# COMBINING PRIOR SYMBOLIC KNOWLEDGE AND CONSTRUCTIVE NEURAL NETWORK LEARNING

Justin Fletcher Zoran Obradović
jfletche@eecs.wsu.edu zoran@eecs.wsu.edu
School of Electrical Engineering and Computer Science
Washington State University, Pullman WA 99164-2752

**Abstract**

The concepts of knowledge-based systems and machine learning are combined by integrating an expert system and a constructive neural networks learning algorithm. Two approaches are explored: embedding the expert system directly and converting the expert system rule base into a neural network. This initial system is then extended by constructively learning additional hidden units in a problem-specific manner. Experiments performed indicate that generalization of a combined system surpasses that of each system individually.

Contact:

Dr. Zoran Obradović
zoran@eecs.wsu.edu
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752
(509) 335-6601
FAX: (509) 335-3818

# COMBINING PRIOR SYMBOLIC KNOWLEDGE AND CONSTRUCTIVE NEURAL NETWORK LEARNING*

Justin Fletcher Zoran Obradović†
jfletche@eecs.wsu.edu zoran@eecs.wsu.edu

School of Electrical Engineering and Computer Science
Washington State University, Pullman WA 99164-2752

**Abstract**

The concepts of knowledge-based systems and machine learning are combined by integrating an expert system and a constructive neural networks learning algorithm. Two approaches are explored: embedding the expert system directly and converting the expert system rule base into a neural network. This initial system is then extended by constructively learning additional hidden units in a problem-specific manner. Experiments performed indicate that generalization of a combined system surpasses that of each system individually.

*Research sponsored in part by the NSF Industry / University Cooperative Center for the Design of Analog-Digital ASICs (CDADIC) under grant NSF-CDADIC-90-1 and by Washington State University Research Grant 10C-3970-9966.

†Also affiliated with the Mathematical Institute, Belgrade, Yugoslavia.

# COMBINING PRIOR SYMBOLIC KNOWLEDGE AND CONSTRUCTIVE NEURAL NETWORK LEARNING[‡]

Justin Fletcher Zoran Obradović[§]
jfletche@eecs.wsu.edu zoran@eecs.wsu.edu

School of Electrical Engineering and Computer Science
Washington State University, Pullman WA 99164-2752

**Abstract**

The concepts of knowledge-based systems and machine learning are combined by integrating an expert system and a constructive neural networks learning algorithm. Two approaches are explored: embedding the expert system directly and converting the expert system rule base into a neural network. This initial system is then extended by constructively learning additional hidden units in a problem-specific manner. Experiments performed indicate that generalization of a combined system surpasses that of each system individually.

# 1 Introduction

Classification systems in artificial intelligence are primarily based on two concepts. The first is that of representing existing human knowledge in a form that can be interpreted by a machine. In particular, knowledge in an expert system is represented by the rule base extracted from a domain expert [9]. The second concept is the extraction of the knowledge from the sample data which is the subject of machine learning research [18]. The knowledge represented by a rule base may be either incomplete or inconsistent requiring techniques such as the conflict resolution heuristics implemented by the inference engine of the expert system. In a machine learning system, the knowledge is usually incomplete as an incomplete set of examples is practically available. The examples themselves may also be noisy or conflicting, leading to incomplete knowledge. Thus, the knowledge retained by each concept may be incomplete or inconsistent. The goal of this research is to integrate the two concepts in such a way that the knowledge from each representation is used.

[‡]Research sponsored in part by the NSF Industry / University Cooperative Center for the Design of Analog-Digital ASICs (CDADIC) under grant NSF-CDADIC-90-1 and by Washington State University Research Grant 10C-3970-9966.

[§]Also affiliated with the Mathematical Institute, Belgrade, Yugoslavia.

The concepts are combined by embedding an expert system into a feed-forward artificial neural network. Two approaches are explored. In the first, the expert system is directly embedded into the neural network. In the second, the expert system rule base is first converted into a neural network. In both approaches the neural network is then extended by constructively adding hidden units to represent additional knowledge obtained from the sample data. By integrating both concepts, we obtain a system in which pre-existing knowledge is complemented by the knowledge obtained from the sample data.

In section 2, we propose an algorithm for learning separating hyperplanes from sample data and constructively creating corresponding neural network hidden units. Section 3 combines this new algorithm with existing methods for representing prior symbolic knowledge. In section 4, the generalization abilities of each concept are studied separately and as well as combined performing experiments using the MONK's problems [16].

# 2 A Constructive Neural Network Learning Algorithm

A neural network is a highly parallel machine consisting of computational units based originally upon the simple binary model of a neuron of McCulloch and Pitts [11]. Since the development of the backpropagation algorithm by Rumelhart [15] (among others), there has been increased interest in their usage for applications where traditional computation has performed poorly (e.g., ambiguous data or large contextual influence).

In selecting an appropriate neural network topology for a classification problem, there are two opposing objectives. The network must be large enough to represent the problem and should be small enough in order to generalize well. In contrast to learning on a prespecified topology, a constructive algorithm also learns the topology in a manner specific to the problem. Examples include the tiling algorithm of Mézard and Nadal [12] and the cascade-correlation algorithm of Fahlman and Lebiere [5].

An iterative construction of hidden units in a feed-forward neural network with a single hidden layer is proposed by Baum [2]. This algorithm constructs a two layer neural network in polynomial time given examples and the ability to query for the classification of specific points within the problem domain. In a neural network, a hidden unit with fan-in $k$ is a representation of a $k$-1 dimensional hyperplane. Baum's method uses a sequence of oracle queries in conjunction with training examples to find the points on

each of the separating hyperplanes between different classes. The hyperplane is then determined by solution of the equation system defined by $k$ points on the hyperplane. In practice, the oracle required by Baum's algorithm may either be too expensive or not available. Here we propose a modified algorithm to construct the hidden units from examples alone (preliminary results appeared in [6]).

In our algorithm, an approximation of the points on the separating hyperplane are found by repeatedly interpolating between points of the different classes in the training set $T$. The interpolation begins by selecting positive and negative examples $m, n \in T$. The unknown region between $m$ and $n$ is then searched for the closest point $q \in T$ to the midpoint of $m$ and $n$. The unknown region is defined as the intersection of the circles with centers $m$ and $n$ and a radius of the distance between $m$ and $n$, as shown in Figure 1. If $q$ is found, the search is then repeated in the smaller unknown region between $q$ and $m$ or $q$ and $n$ respectively depending on whether $q$ is positive or negative (Figure 2).

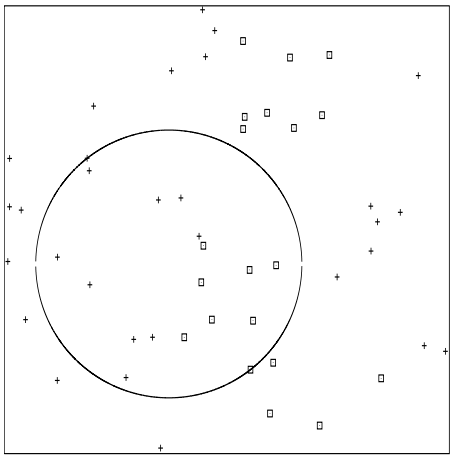

Figure 1: First unknown region

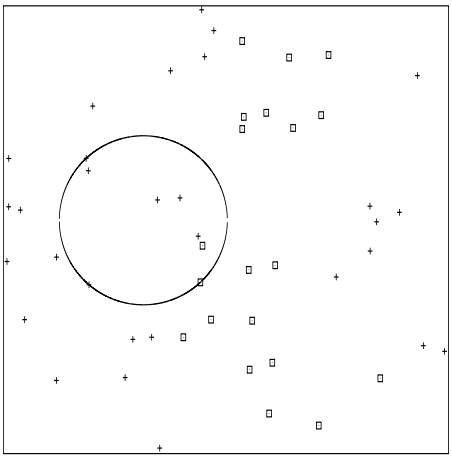

Figure 2: Next unknown region

If no point from $T$ is found in the current unknown region, its midpoint $p^1$ is the closest approximation to a point on the separating hyperplane. A new pair of starting points is selected and the search is repeated if $p^1$ is determined to be within a specified tolerance of an existing hyperplane. The remaining points $p^2$ through $p^k$ on the separating hyperplane are found by taking a random vector from $p^1$ to a point $v \in T$ (Figure 3) and interpolating between either $m$ and $v$ or $n$ and $v$ to $p^i$ based on the class of $v$. If distinct $k$ points on the hyperplane cannot be determined after a pre-specified number of

trials are made, the current hyperplane search is abandoned and a new pair of starting points is then chosen to begin another hyperplane search. The generated hyperplane is shown in Figure 4.

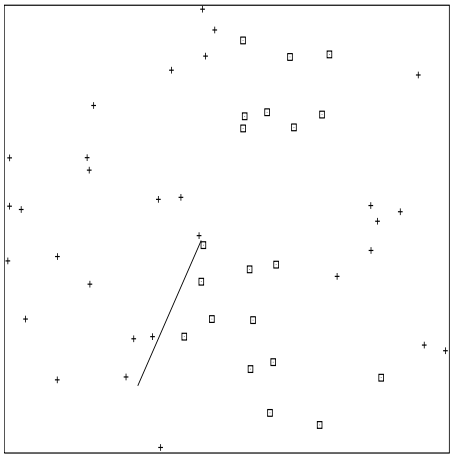

Figure 3: Random vector

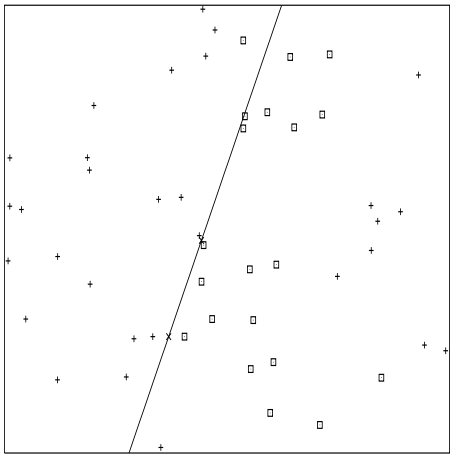

Figure 4: Separating hyperplane

The hyperplanes are determined from examples alone rather than from oracle queries. These hyperplanes may not correspond to an optimal separation of the training classes. To compensate for this we search for an additional point on the hyperplane. A more accurate separating hyperplane may then be constructed by averaging the $k+1$ hyperplanes determined from $k$ of the $k+1$ points.

The hidden layer representation of the generated network with the same number of hidden units as in the minimal network for a given problem may not be linearly separable. In order to account for this possibility, hidden units continue to be generated beyond the minimal architecture; for example, until the data is exhausted, a number of data points have been examined without generating a new hidden unit or a predetermined number of units have been created. As in Baum's original algorithm, the connection weights from the hidden units to the output layer must be learned once the hidden unit layer has been generated.

In generated networks there are no hidden connections to learn and so one can use any number of algorithms (including backpropagation) to train the output layer weights. A natural choice is the pocket algorithm [7], which is a single-layer neural network algorithm that finds the optimal separation even for non-linearly separable problems. The algorithm keeps the best set of weights in the "pocket" while the perceptron is trained

incrementally. A practical modification of the pocket learning algorithm is proposed in [13] which is faster and still has the same guarantee for convergence to the optimal separating hyperplane. We use this parallel dynamic algorithm to determine the output layer weights in the constructed network. Computational and generalization abilities of this algorithm are discussed in section 4.

# 3 Embedding of Prior Symbolic Knowledge

In artificial intelligence, prior knowledge is often represented as a set of rules to be interpreted by an expert system. This rule base represents the knowledge of skilled professionals as interpreted by knowledge engineers. The knowledge may be incomplete, contradictory, be miscommunicated by the expert or be misinterpreted by the knowledge engineer. Such a system is also static. Without modification to the rule base, it cannot learn from the sample data presented.

As a sample expert system we used the C Language Integrated Production System (CLIPS) developed by the Software Technology Branch of NASA at the Lyndon B. Johnson Space Center [8]. Through the use of CLIPS, it was possible to embed an expert system as a starting point for the iterative construction of a neural network. As shown in Figure 5, the prior knowledge from the rule base is extended by incrementally adding additional hidden units as necessary.

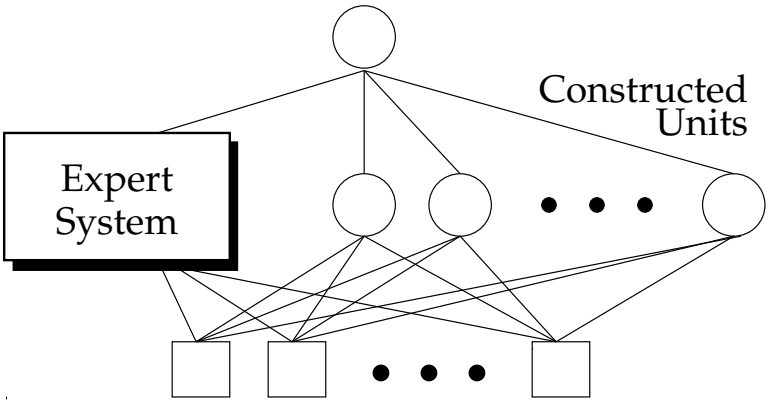


Figure 5: Integrated system

An alternate approach is to convert the rule base into an explicit neural network and use this network as a starting point for incremental learning. The knowledge-based

neural network algorithms assume the existence of approximate pre-existing symbolic knowledge which is used to construct the initial hypothesis (network topology and initial set of weights). The learning algorithm described in the Section 2 is then used to expand the hypothesis.

KBANN, a method of generating knowledge-based artificial neural networks from hierarchically-structured rules, was developed by Towell et. al. [17]. Although KBANN performs well when much of the domain theory is known, it is restricted to refining the existing rules rather than discovering new rules. The principal reason is that the network's representational power is bounded by a fixed topology determined only by pre-existing set of rules.

In our hybrid system the domain theory is transformed into an initial network through an extended version of KBANN's symbolic knowledge encoding. As an example, consider the rule base in Table 1 which is a modified version of the simple financial advisor from [10].

| | | |
|---|---|---|
| if (savings_adequate and income_adequate) | then | invest_stocks |
| if dependent_savings_adequate | then | savings_adequate |
| if assets_high | then | savings_adequate |
| if (dependent_income_adequate and earnings_steady) | then | income_adequate |
| if debt_low | then | income_adequate |
| if (savings $\geq$ dependents $\times$ 5000) | then | dependent_savings_adequate |
| if (income $\geq$ 25000 + dependents $\times$4000) | then | dependent_income_adequate |
| if (assets $\geq$ income $\times$ 10) | then | assets_high |
| if (debt_payments $<$ income $\times$ 0.30) | then | debt_low |

Table 1: Financial advisor rule base

The KBANN algorithm is able to handle the first five rules of Table 1, which is represented in and/or graph form in Figure 6.

The network topology corresponding to the and/or graph (rules 1–5) is created following the KBANN transformation. The remaining hidden layer units are created by extending KBANN to handle rules 6–9 from Table 1. In the constructed network, all units compute binary threshold functions. Figure 7 represents the generated network with connection weights and thresholds. In our algorithm KBANN's additional steps of fully connecting the constructed network layers and training via backpropagation are not performed. These steps, which compensate for an incomplete or contradictory rule base, are replaced by incremental hidden unit generation. We treat the constructed network as the expert system of Figure 5 and add constructed units accordingly.

As KBANN is restricted by a fixed network topology it refines the domain theory

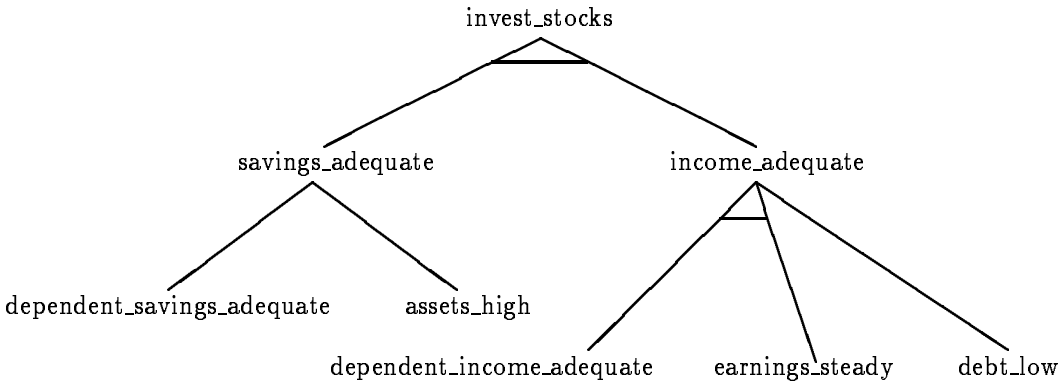


Figure 6: Rule base and/or graph

rather than extending it. Additional knowledge not anticipated in the pre-existing rule base but contained in the examples may not be representable by the existing architecture. Combining prior symbol knowledge with the constructive approach of Section 2, there is no restriction on the representational ability of the separating surface since it is defined by the units constructed from the sample data.

# 4 Results

The total running time of our hybrid algorithm depends on the time to embed prior symbolic knowledge, the time required to construct neural network hidden units as well as on the time required to learn the output layer weights. As the major cost is learning the hidden layer, we examine the time required to determine if a separating hyperplane can be determined from a given pair of points. Search for a point on the hyperplane takes $O(\log N)$ interpolation steps since each interpolation removes at least half of the $N$ training examples. $O(\log N)$ interpolations are required to search for a point on the hyperplane. In each interpolation step, finding the closest point to the midpoint can be determined in time $O(\log N)$ through use of the $k$-$d$ tree of Bentley [3]. Thus, the time required to search for one point on the hyperplane is $O(\log^2 N)$. A hyperplane is defined by $k$ points, and so the time to determine if a hyperplane can be found from a given pair of starting points is $O(k \log^2 N)$. With $m$ and $n$ points of each class, an exhaustive data partitioning is performed in time $O(mnk \log^2 N)$.

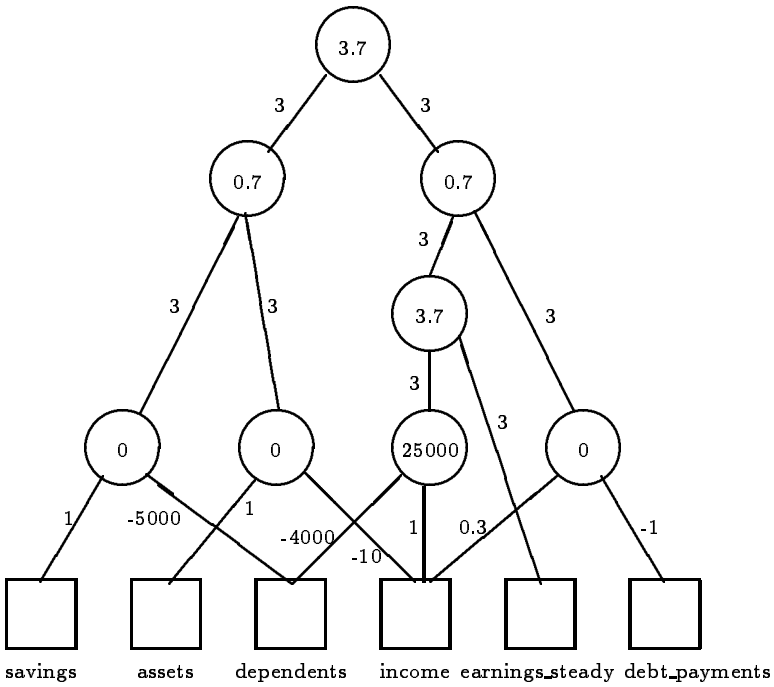


Figure 7: Initial network

Three hundred trials of our algorithm were performed using randomly generated networks. Each random network had two input units and averaged 6.28 units in the hidden layer. The constructed network averaged 9.47 units in its hidden layer. Using one hundred training points and one thousand test points, an average accuracy of 99.45% was achieved on the training set with 95.75% accuracy on the test set.

In order to evaluate our combined system generalization abilities, experiments were performed using the financial advisor previously described and the MONK's problems [16]. Integration with KBANN was tested by performing experiments based on the financial advisor rule base presented earlier. For purposes of comparison, three previously described techniques were used for network construction using varying amounts of prior knowledge: extended KBANN using rules alone, extended KBANN with hidden unit generation from examples and queries and extended KBANN with hidden unit generation from examples alone. In order to evaluate the effect of an incomplete knowledge base, rules and their antecedents were pruned to reduce prior knowledge. For example, elimination of the savings_adequate rule and its antecedents dependent_savings_adequate and

assets_high would be indicated by a prior knowledge pruning point of savings_adequate. Average results of five tests performed using 500 training and 5000 test examples randomly generated consistent with the full rule base are show in Table 2.

| Prior Knowledge Pruning Point | Algorithm | Hidden Units Constructed | Test Data Success Rate |
|---|---|---|---|
| No pruning | Rules Alone | n/a | 100.00 |
| | Queries, Examples & Rules | 0 | 100.00 |
| | Examples & Rules | 0 | 100.00 |
| No prior knowledge | Queries & Examples | 175 | 94.98 |
| | Examples Alone | 33.2 | 79.26 |
| dependent_savings_adequate | Rules Alone | n/a | 75.06 |
| | Queries, Examples & Rules | 139 | 98.14 |
| | Examples & Rules | 31 | 92.70 |
| assets_high | Rules Alone | n/a | 93.36 |
| | Queries, Examples & Rules | 81 | 98.82 |
| | Examples & Rules | 23 | 92.38 |
| dependent_income_adequate | Rules Alone | n/a | 84.46 |
| | Queries, Examples & Rules | 57 | 98.94 |
| | Examples & Rules | 25 | 85.84 |
| earnings_steady | Rules Alone | n/a | 95.60 |
| | Queries, Examples & Rules | 0 | 95.60 |
| | Examples & Rules | 0 | 95.60 |
| debt_low | Rules Alone | n/a | 61.76 |
| | Queries, Examples & Rules | 189 | 97.94 |
| | Examples & Rules | 30 | 84.72 |
| savings_adequate | Rules Alone | n/a | 90.88 |
| | Queries, Examples & Rules | 65 | 98.82 |
| | Examples & Rules | 19 | 92.16 |
| income_adequate | Rules Alone | n/a | 64.64 |
| | Queries, Examples & Rules | 96 | 94.24 |
| | Examples & Rules | 32 | 81.18 |

Table 2: Performance summary

It is interesting to note that even without any prior knowledge the constructive algorithms perform significantly better than more traditional learning from examples

on a pre-specified architecture. In particular, observe in Table 2 that the constructive learning without prior knowledge from examples alone has average 79.26% accuracy. By contrast, backpropagation on networks with between 10 and 150 hidden units fared no better than 56.56% accuracy when provided with the same training and test sets.

Also note that when learning without the debt_low rule, both constructive algorithms showed an impressive increase in prediction quality. Predictive quality of 61.76% from rules alone increased to 97.84% for rules, examples and queries and to 84.72% for rules and examples. In comparison, the knowledge refinement technique of the extended KBANN rule-based network using additional connections and backpropagation as in [17] provided an increase to 64.64%.

Integration with an expert system was tested on the MONK's problems. The MONK's problems were developed in order to challenge a variety of machine learning approaches. Each of the three problems involves binary classification of a six feature domain, where each feature has two to four possible values.

The first problem is in disjunctive normal form. The second is similar to parity problems, and the third is similar to the first with the addition of five percent noise.

The rule base, representing good but imperfect prior knowledge, was generated by the PRISM algorithm of Cendrowska [4] and is based on Quinlan's induction algorithm ID3 [14]. Hyperplane determination from examples (HDE) is again used to combine the prior knowledge with the knowledge available from the example data.

| | Problem 1 | | Problem 2 | | Problem 3 | |
|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test |
| HDE | 94.35 | 85.19 | 95.27 | 73.61 | 91.80 | 75.69 |
| CLIPS | 100.00 | 86.34 | 98.82 | 68.06 | 100.00 | 93.98 |
| CLIPS & HDE | 91.13 | 87.04 | 100.00 | 81.02 | 92.62 | 93.98 |

Table 3: Generalization on the MONK's problems

The generalization accuracy of the individual learning algorithms on the MONK's problems and that of combining prior symbolic knowledge with machine learning is shown in Table 3. As anticipated, the generalization of a combined approach surpasses that of each approach individually.

# 5 Conclusions

In our experiments, the prediction quality increased markedly over rules alone when the constructive learning algorithm was used in conjunction with pre-existing symbolic knowledge. In all cases the performance increased when integrated with query-based learning. The performance also increased where the pre-existing knowledge is limited when examples alone were used in the constructive portion of the algorithm. The constructive learning technique appears to be most effective when integrated with query-based learning but still provides highly satisfactory results using rules and examples.

Neural networks efficiency and prediction quality depends significantly on how we select network architecture, learning algorithm and initial set of weights. The knowledge-based KBANN algorithm efficiently constructs the initial topology and weights from approximate pre-existing symbolic knowledge. However, KBANN's discovery of new knowledge contained in examples is restricted by a fixed network topology. The constructive algorithm efficiently learns not just connection weights but also extends the architecture as needed. The advantage of our integrated system is that given a pre-existing set of rules, a better initial hypothesis is generated from which constructive learning using examples and possibly queries may be used to efficiently extend the hypothesis.

# 6 Acknowledgements

The authors wish to thank Dr. Geoffrey Towell for providing his implementation of KBANN.

# References


[1] J. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research.* MIT Press, Cambridge, 1988.

[2] E. B. Baum. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2(1):5–19, January 1991.

[3] J. L. Bentley. Multidimensional binary search tree used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[4] J. Cendrowska. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 26,27(1,2,4;2,3,4), 1987.

[5] S. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, Denver 1989, 1990. Morgan Kaufmann, San Mateo.

[6] J. Fletcher and Z. Obradović. Creation of neural networks by hyperplane generation from examples alone. In *Neural Networks for Learning, Recognition and Control*, page 23, Boston, 1992.

[7] S. I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, June 1990.

[8] J. C. Giarratano. *CLIPS User's Guide.* Athens, GA, 1991.

[9] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, editors. *Building Expert Systems.* Addison-Wesley, Reading, MA, 1983.

[10] G. F. Luger and W. A. Stubblefield. *Artificial Intelligence and the Design of Expert Systems.* Benjamin/Cummings, Redwood City, CA, 1989.

[11] W. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 1943. Reprinted in [1].

[12] M. Mézard and J.-P. Nadal. Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A*, 22:2191–2204, 1989.

[13] Z. Obradović and R. Srikumar. Dynamic evaluation of a backup hypothesis. In *Neural Networks for Learning, Recognition and Control*, page 71, Boston, 1992.

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

[15] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, 1986. Reprinted in [1].

[16] S. B. Thrun et al. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CMU–CS–91–197, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991.

[17] G. G. Towell, J. W. Shavlik, and M. O. Noordwier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, July 1990.

[18] S. M. Weiss and C. A. Kulikowski, editors. *Computer Systems That Learn.* Morgan Kaufmann, San Mateo, 1991.