To appear in *Machine Learning: Proceedings of the Tenth International Conference*,
Morgan Kauffman, San Mateo, CA, 1993.

Explanation Based Learning: A Comparison of Symbolic and Neural Network Approaches

Tom M. Mitchell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
Tom.Mitchell@cs.cmu.edu

Sebastian B. Thrun
University of Bonn
Institut für Informatik III
Römerstr. 164, D-5300 Bonn, Germany
Thrun@uran.informatik.uni-bonn.de

Abstract

Explanation based learning has typically been considered a symbolic learning method. An explanation based learning method that utilizes purely neural network representations (called EBNN) has recently been developed, and has been shown to have several desirable properties, including robustness to errors in the domain theory. This paper briefly summarizes the EBNN algorithm, then explores the correspondence between this neural network based EBL method and EBL methods based on symbolic representations.

1 Introduction

Explanation based learning (EBL) is an important paradigm for machine learning because it offers a means of using prior knowledge to generalize more correctly from fewer training examples. EBL was originally conceived as a symbolic learning method ([DeJong and Mooney, 1986], [Mitchell *et al.*, 1986]), based on creating and analyzing explanations derived from symbolic rules. While it has been used successfully to learn control knowledge to speed up search ([Laird *et al.*, 1988], [Minton, 1988]), the chief limit to its usefulness lies in the requirement that its prior knowledge, or domain theory, be complete and correct. Recent research has produced a number of proposals for extending EBL approaches to accommodate imperfect domain theories (e.g., for several dozen, see the Workshop of Combining Inductive and Analytical Learning [Machine Learning Workshop, 1989]). Despite progress in this area, we still have not found the final answer to the question of how to best unify EBL and inductive learning.

An explanation based learning method based purely on neural network representations has recently been proposed [Mitchell and Thrun, 1993], [Thrun and Mitchell, 1993], which provides a means of combining an inductive learning component (neural network Backpropagation) with an explanation based learning component. In preliminary experiments, this algorithm (called EBNN) has been demonstrated to successfully use approximate, previously learned, domain knowledge of widely varying quality to guide generalization. EBNN is a direct analog to symbolic EBL methods in that it maintains separate representations of the domain theory and the target function, and uses the domain theory to explain and generalize each individual training example.

The goal of this paper is to examine the correspondence between EBL methods based on symbolic and neural network representations. To ground the comparison, we consider two specific algorithms: a prototypical pure symbolic EBL algorithm, Prolog-EBG [Kedar-Cabelli and McCarty, 1987], and explanation based neural network learning, EBNN. We first present an overview of the EBNN algorithm. Following this we define the abstract EBL problem and EBL method that is common to both symbolic and neural network approaches. Finally, we examine key differences in detailed representations and algorithms, that lead to different capabilities and convergence properties for these two approaches. The main result revealed by this analysis is a fundamental difference in the types of information that can be extracted from symbolic versus neural network explanations, and a fundamental difference in convergence characteristics when learning a target function represented by sets of rules versus a single neural network.

2 An Overview of EBNN

This section summarizes the EBNN learning algorithm and presents preliminary experimental results illustrating its robustness to errors in the domain theory. Additional details regarding the algorithm and the experimental results are available in [Thrun and Mitchell, 1993] and [Mitchell and Thrun, 1993]. Note this section is a summary extracted from those papers.

Figure 1: **Training Episode:** Starting from 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 derivative of the goal feature of the final state with respect all features of the initial state) from this explanation.

2.1 Neural Network Domain Theories

In EBNN, the domain theory is represented by artificial neural networks. By using neural network learning algorithms such as the Backpropagation procedure [Rumelhart *et al.*, 1986], the domain theory can be learned from scratch and does not require the availability of a priori knowledge from a human trainer. As in symbolic EBL, EBNN uses the domain theory to guide learning of the target concept by explaining and analyzing observed training examples.

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 (eventually) achieve its goal. Consider, for example, the episode shown in figure 1. Starting from 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 is this case is to learn the concept "the class of states, s for which the action a will lead eventually to a goal state." This concept will be represented by a boolean function from states and actions to $\{0, 1\}$. Once learned, this evaluation function allows the agent to select actions that achieve its goal.

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 type of information as 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 in which the agent successfully achieves its goal:

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, why action a_1 applied at state s_1 led to the observed state s_2, why 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 dependency 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, one can use 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. EBNN examines the specific chain of neural net activations and weights in the explanation, to analytically extract these derivatives.

More specifically, consider the last state-action pair $\langle s_{n-1}, a_{n-1} \rangle$ shown in figure 1, which led to the goal state s_n. Neural networks correspond to 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

Figure 2: **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.

slopes can be chained back through the entire episode by applying the chain rule of differentiation to the entire explanation. 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 pointwise *dependence* of the target concept on the features of the states and actions in the observed episode.

3. Refine: The slopes extracted from the explanation, along with the observed training example itself, are used to refine the learner's hypothesis regarding 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 is analogous to symbolic EBL, in which the learned target function is also represented separately from the initial domain theory. However, whereas symbolic EBL produces a new rule for each explained example, EBNN uses just a single network[1] to represent the target function. This single target function network 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, $\langle s_i, a_i\rangle$ 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 2, these slopes influence the learned network by overriding the default bias of interpolating between observed points. Therefore the analytical component in EBNN enables faster 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 concept 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[2], its accuracy might thus be arbitrarily 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 (this heuristic 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. For states 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 heuristic nature of the assumption LOB*.

2.4 Experimental Results in a Simulated Robot Learning Domain

EBNN was evaluated in a simulated robot navigation domain. The world and the action space are depicted in Figure 3a&b. The learning task is to find an evaluation function Q, for which the greedy policy navigates

[1] In our experiments we used 5 networks, one for each action. The important point is that the target function is represented in a fixed (set of) networks, rather than a growing number of rules. We also experimented with a nearest neighbor representation of the target function.

[2] This process is not to be confused with the inductive learning of the target function.

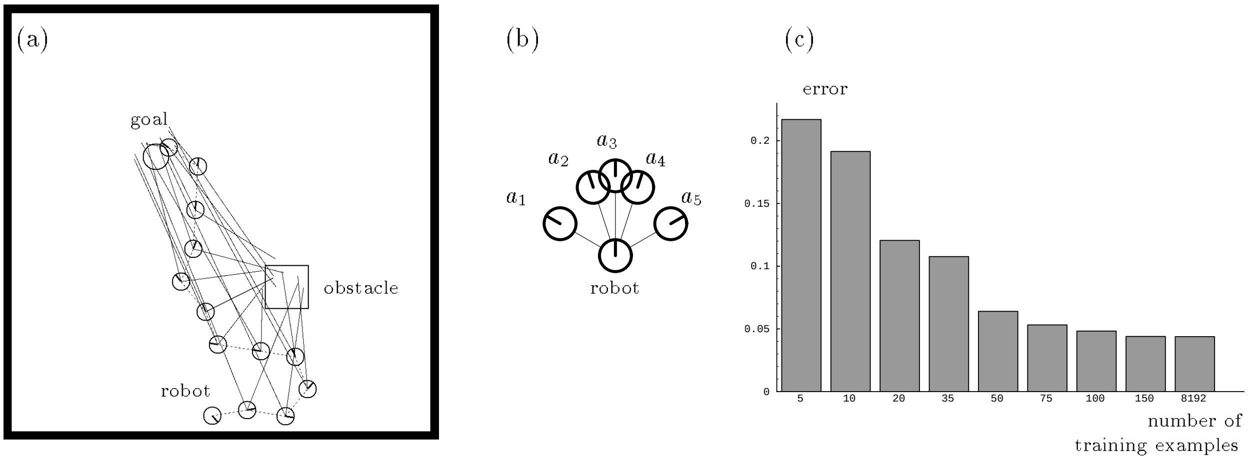

Figure 3: a. The simulated robot world. b. Actions. c. The squared generalization error for each of nine domain theories trained on different numbers of training examples.

the agent to its goal location (circle) from arbitrary starting locations, while avoiding collisions with the walls or the 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' Q-Learning [Watkins, 1989] together with Sutton's *temporal difference learning* $TD(\lambda)$ [Sutton, 1988] (with $\lambda = 0.7$ and a reward *discount* $\gamma = 0.8$)[3]. Each discrete action was modeled by a separate neural network. We used neural network Backpropagation learning for learning action models. The evaluation functions Q were approximated by an instance-based local approximation technique, modeling the 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 outperformed 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 this experiment we first allowed the agent to train each of the action modeling networks that form its domain theory using 8192 randomly generated training examples. This results in a fairly accurate, but still imperfect, domain theory, which was then used by EBNN. The results of EBNN are shown by the thick black line in Figure 4. In contrast, the results of applying standard inductive learning (Backpropagation) are shown by the line marked "without analytical learning". EBNN required significantly less training data than inductive learning in order to reach the same level of performance.

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 8192 training examples per action network. The additional curves in Figure 4 show the results of using EBNN with these weaker domain theories. These curves demonstrate that EBNN degrades gracefully to the performance of pure induction, as the accuracy of the domain theory decreases.

3 Correspondence between Symbolic and Neural Network EBL

As described above, EBNN is the neural network analog of symbolic explanation-based learning methods. In this section we describe the common abstraction of symbolic EBL and EBNN. We then explore the specific differences between these two algorithms, and the significant impact that these algorithmic differences have on their relative capabilities. While there are many variants of symbolic EBL, for the purposes of grounding the discussion we will consider only one prototypical algorithm: Prolog-EBG, as described in [Kedar-Cabelli and McCarty, 1987]. Furthermore, while both neural networks and nearest-neighbor schemes have been used to represent the target function in EBNN, in this discussion we assume the target function is represented by a neural network.

It is also important to note that a variety of techniques have been proposed for combining analytical and in-

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

(a)

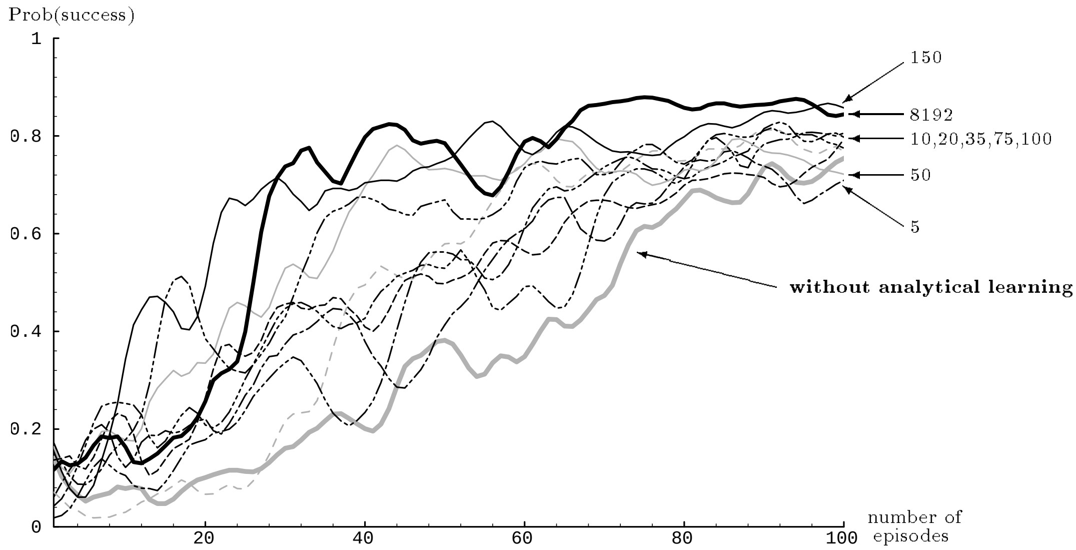

Figure 4: EBNN using various domain theories, compared against pure induction. The bold grey line is the learning curve for pure inductive learning, i.e., Q-Learning and TD(λ). Other curves show the performance of EBNN, using domain theories of differing accuracies, pre-trained with from 5 to 8192 training examples for each action model network. The performance (vertical axis) is measured on an independent test set of 20 starting positions. All curves are averaged over 3 learning runs and are also locally window-averaged.

ductive learning, which do not involve explaining individual training examples, and which are therefore outside the scope of this discussion. These include, for example, Shavlik's [Shavlik and Towell, 1989] and Fu's [Fu, 1989]) methods for using symbolically encoded domain knowledge to initialize the topology and weights of neural networks which are then inductively refined.

3.1 The Explanation-Based Learning Problem

The common problem addressed by the symbolic Prolog-EBG method and the neural network EBNN method can be characterized as follows:

Given:

- **Target function:** the function to be learned.
- **Domain theory:** prior knowledge capable of predicting/explaining the output of the target function given its input
- **Training instances:** examples of input, output pairs of the target function
- **Operationality criterion:** constraints on how the learned function must be described

Determine:

An operational representation of the target function that best fits both the observed training instances and the given domain theory

Note that while both Prolog-EBG and EBNN make use of a domain theory, it is represented in very different forms in the two cases. In Prolog-EBG, the domain theory is described by a collection of horn clauses, whereas in EBNN it is a collection of neural networks. Note that each neural network in the domain theory can be interpreted as a complex inference rule, capable of concluding an output assertion (represented by its output activations) from the input assertion (represented by its input activations).

The operationality criterion, which specifies the acceptable form for the learned target function, is similar in Prolog-EBG and EBNN. In both cases, the target function value is computed directly from the observable attributes of the example. In Prolog-EBG this target function is represented by horn clauses, while in EBNN it is represented by a neural network or some other approximator for real-valued functions.

Notice that the above problem definition requires that the learned function be a best fit to both the training instances and the domain theory. Strictly speaking, Prolog-EBG is used only when the training examples are consistent with the domain theory, and it therefore produces learned rules that constitute a *perfect* fit to both the domain theory and observed examples. However, much current work on symbolic EBL (e.g., [Machine Learning Workshop, 1989], [Mooney and Ourston, 1989], [Pazzani, 1989]) deals with the same class of situations as EBNN: situations in which the observed training examples are not strictly entailed by the domain theory, and for which the goal is therefore to learn a function that best fits both the imperfect domain theory and the observed training examples.

3.2 The Explanation-Based Learning Method

The common abstraction of the Prolog-EBG and EBNN algorithms is to perform, for each observed training example, i, the following steps:

1. *Explain* i, by showing how the domain theory predicts the value of the target function for input i
2. *Analyze* this explanation to determine the relevance of different features of i with respect to the target function
3. *Refine* the current representation of the target function to take into account this new training example and the information about feature relevance extracted from the explanation.

Both Prolog-EBG and EBNN construct similar explanations: chains of inferences based on the domain theory, predicting the observed target function value from attributes of the examples. However, they differ significantly in the ways in which they use their explanations to refine the hypothesized target function.

The key difference in the use of explanations stems from the difference in representing dependencies using symbolic versus neural network domain theories: symbolic rules typically refer to (i.e., match) only a *subset* of the training example attributes, whereas neural networks compute their output based on weighted combinations of *all* example attributes. Therefore, symbolic explanations typically indicate that the target function value depends on only *some subset of the training example attributes*, whereas neural network explanations indicate that the explained value depends to some degree on *every training example attribute*.

In Prolog-EBL, the features of the training example that are mentioned in the symbolic explanation are easily distinguished from those that are not, and the regression of the target concept expression through the explanation yields the weakest preconditions under which that explanation will hold. A new rule is then created using these weakest preconditions for its antecedent, and the explained target function value as its consequent.

With explanations based on neural networks, the process of regressing logical expressions in order to extract weakest preconditions cannot be applied. Instead, the dependency of the target function value on example features is extracted in the form of partial derivatives of the target function with respect to the example attributes, evaluated at the training example point. These partial derivatives, or slopes, summarize the knowledge implicit in the domain theory as it applies to the current explanation. These analytically derived slopes are then used to update the neural network that defines the target function. In particular, the weights of the target function network are adjusted to make its slope closer to the slope suggested by the explanation.

The key differences between symbolic and neural network EBL methods include the following:

Difference in dependencies extracted from explanations. As discussed above, symbolic EBL extracts weakest preconditions from explanations, whereas neural network EBL extracts partial derivatives of the target function with respect to example attributes.

Difference in target function coverage and correctness. As new training examples are encountered, symbolic EBL methods acquire an increasing number of rules that allow it to predict the target function value for an increasing proportion of the space of examples. In contrast, neural network EBL begins with a (very poor) description of the target function that covers the entire space of examples. As it acquires examples, its improvement lies in improving the quality of its approximation to the target function, rather than its coverage. In brief, symbolic EBL learns a function that is always correct and increasingly complete in its coverage, whereas neural net EBL learns a function that is always complete and increasingly correct.

Difference in mechanisms for learning new features. Symbolic EBL procedures have been demonstrated to acquire new features analytically, as a side-effect of extracting weakest preconditions (e.g., [Utgoff, 1986]). In contrast, Backpropagation neural net learning has been demonstrated to acquire useful new features inductively, in the hidden layer units of the network (e.g., [Rumelhart *et al.*, 1986]). In EBNN, both analytical and inductive components have an influence on the features that are acquired in the hidden layer of the target function. While we have not explored this issue, it seems an interesting opportunity for combining the analytical derivation of useful features based on prior knowledge with the statistical approaches associated with inductive neural network learning.

Difference in permanence of memory. Symbolic methods that learn rules update the target function by adding new rules which are usually assumed to be permanent. EBNN, however, has no corresponding notion of permanent memory. Instead, all new information, whether empirically or analytically derived, can at most cause an incremental change to the weights of the learned network. Furthermore, the influence of that new information will eventually be forgotten as it is incrementally overridden by subsequent data. This difference in memory permanence is primarily a result of the difference between bounded memory representations (e.g., fixed-size neural networks) and unbounded (e.g., arbitrarily large rule sets). If EBNN is used with a nearest neighbor (unbounded memory) representation of the target function, then the situation changes significantly. The difference in permanence of memory appears to have important implications for recovering from incorrect learning, for learning performance in cases where certain rare experiences are very important, and for maintaining consistency when the domain theory and target function are being simultaneously learned.

3.3 Differences in Capabilities

The above sections have described the EBL problems addressed by symbolic and neural network EBL methods, as well as the correspondence between the representations and algorithms they employ. These differences in algorithms have a number of implications for the relative capabilities of these two approaches:

Robustness to errors in domain theory and training data. One central motivation for developing the EBNN algorithm was to provide an explanation-based learning mechanism capable of using imperfect training data and domain theories which were themselves learned from scratch and of unknown reliability. As shown in the experimental results, EBNN is able to successfully learn from very weak or very strong domain theories. Its robustness follows from two characteristics: First, for each individual training example it uses both inductive and analytical methods for updating the target function. Second, it relies more strongly on the analytical learning component for examples that are explained accurately, and more strongly on the inductive component for poorly explained examples. While a variety of symbolic EBL methods have been developed that address imperfect domain theories none has yet been demonstrated to operate over the broad spectrum of learned domain theories of varying quality illustrated for EBNN.

Scaling issues. One well-known scaling problem associated with EBL (and other rule learning methods) is that the more rules learned, the longer it can require to match them to a new example. Because neural nets have a fixed response time regardless of how much they are trained, they have an advantage along this dimension. However, an offsetting issue has to do with the ability to represent arbitrarily complex functions. Because symbolic methods create collections of rules, each of which may cover some particular part of the domain, they can represent arbitrarily complex target functions by arbitrarily large rule sets. Here neural networks are at a disadvantage, since for any fixed-size network there will be complex functions that cannot be accurately represented. Put briefly, neural networks provide fixed response time but sacrifice fidelity in approximating the target function, whereas symbolic rules can provide better fidelity but sacrifice response time.

Ability to learn real-valued functions. Symbolic EBL is typically applied to problems where the target function can take on only a finite number of possible values. In part this is because a distinct learned rule is typically required for each observed output value of the target function. Neural net EBL offers the opportunity to apply explanation based methods to learning real-valued functions, such as the Q function learned in the robot control example. One potential advantage of learning real-valued functions is that it provides a way of applying EBL to learning about optimizing search (again, as illustrated by the robot learning example in which the goal is to maximize the robot's total reward). For example, consider the task of learning rules to control search. Symbolic EBL has been applied with great success to learning rules that recommend actions leading to a goal state, but has been remarkably unsuccessful at learning rules that lead along the minimum cost solution path. The reason is that explaining why one action has lower cost than another involves explanations regarding real-valued quantities, typically leading to very detailed explanations and very specific rules. The robot Q-Learning example above demonstrates that EBNN can be practically applied to learning such optimizing search control.

Human understandability. Symbolic rule-based representations are without doubt more understandable to humans than neural network representations. While there is work on extracting symbolic summaries of the knowledge implicit in neural network weights, an acceptable general-purpose method is still not available.

4 Summary and Conclusions

We have examined the correspondence between EBL methods using symbolic and neural network representations. The main points following from this discussion may be summarized as follows:

1. Explanation-based learning which has previously been considered a symbolic learning method can naturally be extended to neural network representations. The EBNN algorithm represents both the domain theory and the target function as neural networks, and successfully uses explanations of individual examples to guide generalization of the target function.
2. There is a qualitative difference between the information that can naturally be extracted from explanations based on symbolic rules and explanations based on neural networks: symbolic explanations yield *weakest preconditions* under which the explanation holds, whereas neural net explanations yield *partial derivatives of the target function* with respect to each of its inputs, evaluated at the training example point. These constitute two different ways of characterizing the relevance of training example attributes with respect to the target function.

3. There is a qualitative difference between the convergence properties stemming from neural net versus rule based representations for the target function. Neural net learning involves incrementally approximating the complete target function, whereas learning sets of rules involves incrementally extending the piecewise coverage of the target function's domain. This leads to significant differences in the cost of applying the learned function, the ability to learn very complex functions, the trustworthiness of the learned function before convergence, and the ease of interleaving learning of the domain theory and target function.

The analysis in this paper sheds some light on the tradeoffs between symbolic and neural net EBL methods. In particular, it shows that there are advantages to both: Neural net EBL offers a robust means of blending inductive and analytical learning, a natural method for learning real-valued functions that are especially relevant to learning optimal search control, and avoids the slow-down effect often associated with learning too many symbolic rules. Symbolic EBL offers advantages in scaling to more complex functions (due to its modular, multi-rule representation of the target function), in human comprehensibility, and in supporting first-order representations. Methods that combine the advantages of both these approaches are an excellent target for further research.

Acknowledgments

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. This research was sponsored in part by the Avionics Lab, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597, and by a grant from Fujitsu Laboratories Ltd.

References

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

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

[Kedar-Cabelli and McCarty, 1987] Smadar Kedar-Cabelli and Thorne McCarty. Explanation based learning as resolution theorem proving. In *Proceedings of the fourth international workshop on machine learning*, pages 383–389, Irvine, CA, 1987. Morgan Kaufmann.

[Laird *et al.*, 1988] John Laird, Paul Rosenbloom, and Allen Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1988.

[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.

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

[Mitchell and Thrun, 1993] 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.

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

[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.

[Thrun and Mitchell, 1993] Sebastian B. Thrun and Tom M. Mitchell. Integrating inductive neural network learning and explanation-based learning. *Proceedings of the IJCAI*, to appear, 1993.

[Utgoff, 1986] Paul E. Utgoff. *Machine Learning of Inductive Bias.* Kluwer Academic Publishers, 1986.

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