three nearest neighbors in the instance memory. This polynomial fit both the values and the slopes. In our initial experiments, this instance-based technique was found to outperform neural networks for representing the target functions.

Experiment 1: *"What is the impact of the analytical component of EBNN, given a strong domain theory?"* In the first experiment, we initially allowed the agent to train each of the action modeling networks that form its domain theory using 8 192 randomly generated training examples. This results in a fairly accurate, but still imperfect, domain theory. Figure 5 shows results of applying EBNN using this pre-learned domain theory, compared to using just the inductive learning component alone. In this figure, the performance was measured on an independent test set of 20 initial locations. Both techniques exhibit asymptotically the same performance and learn the desired control function successfully. However, there is a significant reduction in the number of training episodes needed by EBNN in order to reach the same level of performance as inductive learning (i.e., the EBNN learning curve is steeper, indicating more correct generalization from the same data).

[4]We will omit the somewhat lengthy details here, since they are not essential for the understanding of EBNN. See [Mitchell and Thrun, 1993b] for a detailed description.

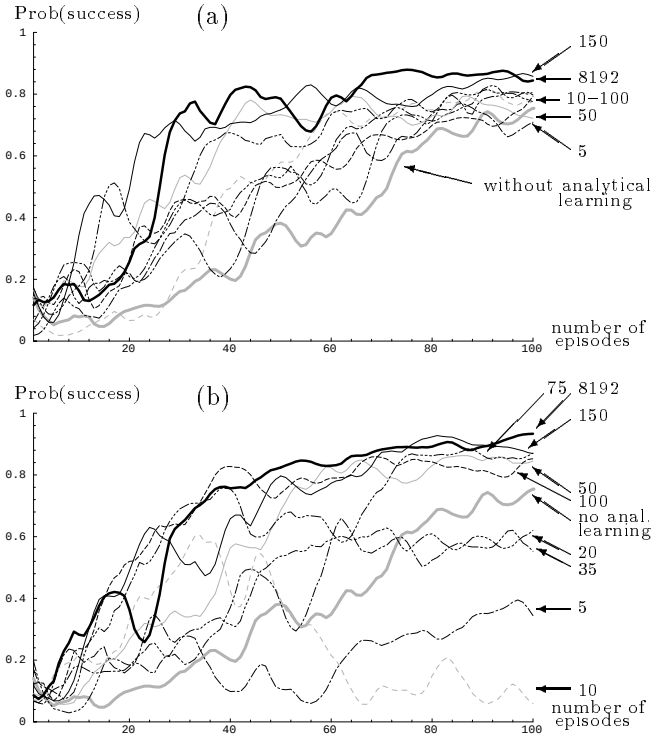

Figure 6: How does domain knowledge improve generalization? a. Averaged results for EBNN domain theories of differing accuracies, pre-trained with from 5 to 8 192 training examples for each action model network. In contrast, the bold gray line reflects the learning curve for pure inductive learning, i.e., $\mathcal{Q}$-Learning and TD(λ). b. Same experiments, but without weighting the analytical component of EBNN by its accuracy, illustrating the importance of LOB*. 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.

Experiment 2: *"How does EBNN degrade with progressively weaker domain theories?"* We repeated Experiment 1 using weaker domain theories, trained with 5, 10, 20, 35, 50, 75, 100, 150, and 8 192 training examples per action network (c.f. Figure 4b). Figure 6a shows clearly that (1) EBNN outperforms purely inductive learning at all accuracy levels, and (2) the more accurate the domain theory, the steeper the learning curve. Thus, EBNN in this experiment degrades gracefully to the performance of a pure inductive system as the accuracy of the domain theory decreases.

Experiment 3: *"How important is the heuristic LOB* for the graceful degradation of EBNN?"* We then repeated Experiment 2 without weighting the slopes relative to their observed accuracy. Figure 6b shows the results. For high-quality domain theories, the learning curves are not affected. As the quality of the domain theory decreases, however, the learning speed of EBNN without LOB* is significantly worse than pure inductive

learning. This result justifies LOB* and illustrates its importance in EBNN.

3 Related work

Recent research has produced a variety of proposals for combining inductive and analytical learning methods (e.g., see the Workshop of Combining Inductive and Analytical Learning [Machine Learning Workshop, 1989]), though none of these achieves a final solution to the problem. Indeed, as research in this area matures we may find that multiple approaches are needed, depending on the type of representations used for the domain theory and target function (e.g., first order domain theories versus propositional, or discrete-valued target functions versus real-valued). Approaches differ in the types of domain theory imperfections they can accommodate, the representations they use for the domain theory and the target concept, and the particular mechanisms by which they combine inductive and analytical components. Mechanisms for combining induction and analysis can be grouped roughly into three categories:

- **Analytical, then inductive.** Here, each training example is first generalized analytically, and inductive methods are then applied to the results. For example, Hirsh's IVSM [Hirsh, 1989] applies explanation-based generalization to each training example, then combines the results from different examples using an inductive method based on version spaces. In some systems, it is the explanations over which induction is applied (e.g., [Dietterich and Flann, 1988], [Kedar-Cabelli, 1988]). In others, inductive methods are applied to the remaining unexplained features to catch relevant features that may have been missed by the domain theory (e.g., [Mooney and Ourston, 1989]).
- **Inductive, then analytical.** Lebowitz [Lebowitz, 1987] has suggested an approach in which statistical regularities are first found from a large set of data. These empirical regularities (e.g., "midwest congressman typically vote in favor of farm subsidies") are then explained (e.g., "midwest states contain many farmers", "congressmen typically vote to help their voters") in order to further refine them and guide the search for variants on this regularity.
- **Interleave inductive and analytical processes.** Some systems interleave inductive and analytical steps. For example, Bergadano and Giordana [Bergadano and Giordana, 1990] construct the explanation not for one example, but simultaneously considering all available examples. Systems such as VanLehn's [VanLehn, 1987], Hall's [Hall, 1988], and Pazzani's [Pazzani, 1989] learn by inductively filling in the gaps in incomplete explanations. Others such as Widmer [Widmer, 1989] and Mahadevan [Mahadevan, 1989] use abstracted domain knowledge such as determinations [Russell, 1988] to form abstract explanations, and then to specialize the domain theory based on the observed example. Oursten and Mooney propose a system that inductively refines an initial domain theory based on noisy training data [Ourston and Mooney, 1991] using ID3 [Quinlan, 1986] as the inductive component. Like EBNN, their system is able to deal with a whole spectrum of domain theories, from weak to strong. Rosenbloom and Aasman [Rosenbloom and Aasman, 1990] and Miller and Laird [Miller and Laird, 1991] have demonstrated that induction can be achieved with purely analytical learning mechanisms by inserting appropriate "inductive rules" into the domain theory.

The EBNN method presented here falls into the first of these categories: each example is explained to extract general information, and the results of these explanations are then combined. However, EBNN differs significantly from previous explanation-based approaches in that it is based on neural network representations for both the domain theory and the target concept. This leads to two useful properties. First, it enables the use of standard inductive methods for learning the domain theory from noisy data (e.g., it can use Backpropagation [Rumelhart *et al.*, 1986], or EBNN itself). Second, it provides a natural method for incrementally refining the learned target concept based both on observed training examples (the inductive component) and on information extracted from explanations (the analytical component).

Researchers working on neural network learning methods have also noted the importance of using prior knowledge to learn more complex functions from less training data. For example, Simard and colleagues [Simard *et al.*, 1992] have shown that network training algorithms can be developed that fit certain types of user-provided constraints on the target function. They developed a system for recognizing visual objects, constraining the network output to be invariant to translation of the object within the image. The key difference between this work and EBNN is that in Simard's work the designer must embed his own knowledge into a task-specific learning algorithm, whereas EBNN is a task-independent method that learns and then uses its own prior (learned) knowledge to constrain subsequent learning.

Others, such as Shavlik and Towell [Shavlik and Towell, 1989], Fu [Fu, 1989] and Mahoney and Mooney [Mahoney and Mooney, 1992] have proposed methods that use explicitly represented domain knowledge to bias neural network learning by initializing the network to reflect this prior knowledge. In their methods, a symbolic domain theory is used to define a neural network (both its topology and weights) so that it infers exactly the same example classifications as the given domain theory. This network is then refined inductively using Backpropagation. EBNN differs from this approach in that (1) EBNN constructs a distinct explanation for each observed example, rather than "compiling" the domain theory in one shot into a neural network, (2) EBNN uses a self-learned domain theory represented by neural networks, rather than a user-provided domain theory represented by symbolic rules, (3) because it uses the domain theory to explain each example, EBNN can use available data to refine both the domain theory and target concept, with domain theory improvements having a direct effect on subsequent analytical steps.

4 Discussion

This paper presents a learning method, EBNN, which combines inductive and analytical learning. An inductively learned approximate domain theory is used to guide learning a separate target function, by a combined inductive and analytical process. The domain theory in EBNN is represented by a collection of learned neural networks (e.g., one to model each action in our robotic example). The target function (e.g., the state-action evaluation function in our example) may be represented by any approximator for real-valued functions that can fit both training values and training slopes of the target function. Fitting the observed training values provides a purely inductive component for learning the target function, whereas fitting the slopes extracted from explanations provides the analytical component. EBNN is demonstrated to learn the target function better from fewer examples, when compared against purely inductive learning, and to degrade gracefully as the quality of the domain theory decreases. The LOB* heuristic appears effective as a means for decreasing the contribution of the analytical component for those training examples for which the domain theory produces poor explanations.

EBNN at least partially fulfills all three of the requirements discussed in section 1, i.e., it is *robust*, *general*, and in part *noise tolerant*. It is *robust* to errors in the domain theory, because inaccurate slopes resulting from inaccurate domain knowledge are identified and their effect reduced, via the n-step accuracy estimate, and because the inductive learning component competes with the analytical learning component. It is *general*, since no a priori domain knowledge is required to initialize the system–it can learn the necessary domain theory inductively. Furthermore, it is *noise tolerant* to the extent that neural networks are capable of dealing with noisy training data.

While these first results suggest EBNN is a promising learning method, there are a number of significant issues that warrant further research:

- In contrast to many other approaches to EBL which utilize first-order predicate logic to represent domain knowledge, the EBNN domain theory expressed by neural networks is propositional.
- The capability of EBNN in stochastic domains is unclear, since the LOB* heuristic for weighting the analytical component of learning relies on the observed prediction error of the *deterministic* predictions of the domain theory.
- Derivatives, or slopes, represent only one kind of knowledge that can be analytically extracted from explanations. It would be interesting to extract other forms of knowledge as well, to further accelerate learning.
- There are several fundamental differences between EBNN and explanation-based approaches that utilize symbolic representations. For example, representing the target function by a single neural network, instead of a collection of learned rules, leads to different scaling problems. When learning collections of rules, the learner can encounter a slowdown in overall performance arising from the rising cost of matching an increasing number of learned rules. If the target function is instead represented by a single neural network, there is no corresponding slowdown as learning proceeds. However, a different scaling issue arises: the correctness of the single-network approximation to the target function might degrade when learning very complex target functions. See [Mitchell and Thrun, 1993a] for a detailed discussion of these issues.
- In EBNN, the complete domain theory is learned from scratch. If a priori domain knowledge is available, it will be interesting to study the correspondence and the interaction between learned and pre-given domain theories.

Acknowledgment

We thank Ryusuke Masuoka for his invaluable help in refining the EBNN algorithm and code. Jude Shavlik and Paul Rosenbloom and the CMU Robot Learning Group have contributed useful ideas during discussions of the correspondence between symbolic and neural network EBL methods. We thank Lonnie Chrisman and Rich Goodwin for comments on earlier drafts of this paper.

References

[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, 1991.

[Bergadano and Giordana, 1990] Francesco Bergadano and Aiello Giordana. *Guiding Induction with Domain Theories*, pages 474–492. Morgan Kaufmann, San Mateo, CA, 1990.

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

[Dietterich and Flann, 1988] Tom Dietterich and Nicholas Flann. An inductive approach to solving the imperfect theory problem. In *Proceedings of the AAAI Explanation-Based Learning Symposium*, pages 42–46, March 1988.

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

[Hall, 1988] R. J. Hall. Learning by failing to explain: Using partial explanations to learn in incomplete or intractable domains. *Machine Learning*, 3(1):45–78, 1988.

[Hirsh, 1989] Haym Hirsh. Combining empirical and analytical learning with version spaces. In Bruce Spatz and John Galbraith, editors, *Proceedings of the Sixth International Workshop on Machine Learning*, pages 29–33, San Mateo, CA, June 1989. Morgan Kaufmann Publishers, Inc.

[Kedar-Cabelli, 1988] Smadar T. Kedar-Cabelli. *Formulating Concepts and Analogies according to Purpose*. PhD thesis, Rutgers University, New Brunswick, NJ, 1988.

[Lebowitz, 1987] Michael Lebowitz. Experiments with incremental concept formation: Unimem. *Machine Learning*, 2(2), 1987.

[Machine Learning Workshop, 1989] Workshop on combining empirical and explanation-based learning. In Alberto M. Segre, editor, *Proceedings of the sixth international workshop on machine learning*, pages 1–93, San Mateo, CA, 1989. Morgan Kaufmann.

[Mahadevan, 1989] Sridhar Mahadevan. Using determinations in EBL: A solution to the incomplete theory problem. In Alberto M. Segre, editor, *Proceedings of the sixth international workshop on machine learning*, pages 320–325, San Mateo, CA, 1989. Morgan Kaufmann.

[Mahoney and Mooney, 1992] J. Jeffrey Mahoney and Raymond J. Mooney. Combining symbolic and neural learning to revise probabilistic theories. In *Proceedings of the 1992 Machine Learning Workshop on Integrated Learning in Real Domains*, Aberdeen Scotland, July 1992.

[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.

[Mitchell and Thrun, 1993a] Tom M. Mitchell and Sebastian B. Thrun. Explanation based learning: A comparison of symbolic and neural network approaches. In *Proceedings of the Conference on Machine Learning*, Amherst, MA, 1993. to appear.

[Mitchell and Thrun, 1993b] Tom M. Mitchell and Sebastian B. Thrun. Explanation-based neural network learning for robot control. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1993. Morgan Kaufmann. (to appear).

[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.

[Mooney and Ourston, 1989] Raymond Mooney and Dirk Ourston. Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. In Alberto M. Segre, editor, *Proceedings of the sixth international workshop on machine learning*, pages 5–7, San Mateo, CA, 1989. Morgan Kaufmann.

[Ourston and Mooney, 1991] Dirk Ourston and Raymond J. Mooney. Theory refinement with noisy data. Technical Report AI 91-153, AI Lab, University of Texas at Austin, March 1991.

[Pazzani, 1989] Michael J. Pazzani. Detecting and correcting errors of omission after explanation-based learning. In *Proceedings of IJCAI-89*, pages 713–718, 1989.

[Quinlan, 1986] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[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.

[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.* MIT Press, 1986.

[Russell, 1988] Stewart Russell. Tree-structured bias. In *Proceeding of Eigth National Conference on Artificial Intelligence AAAI-88*, pages 641–645, San Mateo, CA, 1988. Morgan Kaufmann.

[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.

[VanLehn, 1987] Kurt VanLehn. Learning one subprocedure per lesson. *Artificial Intelligence*, 31:1–40, 1987.

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

[Widmer, 1989] Gerhard Widmer. A tight integration of deductive and inductive learning. In Alberto M. Segre, editor, *Proceedings of the sixth international workshop on machine learning*, pages 11–13, San Mateo, CA, 1989. Morgan Kaufmann.