

Data-Driven Theory Refinement Using KBDistAl

Jihoon Yang¹, Rajesh Parekh², Vasant Honavar³, and Drena Dobbs⁴

¹ HRL Laboratories, 3011 Malibu Canyon Rd, Malibu, CA 90265, USA
yang@wins.hrl.com

² Allstate Research & Planning Ctr, 321 Middlefield Rd, Menlo Park, CA 94025, USA
rpare@allstate.com

³ Computer Science Dept., Iowa State University, Ames, IA 50011-1040, USA
honavar@cs.iastate.edu

⁴ Zoology & Genetics Dept., Iowa State University, Ames, IA 50011-1040, USA
d_dobbs@molebio.iastate.edu

Abstract. Knowledge based artificial neural networks offer an attractive approach to extending or modifying incomplete knowledge bases or domain theories through a process of data-driven theory refinement. We present an efficient algorithm for data-driven knowledge discovery and theory refinement using DistAl, a novel (inter-pattern distance based, polynomial time) constructive neural network learning algorithm. The initial domain theory comprising of propositional rules is translated into a knowledge based network. The domain theory is modified using DistAl which adds new neurons to the existing network as needed to reduce classification errors associated with the incomplete domain theory on labeled training examples. The proposed algorithm is capable of handling patterns represented using binary, nominal, as well as numeric (real-valued) attributes. Results of experiments on several datasets for *financial advisor* and the *human genome project* indicate that the performance of the proposed algorithm compares quite favorably with other algorithms for connectionist theory refinement (including those that require substantially more computational resources) both in terms of generalization accuracy and network size.

1 Introduction

Inductive learning systems attempt to learn a concept description from a sequence of labeled examples [13,17,21]. Artificial neural networks, because of their massive parallelism and potential for fault and noise tolerance, offer an attractive approach to inductive learning [10,21,30]. Such systems have been successfully used for data-driven knowledge acquisition in several application domains. However, these systems generalize from the labeled examples alone. The availability of domain specific knowledge (domain theories) about the concept being learned can potentially enhance the performance of the inductive learning system [31]. Hybrid learning systems that effectively combine domain knowledge with the inductive learning can potentially learn faster and generalize better than those based on purely inductive learning (learning from labeled

examples alone). In practice the domain theory is often *incomplete* or even *inaccurate*.

Inductive learning systems that use information from training examples to modify an existing domain theory by either augmenting it with new knowledge or by refining the existing knowledge are called *theory refinement* systems.

Theory refinement systems can be broadly classified into the following categories.

- **Approaches based on Rule Induction** which use decision tree or rule learning algorithms for theory revision. Examples of such systems include RTLS [9], EITHER [24], PTR [16], and TGCI [3].
- **Approaches based on Inductive Logic Programming** which represent knowledge using first-order logic (or restricted subsets of it). Examples of such systems include FOCL [27] and FORTE [29].
- **Connectionist Approaches using Artificial Neural Networks** which typically operate by first embedding domain knowledge into an appropriate initial neural network topology and refine it by training the resulting neural network on the set of labeled examples. The KBANN system [34, 35] as well as related approaches [6] and [15] offer examples of this approach.

In experiments involving datasets from the Human Genome Project¹, KBANN has been reported to have outperformed symbolic theory refinement systems (such as EITHER) and other learning algorithms such as backpropagation and ID3 [34]. KBANN is limited by the fact that it does not modify the network's topology and theory refinement is conducted solely by updating the connection weights. This prevents the incorporation of new rules and also restricts the algorithm's ability to compensate for inaccuracies in the domain theory. Against this background, constructive neural network learning algorithms, because of their ability to modify the network architecture by dynamically adding neurons in a controlled fashion [14, 26, 37], offer an attractive connectionist approach to data-driven theory refinement. Available domain knowledge is incorporated into an initial network topology (e.g., using the rules-to-network algorithm of [35] or by other means). Inaccuracies in the domain theory are compensated for by extending the network topology using training examples. Figure 1 depicts this process.

Constructive neural network learning algorithms [14, 26, 37], that circumvent the need for a-priori specification of network architecture, can be used to construct networks whose size and complexity is commensurate with the complexity of the data, and trade off network complexity and training time against generalization accuracy. A variety of constructive learning algorithms have been studied in the literature [4, 8, 11, 14, 26, 37]. DistAl [37] is a polynomial time learning algorithm that is guaranteed to induce a network with zero classification error on any non-contradictory training set. It can handle pattern classification tasks

¹ These datasets are available at <ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/datasets/>.

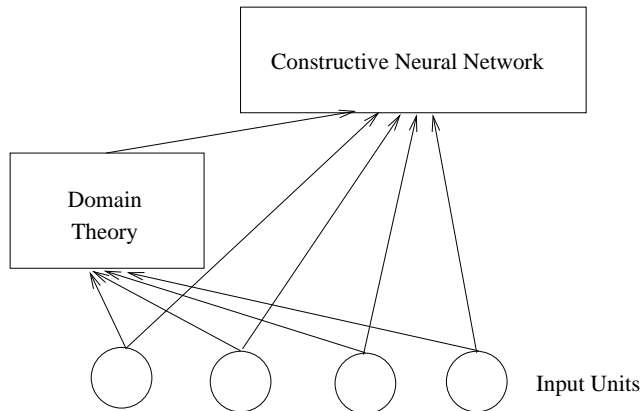


Fig. 1. Theory Refinement using a Constructive Neural Network

in which patterns are represented using binary, nominal, as well as numeric attributes. Experiments on a wide range of datasets indicate that the classification accuracies attained by `DistAl` are competitive with those of other algorithms [37, 38]. Since `DistAl` uses inter-pattern distance calculations, it can be easily extended to pattern classification problems wherein patterns are of variable sizes (e.g., strings or other complex symbolic structures) as long as suitable distance measures are defined [18]. Thus, `DistAl` is an attractive candidate for use in data-driven refinement of domain knowledge. Available domain knowledge is incorporated into an initial network topology. Inaccuracies in the domain theory are corrected by `DistAl` which adds additional neurons to eliminate classification errors on training examples.

Against this background, we present `KBDistAl`, a data-driven constructive theory refinement algorithm based on `DistAl`.

2 Constructive Theory Refinement Using Knowledge-Based Neural Networks

This section briefly describes several constructive theory refinement systems that have been studied in the literature.

Fletcher and Obradović [5] designed a constructive learning method for dynamically adding neurons to the initial knowledge based network. Their approach starts with an initial network representing the domain theory and modifies this theory by constructing a single hidden layer of threshold logic units (TLUs) from the labeled training data using the HDE algorithm [1]. The HDE algorithm divides the feature space with hyperplanes. Fletcher and Obradović’s algorithm maps these hyperplanes to a set of TLUs and then trains the output neuron using the pocket algorithm [8]. The `KBDistAl` algorithm proposed in this paper, like that of Fletcher and Obradović, also constructs a single hidden layer. However it differs in one important aspect: It uses a computationally efficient `DistAl`

algorithm which constructs the entire network in one pass through the training set instead of relying on the iterative approach used by Fletcher and Obradović which requires a large number of passes through the training set.

The RAPTURE system is designed to refine domain theories that contains probabilistic rules represented in the certainty-factor format [20]. RAPTURE's approach to modifying the network topology differs from that used in KBDistAl as follows: RAPTURE uses an iterative algorithm to train the weights and employs the information gain heuristic [28] to add links to the network. KBDistAl is simpler in that it uses a non-iterative constructive learning algorithm to augment the initial domain theory.

Opitz and Shavlik have extensively studied connectionist theory refinement systems that overcome the fixed topology limitation of the KBANN algorithm [22, 23]. The TopGen algorithm [22] uses a heuristic search through the space of possible expansions of a KBANN network constructed from the initial domain theory. TopGen maintains a queue of candidate networks ordered by their test accuracy on a cross-validation set. At each step, TopGen picks the best network and explores possible ways of expanding it. New networks are generated by strategically adding nodes at different locations within the best network selected. These networks are trained and inserted into the queue and the process is repeated.

The REGENT algorithm uses a genetic search to explore the space of network architectures [23]. It first creates a diverse initial population of networks from the KBANN network constructed from the domain theory. Genetic search uses the classification accuracy on a cross-validation set as a fitness measure. REGENT's mutation operator adds a node to the network using the TopGen algorithm. It also uses a specially designed crossover operator that maintains the network's rule structure. The population of networks is subjected to fitness proportionate selection, mutation, and crossover for many generations and the best network produced during the entire run is reported as the solution. KBDistAl is considerably simpler than both TopGen and REGENT. It constructs a single network in one pass through the training data as opposed to training and evaluating a population of networks using the computationally expensive backpropagation algorithm for several generations. Thus, it is significantly faster than TopGen and REGENT.

Parekh and Honavar [25] propose a constructive approach to theory refinement that uses a novel combination of the Tiling and Pyramid constructive learning algorithms [8, 26]. They use a symbolic knowledge encoding procedure to translate a domain theory into a set of propositional rules using a procedure that is based on the *rules-to-networks* algorithm of Towell and Shavlik [35] which is used in KBANN, TopGen, and REGENT. It yields a set of rules each of which has only one antecedent. The rule set is then mapped to an AND-OR graph which in turn is directly translated into a neural network. The Tiling-Pyramid algorithm uses an iterative perceptron style weight update algorithm for setting the weights and the Tiling algorithm to construct the first hidden layer (which maps binary or numeric input patterns into a binary representation at the hidden layer) and the Pyramid algorithm to add additional neurons if needed. While

Tiling-Pyramid is significantly faster than TopGen and REGENT, it is still slower than KBDistAl because of its reliance on iterative weight update procedures.

3 KBDistAl: A Data-Driven Theory Refinement Algorithm

This section briefly describes our approach to knowledge based theory refinement using DistAl.

3.1 DistAl: An Inter-Pattern Distance Based Constructive Neural Network Algorithm

DistAl [14,37,38] is a simple and relatively fast constructive neural network learning algorithm for pattern classification. The key idea behind DistAl is to add *hyperspherical* hidden neurons one at a time based on a greedy strategy which ensures that each hidden neuron that is added correctly classifies a maximal subset of training patterns belonging to a single class. Correctly classified examples can then be eliminated from further consideration. The process is repeated until the network correctly classifies the entire training set. When this happens, the training set becomes linearly separable in the transformed space defined by the hidden neurons. In fact, it is possible to set the weights on the hidden to output neuron connections without going through an iterative, time-consuming process. It is straightforward to show that DistAl is guaranteed to converge to 100% classification accuracy on any finite training set in time that is polynomial (more precisely, quadratic) in the number of training patterns [37]. Experiments reported in [37] show that DistAl, despite its simplicity, yields classifiers that compare quite favorably with those generated using more sophisticated (and substantially more computationally demanding) learning algorithms.

3.2 Incorporation of Prior Knowledge into DistAl

The current implementation of KBDistAl makes use of a very simple approach to the incorporation of prior knowledge into DistAl. First, the input patterns are classified using the rules. The resulting outputs (classification of the input pattern) are then augmented to the pattern, which is connected to the constructive neural network. This explains how DistAl is used for the constructive neural network in Figure 1 efficiently without requiring a conversion of rules into a neural network.

4 Experiments

This section reports results of experiments using KBDistAl on data-driven theory refinement for the financial advising problem used by Fletcher and Obradović [5], as well as the ribosome binding site and promoter site prediction used by Shavlik's group [22,23,31,34,35]:

– **Ribosome**

This data is from the Human Genome Project. It comprises of a domain theory and a set of labeled examples. The input is a short segment of DNA nucleotides, and the goal is to learn to predict whether the DNA segments contain a ribosome binding site. There are 17 rules in the domain theory, and 1880 examples in the dataset.

– **Promoters**

This data is also from the Human Genome Project, and consists of a domain theory and a set of labeled examples. The input is a short segment of DNA nucleotides, and the goal is to learn to predict whether the DNA segments contain a promoter site. There are 31 rules in the domain theory, and 940 examples in the dataset.

– **financial advisor**

The financial advisor rule base contains 9 rules as shown in Figure 2 [19]. As in [5], a set of 5500 labeled examples that are consistent with the rule base is randomly generated. 500 examples are used for training and the remaining 5000 is used for testing.

1	if (sav_adeq and inc_adeq) then invest_stocks
2	if dep_sav_adeq then sav_adeq
3	if assets_hi then sav_adeq
4	if (dep_inc_adeq and earn_steady) then inc_adeq
5	if debt_lo then inc_adeq
6	if (sav \geq dep * 5000) then dep_sav_adeq
7	if (assets \geq income * 10) then assets_hi
8	if (income \geq 25000 + dep * 4000) then dep_inc_adeq
9	if (debt_pmt < income * 0.3) then debt_lo

Fig. 2. Financial advisor rule base.

4.1 Human Genome Project Datasets

The reported results are based on a 10-fold cross-validation. The average training and test accuracies of the rules in domain theory alone were 87.29 ± 0.22 and 87.29 ± 2.03 for **Ribosome** dataset and 77.45 ± 0.56 and 77.45 ± 5.01 for **Promoters** dataset, respectively. Table 1 and 2 shows the average generalization accuracy and the average network size (along with the standard deviations² where available) for **Ribosome** and **Promoters** datasets, respectively.

Table 1 and 2 compare the performance of **KBDistAl** with that of some of the other approaches that have been reported in the literature. For **Ribosome** dataset, it produced a lower generalization accuracy than the other approaches

² The standard error can be computed instead, for better interpretation of the results.

Table 1. Results of **Ribosome** dataset.

	<i>Test %</i>	<i>Size</i>
Rules alone	87.3 ± 2.0	—
KBDistAl (no pruning)	86.3 ± 2.4	40.3 ± 1.3
KBDistAl (with pruning)	91.8 ± 1.8	16.2 ± 3.7
Tiling-Pyramid	90.3 ± 1.8	23 ± 0.0
TopGen	90.9	42.1 ± 9.3
REGENT	91.8	70.1 ± 25.1

Table 2. Results of **Promoters** dataset.

	<i>Test %</i>	<i>Size</i>
Rules alone	77.5 ± 5.0	—
KBDistAl (no pruning)	93.0 ± 2.8	12.2 ± 1.0
KBDistAl (with pruning)	95.5 ± 3.3	3.9 ± 2.3
Tiling-Pyramid	96.3 ± 1.8	34 ± 0.0
TopGen	94.8	40.2 ± 3.3
REGENT	95.8	74.9 ± 38.9

and generated networks that were larger than those obtained by Tiling-Pyramid. We believe that this might have been due to overfitting. In fact, when the network pruning procedure was applied, the generalization accuracy increased to 91.8 ± 1.8 with smaller network size of 16.2 ± 3.7 . In the case of the **Promoters** dataset, KBDistAl produced comparable generalization accuracy with smaller network size. As in **Ribosome**, network pruning boosted the generalization accuracy to 95.5 ± 3.3 with significantly smaller network size of 3.9 ± 2.3 .

The time taken in our approach is significantly less than that of the other approaches. KBDistAl takes fraction of a minute to a few minutes of CPU time on each dataset used in the experiments. In contrast, TopGen and REGENT were reported to have taken several days to obtain the results reported in [23].

4.2 Financial Advisor Rule Base

As explained earlier, 5500 patterns were generated randomly to satisfy the rules in Figure 2, of which 500 patterns were used for training and the remaining 5000 patterns were used for testing the network. In order to experiment with several different incomplete domain theories, some of the rules were pruned with its antecedents in each experiment. For instance, if *sav_adeq* was selected as the pruning point, then the rules for *sav_adeq*, *dep_sav_adeq*, and *assets_hi* are eliminated from the rule base. In other words rules 2, 3, 6, and 7 are pruned. Further, rule 1 is modified to read “if (*inc_adeq*) then *invest_stocks*”. Then the initial network is constructed from this modified rule base and augmented using constructive learning.

Our experiments follow those performed in [5] and [25]. As we can see in Table 3 and 4, KBDistAl either outperformed the other approaches or gave compa-

rable results. It resulted in higher classification accuracy than other approaches in several cases, and it always produced fairly compact networks while using substantially lower amount of computational resources. Again, as in the Human Genome Project datasets, network pruning boosted the generalization in all cases with smaller network size. For the pruning points in Table 4 (the sequence from *dep_sav_adeq* to *inc_adeq*), the generalization accuracy improved to 89.2, 99.5, 98.4, 92.9, 94.9 and 93.0 with network sizes of 17, 2, 5, 9, 5 and 12, respectively.

Table 3. Results of financial advisor rule base (HDE).

Pruning point	HDE		Rules alone
	Test %	Hidden Units	Test %
dep_sav_adeq	92.7	31	75.1
assets_hi	92.4	23	93.4
dep_inc_adeq	85.8	25	84.5
debt_lo	84.7	30	61.7
sav_adeq	92.2	19	90.9
inc_adeq	81.2	32	64.6

Table 4. Results of financial advisor rule base (KBDistAl and Tiling-Pyramid).

Pruning point	KBDistAl		Tiling-Pyramid		Rules alone
	Test %	Size	Test %	Size	Test %
dep_sav_adeq	88.5	21	91.2 ± 1.7	28.2 ± 3.6	52.4
assets_hi	99.5	2	99.4 ± 0.2	10 ± 0.0	99.5
dep_inc_adeq	98.0	8	94.3 ± 1.5	21.0 ± 3.1	90.4
debt_lo	91.6	16	94.1 ± 2.0	22.1 ± 4.0	81.2
sav_adeq	93.8	10	90.8 ± 1.5	26.4 ± 3.3	87.6
inc_adeq	91.2	18	83.8 ± 2.2	32.7 ± 2.9	67.4

5 Summary and Discussion

Theory refinement techniques offer an attractive approach to exploiting available domain knowledge to enhance the performance of data-driven knowledge acquisition systems. Neural networks have been used extensively in theory refinement systems that have been proposed in the literature. Most of such systems translate the domain theory into an initial neural network architecture and then train the network to refine the theory. The KBANN algorithm is demonstrated to outperform several other learning algorithms on some domains [34,35]. However, a significant disadvantage of KBANN is its fixed network topology. TopGen and

REGENT algorithms on the other hand allow modifications to the network architecture. Experimental results have demonstrated that **TopGen** and **REGENT** outperform **KBANN** on several applications. [22,23]. The **Tiling-Pyramid** algorithm proposed in [25] for constructive theory refinement builds a network of perceptrons. Its performance, in terms of classification accuracies attained, as reported in [25], is comparable to that of **REGENT** and **TopGen**, but at significantly lower computational cost.

The implementation of **KBDistAl** used in the experiments reported in this paper uses the rules directly (by augmenting the input patterns with the outputs obtained from the rules) as opposed to the more common approach of incorporating the rules into an initial network topology. The use of **DistAl** for network construction makes **KBDistAl** significantly faster than approaches that rely on iterative weight update procedures (e.g., perceptron learning, backpropagation algorithm) and/or computationally expensive genetic search. Experimental results demonstrate that **KBDistAl**'s performance in terms of generalization accuracy is competitive with that of several of the more computationally expensive algorithms for data-driven theory refinement. Additional experiments using real-world data and domain knowledge are needed to explore the capabilities and limitations of **KBDistAl** and related algorithms for theory refinement. We conclude with a brief discussion of some promising directions for further research.

It can be argued that **KBDistAl** is not a *theory refinement* system in a strict sense. It makes use of the domain knowledge in its inductive learning procedure rather than *refining* the knowledge. Perhaps **KBDistAl** is more accurately described as a *knowledge guided* inductive theory construction system.

There are several extensions and variants of **KBDistAl** that are worth exploring. Given the fact that **DistAl** relies on inter-pattern distances to induce classifiers from data, it is straightforward to extend it so as to handle a much broader class of problems including those that involve patterns of variable sizes (e.g., strings) or symbolic structures as long as suitable inter-pattern distance metrics can be defined. Some steps toward rigorous definitions of distance metrics based on information theory are outlined in [18]. Variants of **DistAl** and **KBDistAl** that utilize such distance metrics are currently under investigation.

Several authors have investigated approaches to rule extraction from neural networks in general, and connectionist theory refinement systems in particular [2,7,33]. One goal of such work is to represent the learned knowledge in a form that is comprehensible to humans. In this context, rule extraction from classifiers induced by **KBDistAl** is of some interest.

In several practical applications of interest, all of the data needed for synthesizing reasonably precise classifiers is not available at once. This calls for incremental algorithms that continually refine knowledge as more and more data becomes available. Computational efficiency considerations argue for the use of data-driven theory refinement systems as opposed to storing large volumes of data and rebuilding the entire classifier from scratch as new data becomes available. Some preliminary steps in this direction are described in [12].

A somewhat related problem is that of knowledge discovery from large, physically distributed, dynamic data sources in a networked environment (e.g., data in genome databases). Given the large volumes of data involved, this argues for the use of data-driven theory refinement algorithms embedded in mobile software agents [12, 36] that travel from one data source to another, carrying with them only the current knowledge base as opposed to approaches rely on shipping large volumes of data to a centralized repository where knowledge acquisition is performed. Thus, data-driven knowledge refinement algorithms constitute one of the key components of distributed knowledge network [12] environments for knowledge discovery in many practical applications (e.g., bioinformatics).

In several application domains, knowledge acquired on one task can often be utilized to accelerate knowledge acquisition on related tasks. Data-driven theory refinement is particularly attractive in applications that lend themselves to such cumulative multi-task learning [32]. The use of KBDistAl or similar algorithms in such scenarios remains to be explored.

Acknowledgements

This research was partially supported by grants from the National Science Foundation (IRI-9409580) and the John Deere Foundation to Vasant Honavar and a grant from the Carver Foundation to Drena Dobbs and Vasant Honavar.

References

1. Baum, E., and Lang, K. 1991. Constructing hidden units using examples and queries. In Lippmann, R.; Moody, J.; and Touretzky, D., eds., *Advances in Neural Information Processing Systems, vol. 3*, 904–910. San Mateo, CA: Morgan Kaufmann.
2. Craven, M. 1996. *Extracting Comprehensible Models from Trained Neural Networks*. Ph.D. Dissertation, Department of Computer Science, University of Wisconsin, Madison, WI.
3. Donoho, S., and Rendell, L. 1995. Representing and restructuring domain theories: A constructive induction approach. *Journal of Artificial Intelligence Research* 2:411–446.
4. Fahlman, S., and Lebiere, C. 1990. The cascade correlation learning algorithm. In Touretzky, D., ed., *Neural Information Systems 2*. Morgan-Kaufman. 524–532.
5. Fletcher, J., and Obradović, Z. 1993. Combining prior symbolic knowledge and constructive neural network learning. *Connection Science* 5(3,4):365–375.
6. Fu, L. M. 1989. Integration of neural heuristics into knowledge-based inference. *Connection Science* 1:325–340.
7. Fu, L. M. 1993. Knowledge based connectionism for refining domain theories. *IEEE Transactions on Systems, Man, and Cybernetics* 23(1).
8. Gallant, S. 1990. Perceptron based learning algorithms. *IEEE Transactions on Neural Networks* 1(2):179–191.
9. Ginsberg, A. 1990. Theory reduction, theory revision, and retranslation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 777–782. Boston, MA: AAAI/MIT Press.

10. Hassoun, M. 1995. *Fundamentals of Artificial Neural Networks*. Boston, MA: MIT Press.
11. Honavar, V., and Uhr, L. 1993. Generative learning structures for generalized connectionist networks. *Information Sciences* 70(1-2):75–108.
12. Honavar, V.; Miller, L.; and Wong, J. 1998. Distributed knowledge networks. In *IEEE Information Technology Conference*.
13. Honavar, V. 1999a. Machine learning: Principles and applications. In Webster, J., ed., *Encyclopedia of Electrical and Electronics Engineering*. New York: Wiley. To appear.
14. Honavar, V. 1999b. Structural learning. In Webster, J., ed., *Encyclopedia of Electrical and Electronics Engineering*. New York: Wiley. To appear.
15. Katz, B. F. 1989. EBL and SBL: A neural network synthesis. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, 683–689.
16. Kopel, M.; Feldman, R.; and Serge, A. 1994. Bias-driven revision of logical domain theories. *Journal of Artificial Intelligence Research* 1:159–208.
17. Langley, P. 1995. *Elements of Machine Learning*. Palo Alto, CA: Morgan Kaufmann.
18. Lin, D. 1998. An information-theoretic definition of similarity. In *International Conference on Machine Learning*.
19. Luger, G. F., and Stubblefield, W. A. 1989. *Artificial Intelligence and the Design of Expert Systems*. Redwood City, CA: Benjamin/Cummings.
20. Mahoney, J., and Mooney, R. 1994. Comparing methods for refining certainty-factor rule-bases. In *Proceedings of the Eleventh International Conference on Machine Learning*, 173–180.
21. Mitchell, T. 1997. *Machine Learning*. New York: McGraw Hill.
22. Opitz, D. W., and Shavlik, J. W. 1995. Dynamically adding symbolically meaningful nodes to knowledge-based neural networks. *Knowledge-Based Systems* 8(6):301–311.
23. Opitz, D. W., and Shavlik, J. W. 1997. Connectionist theory refinement: Genetically searching the space of network topologies. *Journal of Artificial Intelligence Research* 6:177–209.
24. Ourston, D., and Mooney, R. J. 1994. Theory refinement: Combining analytical and empirical methods. *Artificial Intelligence* 66:273–310.
25. Parekh, R., and Honavar, V. 1998. Constructive theory refinement in knowledge based neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, 2318–2323.
26. Parekh, R.; Yang, J.; and Honavar, V. 1997. Constructive neural network learning algorithms for multi-category real-valued pattern classification. Technical Report ISU-CS-TR97-06, Department of Computer Science, Iowa State University.
27. Pazzani, M., and Kibler, D. 1992. The utility of knowledge in inductive learning. *Machine Learning* 9:57–94.
28. Quinlan, R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.
29. Richards, B., and Mooney, R. 1995. Automated refinement of first-order horn-clause domain theories. *Machine Learning* 19:95–131.
30. Ripley, B. 1996. *Pattern Recognition and Neural Networks*. New York: Cambridge University Press.
31. Shavlik, J. W. 1994. A framework for combining symbolic and neural learning. In *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. Boston: Academic Press.
32. Thrun, S. 1995. Lifelong learning: A case study. Technical Report CMU-CS-95-208, Carnegie Mellon University.

33. Towell, G., and Shavlik, J. 1993. Extracting rules from knowledge-based neural networks. *Machine Learning* 13:71–101.
34. Towell, G., and Shavlik, J. 1994. Knowledge-based artificial neural networks. *Artificial Intelligence* 70(1–2):119–165.
35. Towell, G.; Shavlik, J.; and Noordwier, M. 1990. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 861–866.
36. White, J. 1997. Mobile agents. In Bradshaw, J., ed., *Software Agents*. Cambridge, MA: MIT Press.
37. Yang, J.; Parekh, R.; and Honavar, V. 1998. DistAl: An inter-pattern distance-based constructive learning algorithm. In *Proceedings of the International Joint Conference on Neural Networks*, 2208–2213.
38. Yang, J.; Parekh, R.; and Honavar, V. 1999. DistAl: An inter-pattern distance-based constructive learning algorithm. *Intelligent Data Analysis*. To appear.