An Overview of Research at Wisconsin on Knowledge-Based Neural Networks*

Jude W. Shavlik
Department of Computer Sciences
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706 USA
shavlik@cs.wisc.edu

ABSTRACT

Recent research at the University of Wisconsin on knowledge-based neural networks is surveyed. This work has focused on (a) using symbolically represented background knowledge to improve neural-network learning and (b) extracting comprehensible symbolic representations from trained networks. Important open issues are discussed.

1. Introduction

We have been developing algorithms that integrate symbolic and neural-network approaches to supervised learning [15]. Our research program has had two main thrusts: using symbolically represented background knowledge to improve neural-network learning, and extracting comprehensible symbolic representations from trained networks. In this report, we summarize our research program since 1989, provide references to our more detailed articles, then close with a brief discussion of some important open issues.

2. Inserting Prior Knowledge into Neural Networks

Our research is focused on the machine-learning paradigm known as *supervised concept learning*, which can be summarized as follows: infer some unknown function $f(\vec{x})$ given a set of *instances* of f, where an instance is an input-output pair $\langle \vec{x}, f(\vec{x}) \rangle$. For many tasks, much is known about the function f in addition to the training instances. How to best use these additional sources of information is a central open question in machine learning. One major focus of our research has been to investigate how to make effective use of some types of prior knowledge about f.

The first significant results of our research program were the EBL-ANN[16] and KBANN [17, 19, 20] algorithms. KBANN enables a neural network to be initialized with a *domain theory* (i.e., a task-specific collection of inference rules), which need not be correct nor complete. It maps the domain theory into a neural network, determining its topology and initial weights. The set of inference rules that constitutes KBANN's input can be viewed as f', an approximation to f. The network created by KBANN initially computes f', and we use the training instances to refine the network, which leads to a better approximation of f. The refinement process simply runs the backpropagation algorithm on the network created by KBANN.

*Appeared in *Proc. of the Intl. Conf. on Neural Networks*, pp. 65–69, Washington, DC, 1996.

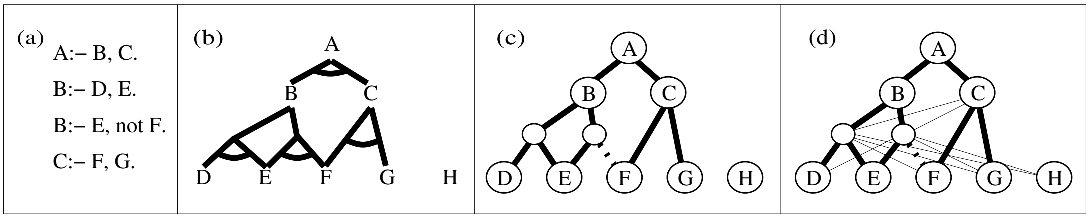

Fig. 1: Sample of the KBANN algorithm: (a) propositional rule set; (b) the rules viewed as an AND-OR dependency graph; (c) each proposition is represented as a unit (extra units are also added to represent disjunctive definitions, e.g., B), and their weights and biases are set so that they implement AND or OR gates, e.g, the weights B$\rightarrow$A and C$\rightarrow$A are set to 4 and the bias (threshold) of A to 6 (the bias of an OR node is 2); (d) low-weighted links are added between layers as a basis for future learning (e.g., an antecedent can be added to a rule by increasing one of these weights).

In one of KBANN's *knowledge-based neural networks*, the units represent Boolean concepts. A concept is assumed to be true if the unit representing the concept is highly active (near 1) and false if the unit is inactive (near 0). To represent the meaning of a set of rules, KBANN connects units with highly-weighted links and sets unit biases (thresholds) in such a manner that the (non-input) units emulate AND or OR gates, as appropriate. Figure 1 shows an example of the KBANN process for a set of simple propositional rules.

We have tested KBANN on a number of difficult, real-world tasks, including four DNA problems, the diagnosis of telephone-line faults, and a chess-like game [10, 13, 19]. KBANN was compared to other learning algorithms by measuring *generalization* performance. Generalization refers to how well a learning system is able to correctly classify previously unseen instances. In these empirical studies, the generalization performance of KBANN was superior to ordinary neural networks and to more than a half dozen other learning algorithms, and the differences were statistically significant.

Our initial KBANN algorithm required that existing knowledge be expressed in the form of non-recursive, propositional rules. Subsequent work extended the types of representations supported by KBANN. Our FSKBANN algorithm allows domain knowledge expressed as *generalized finite-state automata* to be mapped into recurrent neural networks [6]. This representation enabled us to consider state-dependent domain theories. We tested FSKBANN by using it to refine the Chou-Fasman algorithm for predicting how proteins fold. The resulting networks outperformed both the Chou-Fasman algorithm (a standard in the biological community), as well as previous neural-network approaches.

We also extended KBANN to handle *numeric* knowledge pertaining to the control of a chemical plant [14]. Again, the KBANN network outperformed the standard algorithm from the chemical engineering literature and an ordinary neural-network approach.

The basic KBANN algorithm determines the initial network topology, then only alters the network's weights. However, if the domain theory is sparse, the network KBANN produces can be too small [9, 10]. To overcome this limitation, we developed a heuristic-search technique that suggested where additional hidden units should be added to a KBANN-produced network [10, 12]. The basic approach is: train the initial KBANN-produced network, use a set of "tuning" examples to locate poorly performing hidden units, add additional hidden units to the weak portions of the network, then repeat as long as improvement is detected. We successfully extended this idea by using genetic operators (specialized versions of crossover and mutation) to create new networks from the current population of candidate network topologies [11]. Finally, we found it useful to create a *set* of knowledge-based networks whose outputs are averaged (weighted) when categorizing new examples [13]; we search for a set of highly accurate networks whose errors are not highly correlated.

Recently, we extended our approach to the reinforcement learning paradigm [5, 7, 8].[1] The approach we developed allows a connectionist reinforcement learner to accept advice, provided at any time during the learning process, by an external observer. In this approach, the advice-giver watches the learner and occasionally makes suggestions, expressed as instructions in a simple programming language. These instructions are then incorporated into the neural network and refined during learning. Effectively, this approach allows a domain theory to be provided piecemeal during learning. Our experiments demonstrated that this approach results in better performance from fewer training examples.

3. Understanding what a Neural Network Learned

The second major focus of our research program has been to develop algorithms for extracting comprehensible, symbolic knowledge from trained neural networks [1, 2, 3, 4, 18]. A current limitation of neural networks is that their concept representations are extremely difficult for humans to understand because they are represented using large numbers of real-valued parameters. The goal of this research is to enable comprehensible concept descriptions to be extracted from trained networks. Our network-extracted rules are more accurate than the rules produced by a decision-tree induction algorithm (called C4.5), and are comparable in terms of comprehensibility. We initially designed rule-extraction techniques tailored for knowledge-based networks [18] and, more recently, for "standard" neural networks [1, 2, 4].

The rule-extraction task can be summarized as follows:

Given: a trained network, the network's training set of labeled examples, and a desired level of fidelity between the extracted rule set and the network

Do: output a comprehensible set of symbolic rules that models the network to the specified degree of fidelity.

We initially developed the MofN algorithm [18] that extracts m-of-n rules from trained Kbann networks. An m-of-n rule is satisfied when at least m of its n antecedents are true. Our studies showed that we can extract comprehensible rules while still maintaining the accuracy of the trained network. Recall that networks created by Kbann are based on the rules that constitute a domain theory about the task at hand. This initial structuring greatly facilitates rule extraction, since Kbann essentially *refines* the rules rather than learning them from scratch.

Recently, we turned our attention to the problem of extracting comprehensible rules from ordinary neural networks. Because the MofN algorithm was designed to extract rules from knowledge-based networks, it makes several assumptions that are generally not valid for ordinary networks. For example, the MofN algorithm assumes that, after training, the weights of a network can be grouped into clusters, so that weights with similar magnitudes fall into the same cluster. This is a fair assumption for knowledge-based networks since their weights are initially clustered (because they implement a set of symbolic inference rules), and empirical results indicate that their weights remain clustered after training. We demonstrated that the MofN algorithm could

[1] In reinforcement learning, the learner is not provided with the correct answer for each training instance. Instead, the learning system periodically gets rewards (or penalties) from its environment and it must decide how to allocate credit/blame to previous actions that it took.

also be successfully applied to ordinary networks by using a training procedure that encouraged the weights to cluster *during* training [1]. This training procedure is a variant of Nowlan and Hinton's *soft weight-sharing* technique. Although their method was motivated by the desire for better generalization, we employed it to facilitate rule extraction.

We next developed a new rule-extraction approach that does not require a special training procedure [2]. It uses a rule-learning algorithm to find a set of rules that is approximately equivalent to the concept represented by a trained network. The *target concept*, in this learning task, is the function computed by the network. In addition to learning from training examples, our method exploits the property that networks can be efficiently queried.

Our most recent algorithm, TREPAN [4], uses queries to induce a decision tree that approximates the concept represented by a given network. Experiments demonstrate that TREPAN is able to produce decision trees that maintain a high level of fidelity to their respective networks, while being comprehensible and accurate. Unlike previous work in this area, the TREPAN algorithm is both general in its applicability and scalable to problems with high-dimensional input spaces.

4. Conclusion

The value of using prior, symbolic knowledge in combination with neural training methods has been demonstrated on a variety of tasks involving comparison to alternate learning methods. The set of knowledge representations mappable into networks is growing (see [15] for a review), though an open issue is the use of prior knowledge expressed in first-order predicate calculus, a topic addressed in the field of *inductive logic programming*.

While knowledge-based neural networks have been applied to several real-world tasks, these have generally involved small amounts of prior knowledge. It would be valuable to apply this approach to increasingly large domain theories and data sets, in order to demonstrate scalability and to determine the limitations of the current algorithms.

Another important future direction in knowledge-based networks is to allow the human user to provide symbolic knowledge at anytime during the network's lifetime. That is, the user should observe the network make its decisions, and whenever any potentially useful advice occurs to the user, he or she should simply be able to provide these "rules" to the network. The network should absorb this advice, refining it as necessary based on subsequent experience. There are several advantages to this view, including: (1) for large problems it may be infeasible to restart network training from "scratch" whenever the user thinks of some useful "prior" knowledge; (2) by observing the network's behavior the user may be more likely to think of good advice; (3) user-provided advice might be able to help a network get out of a local minima. In general, we should think of a continual dialog between the human user and the machine learner. Maclin's thesis work [5] provides an initial approach to this task.

Finally, in most domains it is important to understand what a trained network has learned. Rule-extraction algorithms produce concept descriptions that now rival the readability of induced decision trees on many problems, though often both approaches produce representations that opaquely describe the concept represented by a set of data. Producing more comprehensible concept descriptions is an important open issue that spans the full space of learning algorithms.

Acknowledgements

This report was partially supported by ONR Grant N00014-93-1-0998. Most of our papers and theses are available at `http://www.cs.wisc.edu/~shavlik/mlrg/publications.html`.

References

[1] M. Craven and J. Shavlik, "Learning symbolic rules using artificial neural networks," in *Proceedings of the Tenth International Conference on Machine Learning*, (Amherst, MA), pp. 73–80, Morgan Kaufmann, 1993.

[2] M. Craven and J. Shavlik, "Using sampling and queries to extract rules from trained neural networks," in *Proceedings of the Eleventh International Conference on Machine Learning*, (New Brunswick, NJ), pp. 37–45, 1994.

[3] M. Craven and J. Shavlik, "Extracting comprehensible concept representations from trained neural networks," in *Presented at the IJCAI Workshop on Comprehensibility in Machine Learning*, (Montreal, Quebec, Canada), 1995.

[4] M. Craven and J. Shavlik, "Extracting tree-structured representations of trained networks," in *Advances in Neural Information Processing Systems, Vol. 8*, 1996.

[5] R. Maclin, *Learning from Instruction and Experience: Methods for Incorporating Procedural Domain Theories into Knowledge-Based Neural Networks.* PhD thesis, Department of Computer Sciences, University of Wisconsin-Madison, 1995.

[6] R. Maclin and J. Shavlik, "Using knowledge-based neural networks to improve algorithms: Refining the Chou-Fasman algorithm for protein folding," *Machine Learning*, vol. 11, no. 2,3, pp. 195–215, 1993.

[7] R. Maclin and J. Shavlik, "Incorporating advice into agents that learn from reinforcements," in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, (Seattle, WA), pp. 694–699, AAAI/MIT Press, 1994.

[8] R. Maclin and J. Shavlik, "Creating advice-taking reinforcement learners," *Machine Learning*, vol. 22, pp. 251–281, 1995.

[9] D. Opitz, *An Anytime Approach to Connectionist Theory Refinement: Refining the Topologies of Knowledge-Based Neural Networks.* PhD thesis, Department of Computer Sciences, University of Wisconsin-Madison, 1995.

[10] D. Opitz and J. Shavlik, "Heuristically expanding knowledge-based neural networks," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, (Chambery, France), pp. 1360–1365, Morgan Kaufmann, September 1993.

[11] D. Opitz and J. Shavlik, "Using genetic search to refine knowledge-based neural networks," in *Proceedings of the Eleventh International Conference on Machine Learning*, (New Brunswick, NJ), pp. 208–216, Morgan Kaufmann, July 1994.

[12] D. Opitz and J. Shavlik, "Dynamically adding symbolically meaningful nodes to knowledge-based neural networks," *Knowledge-Based Systems*, pp. 301–311, 1996.

[13] D. Opitz and J. Shavlik, "Generating accurate and diverse members of a neural-network ensemble," in *Advances in Neural Information Processing Systems, Vol. 8*, 1996.

[14] G. Scott, J. Shavlik, and W. Ray, "Refining PID controllers using neural networks," *Neural Computation*, vol. 5, no. 4, pp. 746–757, 1992.

[15] J. Shavlik, "Combining symbolic and neural learning," *Machine Learning*, vol. 14, no. 2, pp. 321–331, 1994.

[16] J. Shavlik and G. Towell, "An approach to combining explanation-based and neural learning algorithms," *Connection Science*, vol. 1, no. 3, pp. 233–255, 1989.

[17] G. Towell, *Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction.* PhD thesis, Department of Computer Sciences, University of Wisconsin-Madison, 1991.

[18] G. Towell and J. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Machine Learning*, vol. 13, no. 1, pp. 71–101, 1993.

[19] G. Towell and J. Shavlik, "Knowledge-based artificial neural networks," *Artificial Intelligence*, vol. 70, no. 1,2, pp. 119–165, 1994.

[20] G. Towell, J. Shavlik, and M. Noordewier, "Refinement of approximate domain theories by knowledge-based neural networks," in *Proceedings of the Eighth National Conference on Artificial Intelligence*, (Boston, MA), pp. 861–866, AAAI/MIT Press, 1990.