Performance Studies on KBANN

R. P. Jagadeesh Chandra Bose
Applied Research Group
Satyam Computer Services Ltd
Bangalore, India 560 012
Email: jagadeesh_bose@satyam.com

G. Nagaraja
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai, India 400 076
Email: gn@cse.iitb.ac.in

Abstract— **Hybrid models combining the analytical (rule-based) and connectionist (artificial neural network (ANN)) paradigms are called Knowledge Based Neural Networks (KBNN). The Knowledge Based Artificial Neural Network (KBANN) is one such model that makes use of the domain theory represented as propositional rules and training examples. In this article, we analyze the performance of KBANN and suggest some ideas of improving its capabilities. A study on the effect of inductive bias on KBANN, use of adaptive learning algorithms instead of the standard backpropagation to improve the training times and the use of regularization methods for improving generalization performance is presented. It is shown that for better performance, the initial weight assignment to links obtained by domain theory varies with the domain and the performance of KBANN is improved by using regularization methods like the weight decay along with Rprop training algorithm instead of the standard backpropagation.**

I. Introduction

Machine Learning research has witnessed an enormous growth during the past years. While machine learning in the past has been primarily oriented towards single-strategy systems, the new era has been increasingly concerned with building systems that integrate two or more learning strategies. Single strategy systems are limited in their capabilities as they apply only to a range of problems. In contrast, systems that combine different strategies enhance their capabilities. This has led to the emergence of a new dimension in machine learning research called Multi-Strategy learning. *A multi-strategy approach to learning represents a heuristic exploration of the potentially exponential space of integrations that combine multiple learning algorithms, representations and paradigms* [1]. One of the multi-strategic approaches that is increasing its popularity involves combining the inductive and analytical learning paradigms.

Inductive and analytical learning are two learning paradigms that have been successfully applied to many real world applications. Purely inductive learning methods formulate general hypotheses by finding empirical regularities over the training examples. Purely analytical methods use prior knowledge to derive general hypotheses deductively. Both these strategies have their own advantages and pitfalls. Hybrid systems that attempt to exploit the complementary advantages of knowledge based and connectionist paradigms to obtain more powerful and robust systems have been proposed by several researchers [2]–[8]. These hybrid methods use theoretical knowledge of a domain and a set of classified examples to develop a method for accurately classifying examples not seen during training. The challenge of hybrid learning systems is to use the information provided by one system to offset the information missing from the other source. Hybrid models combining the analytical (rule-based) and connectionist (artificial neural network (ANN)) paradigms are called Knowledge Based Neural Networks (KBNN). A few examples of such KBNN's are the Knowledge Based Artificial Neural Networks (KBANN) [6], Finite State Knowledge Based Artificial Neural Networks (FSKBANN) [7] that incorporate state information in the domain theories, the RAPTURE system [8] dealing with certainty factor rule bases. In this article, we analyze the KBANN [6] from its performance point of view and suggest some ideas for improving its capabilities. A study on the effect of inductive bias on KBANN and the usage of adaptive learning algorithms instead of the standard backpropagation along with the incorporation of regularization methods for improving generalization performance is presented. First we briefly present the KBANN system proposed by Towell and Shavlik [6].

II. KBANN

KBANN is a hybrid system that learns from both the domain theory and training examples. The building of a KBANN involves two steps:

1) *Rules-to-Network* translation
2) Training the built network using a learning algorithm

The *Rules-to-Network* translation is accomplished by establishing a mapping between a rule-set and a neural network. The rules must be propositional and acyclic. Further, the rules are hierarchically structured. For a detailed description of the algorithm, the reader is referred to [6]. Here, we present the important concept of assignment of weights corresponding to the domain knowledge in the built network. Rules contain the logical connectives AND, OR, and NOT. The rules-to-network translator sets the weights on links and biases of units such that units have significant activation only when the corresponding deduction can be made using the domain knowledge. KBANN translates conjunctive rules into a neural network by setting weights on all links corresponding to positive antecedents to ω, weights on all links corresponding to negated antecedents to $-\omega$, and the bias on the unit corresponding to the rule's

Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)
0-7695-2291-2/04 $ 20.00 IEEE

consequent to $(P - \frac{1}{2})\omega$. P being the number of positive antecedents in a rule. For disjunctive rules that are rewritten as defined in the *rules-to-network* translator [6], the weights are set to ω and the bias is set to $\frac{\omega}{2}$. Results reported by Towell and Shavlik [6] show better performance for KBANN as compared to both the domain theory and artificial neural networks implemented alone. Towell et al [6] suggested a weight value of $\omega = 4.0$ for good performance in their experiments.

III. Improving the Performance of KBANN

We analyzed the performance of KBANN and have tried to improve its training time and generalization capability. We have largely concentrated on two major aspects towards improving the performance of KBANN. They are:

1) Inductive Bias
2) Training/Learning Algorithm

A. Inductive Bias

Mitchell [9] defines *bias* as "any basis for choosing one generalization over another, other than strict consistency with the instances". Jardins et al [10] define *bias* as any factor that influences the definition or selection of an inductive hypotheses. The KBANN [6] paradigm is enriched with prior knowledge about a task that is used to initialize the network prior to training. Encoding of prior knowledge is done by programming some network weights to specified values instead of choosing random ones with a hope that the assigned weights define a good starting point in weight space leading to faster convergence. However, the accuracy of mapping the prior knowledge into a network is very important as a network may fail to take full advantage of poorly encoded prior knowledge and also altering the essence of prior knowledge by poor encoding may further hinder the learning process. There are advantages to making effective use of prior knowledge:

1) The learning performance may lead to faster convergence to a solution
2) Networks trained with hints may generalize better on future examples
3) Explicit rules may be used to generate additional training examples

As described earlier, in a KBANN, Towell and Shavlik [6] suggested that all weights which reflect prior knowledge be set to $w = 4$. We feel that this choice of inductive bias has two major drawbacks:

1) There is a possibility of different choices of the inductive bias w leading to faster convergence and good generalization performance for different applications.
2) Such a choice of weights cannot handle the uncertainties in the initial domain theory.

To test the effect of choosing a weight value, ω, we have tested the KBANN with ω ranging from 0 to 6 in steps of 0.1. Based on our results, it can be conjectured that a good choice of inductive bias depends on the application, the training data and the network architecture.

B. Training/Learning Algorithm

Our second direction towards improving the performance of KBANN is on the learning procedure. The KBANN [6] and its variants [7] with the exception of RAPTURE [8] have all been reported in the literature with the standard backpropagation as its training procedure. The training times are a bit high in the results published. From the perspective of training procedure, we have studied the following aspects on its improvement.

1) Using a different learning algorithm like the adaptive learning rules
2) Adapting the regularization principles for improving the generalization

The next two subsections present a brief review on these methods.

1) Improving the Performance of backpropagation: Gradient descent makes changes to the weights which are proportional to the gradient; the constant of proportionality is called the step size (or learning rate). If the step size is not chosen correctly, it may perform poorly. Several variants of gradient descent have been proposed as a remedy to this. Using second order information (i.e., second order derivatives), conjugate gradients adopting line searches, using adaptive step sizes are a few among others cited as a learning technique. Here, in order to test the performance of KBANN, we have chosen the gradient descent with adaptive step sizes to compare with backpropagation. Particularly, we have adopted the Resilient Backpropagation (Rprop) proposed by Riedmiller et al [11] as it has been shown to perform better among other adaptive techniques [11]. In the adaptive step size approach, the step size is increased when the gradient is improving performance, and decreased when the gradient is making the error worse. The reader is referred to [11] for details of the algorithm.

2) Improving Generalization: Generalization refers to the performance of a learned network over unseen examples. Controlling the complexity of the network is essential to getting good generalization. The simplest network which gives low apparent error is the one which is most likely to give the best generalization performance. One way to control the complexity is through the network architecture. A network with few weights will be less complex than the one with more weights. The task of finding the best network architecture is called *model selection*. Good generalization can be expected only if there is enough training data and the required training data (sample complexity) grows with the complexity of the network. Apart from the model selection technique, generalization can be improved using *regularization* methods. Regularization methods try to make the output of the network a smoother function of the input. Two regularization methods that we have incorporated in KBANN to test its performance are the

1) Early stopping
2) Weight decay

3) Early stopping: During neural network training, the true error will usually decrease initially. However, as training continues, the true error can start to increase. This is

because as training continues, weights become more tuned to particular features of training data. This phenomenon is called *over-fitting* or *over-training*. A regularization technique to cure over-training is early stopping. Early stopping is a technique for avoiding over-fitting in models that use iterative learning procedures. This approach in one way helps in overcoming the bias/variance dilemma.

In early stopping, a fraction of the training examples are held out for validation, and performance on this set is monitored while the iterative learning procedure is applied. Typically, the validation error will initially decrease as the model fits the data better and better, but later on, when the model begins to over-fit, the validation error starts rising. The idea is to stop learning as soon as this minimum in validation error is achieved. For neural networks, one of the great advantages of early stopping is that the difficult question of how long to train vanishes. One inherent problem in early stopping is to decide how large a fraction of the training data should be used for validation. We have two conflicting cases to choose; we want a large validation set to achieve a good estimate of performance and also we want a large training set to be able to train. For our experiments, we have chosen 20% of the training examples for validation. Another important question is the criterion for deciding when to stop the training. We do not want to stop training merely when small fluctuations in the validation error occur. We have used a simple rule of training until the iteration with smallest validation error lies 25% backwards in the run.

4) Weight decay: Another regularization method that we adopted to train KBANN is the weight decay. In this approach, the function which gradient descent minimizes is changed. A term is added to the error which is large if the magnitude of the weights is large and small if the magnitude of the weights is small. Thus, gradient descent is trying to make the error small, but at the same time is trying to make the size of the weights small. This prevents the weights from differing greatly in value, which smooths the output of the network. Adding some regularization (penalty) term to an objective function in the learning of neural networks can lead to significant improvements in network generalization. Supporting the heuristics of weight decay and early stopping, Bartlett [12] has shown that the sample complexity of pattern classification with neural networks depends on the size of the weights too in addition to the size of the network. Some mathematics involving the weight decay is presented next.

Let $E(W)$ be the error, which is a function of the vector of weights W. W includes all the weights of the network along with the thresholds of the nodes. Let $D(W)$ be a function of the weights which increases with the magnitude of the weights. A possible function is

$$D(W) = \sum_i W_i^2 \tag{1}$$

where W_i is the i^{th} weight. The function D will be zero if the weights are all 0 and will increase with the size of the weights. In weight decay, we do gradient descent on

$$E(W) + \lambda D(W). \tag{2}$$

This will be decreased either by making the error small or making the weights small or making both small. The quantity λ determines the relative importance of the two terms. If λ is very small, only the error will be minimized. If λ is very large, the weights will be made small with little regard to minimization of the error. Choosing the appropriate value of this quantity is important to making error minimization sufficiently important while still getting some smoothing effect of the second term. For our experiments, we have used the following term to be minimized.

$$E(W) + 10^{-\alpha} D(W) \tag{3}$$

where α is a parameter which determines the relative importance of the two terms and $\alpha > 0$. Determining an adequate regularization factor λ in (2) that can lead to better generalization performance is still an open research problem.

To study the utility of the regularization factor, we have tested the KBANN system with and without the weight decay and the results are discussed in the next section.

IV. Experimental Details

We find that the weight value of $w = 4.0$ for links representing domain theory as suggested by Towell and Shavlik [6] to be biased. In order to study the influence of different weight initializations to the links obtained by domain theory, we have run the KBANN by initializing the weights of the links defining the domain theory from 0 to 6 in increments of 0.1. As KBANN and most of its variants have been reported with backpropagation as its training algorithm and the training times reported being high, we tried to study the behavior of KBANN to adaptive learning techniques in comparison with backpropagation. For our experiments, we have chosen Rprop as a representative of adaptive learning technique.

Problem Description

The first problem studied is that of *promoter recognition*. Promoters are short DNA sequences that precede the beginnings of genes. The input features for promoter recognition are a sequence of 57 DNA nucleotides. The reference point for promoter recognition is the site at which gene transcription begins. The reference point is located seven nucleotides from the right. The rule set used in promoter recognition is available at [13]. There are 106 DNA sequences of which none satisfy the initial domain theory. The nucleic acids are encoded as A = 1000, T = 0100, G = 0010 and C = 0001.

In order to study the effect of bias more closely, we have tested the KBANN system on hypothetical problems like the xor boolean classification problem; particularly with the

10-input, 12-input and 14-input xor problem. Figure 1 shows the sequence of flow of our experimental setup. In order to

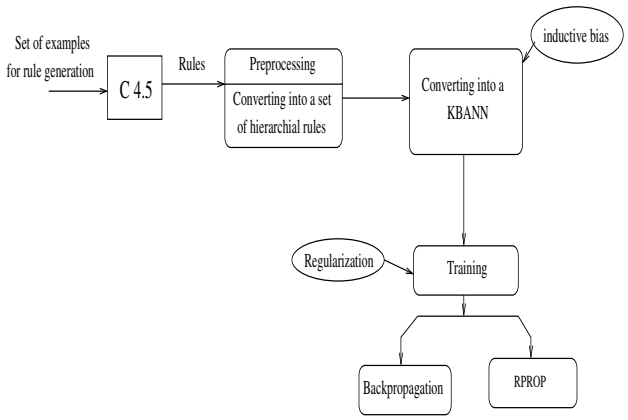

Fig. 1. Flow sequence of our testing environment

generate the domain theory, we used Quinlan's C4.5 [14] decision tree algorithm. The decision tree is then converted into a set of IF-THEN rules. The rules thus generated are pruned so that nearly correct rules are retained. Rules that convey no information on the property of the problem are also removed. For e.g., the xor-problem is "If the number of 1's in the input is odd then the output is 1, else 0". Suppose, a rule generated from the decision tree is "If $x_1 = 0$ and $x_4 = 0$ and $x_5 = 0$ and $x_8 = 0$ and $x_9 = 0$ Then NO"(Where the conclusion YES corresponds to the output 1 and NO to 0). Though the above rule might have been generated as a branch of the decision tree in light of many examples used in the tree construction satisfying it, it doesn't convey much about the xor property. Such rules are pruned. The pruned rules are then rewritten as required by KBANN. The rewritten rules are then converted to hierarchical rules (if need be) by selecting subsets of antecedents that occur most often (e.g., sets of 3 antecedents occurring in more than 2/5 of the rules can be separated out) in all the rules. For each set of such antecedents, a new consequent is made and that consequent is substituted for the set of antecedents in the original rule. A few rules thus obtained for the 14 input xor problem is shown below:

IF C1_1 = 1 OR C1_2 = 1 THEN No

IF C2_1 = 1 OR C2_2 = 1 THEN Yes

IF x_13 = 0 AND x_5 = 0 AND x_2 = 0 THEN T_1

IF x_5 = 1 AND x_13 = 0 AND x_4 = 0 THEN T_2

IF T_2 = 1 AND x_7 = 0 AND x_12 = 1 AND x_6 = 0 AND x_8 = 0 THEN C1_1

IF T_1 = 1 AND x_9 = 1 AND x_3 = 1 AND x_14 = 1 AND x_11 = 1 THEN C1_2

IF T_1 = 1 AND x_9 = 0 AND x_8 = 1 AND x_4 = 0 THEN C2_1

IF T_2 = 1 AND x_7 = 1 AND x_10 = 1 AND x_9 = 1 AND x_6 = 1 THEN C2_2

In the above rules, the consequents Ci_j correspond to consequents introduced by the rewriting rule of the KBANN 'rules-to-network' translation algorithm. The T_i's correspond to consequents introduced in converting the initial rules into hierarchical rules. The hierarchically rewritten rules are then mapped into a neural network. Positive antecedents are represented by positive weight ω and negative antecedents by $-\omega$. A consequents bias is set to $(P - \frac{1}{2})\omega$ or $\frac{\omega}{2}$ for conjunctive and disjunctive rules respectively, where P is the number of positive antecedents for that consequent.

V. Results and Discussion

A. Promoter Recognition

Figure 2 shows the SSE curves for the training set using backpropagation and rprop for different weight initializations The learning rate and momentum factors for backpropagation set as $\eta = 0.1$ and $\beta = 0.1$ respectively and the initial weight update Δ_0, the maximum weight update Δ_{max} to 0.01 and 15 respectively for the rprop. The weight decay parameter α is set to 3. The training set used consists of 85 examples. It can be

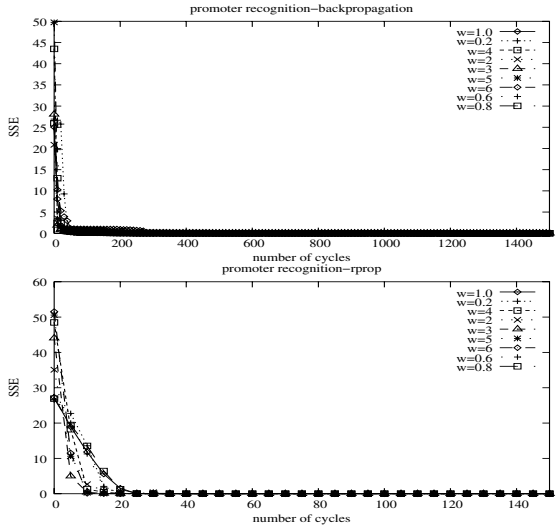

Fig. 2. Comparison of backpropagation and rprop over the training set for promoter recognition problem. $\eta = 0.1, \beta = 0.1$ for backpropagation and $\Delta_0 = 0.01, \Delta_{max} = 15$ for rprop and $\alpha = 3$ for weight decay

seen that there is no pronounced difference in the performance of KBANN for different weight initializations. One possible reason could be the fraction of links corresponding to the domain theory being quite small as compared to the total number of links. The training data being few along with the vast number of links makes the weight assignment to links obtained by domain theory less important.

B. XOR-10

The hierarchical rules are then converted to a network using the KBANN algorithm [6]. This resulted in a network with 10-17-15-2 architecture where the first and last numbers correspond to the number of input and output nodes respectively and the middle numbers denote the number of hidden nodes in each layer. The inputs are fully connected to the nodes in a layer above while the others are not fully connected. This is done in order to capture the essence of the domain theory that for the outputs only the antecedents obtained through domain theory should contribute. In order

to reduce the complexity of the network, we tried to remove some rules from the domain theory obtained by the pruned decision tree and check the performance of KBANN. We have removed rules that are less satisfactory over the examples used in rule generation i.e., we removed rules that have a less hit-ratio ($\frac{\text{number of correctly classified}}{\text{number of correctly classified+number of wrongly classified}}$).

Backpropagation Vs Rprop

We have applied the backpropagation and the rprop [11] algorithms and the error function used is as given in Eqn(3). Figure 3a and Figure 3b show the SSE curves for the training and validation sets using backpropagation and rprop with the weights initialized to $\omega = 0.8$ and the learning rate and momentum factors for backpropagation set as $\eta = 0.1$ and $\beta = 0.1$ respectively and the initial weight update Δ_0, the maximum weight update Δ_{max} to 0.01 and 15 respectively for the rprop. The weight decay parameter α is set to 3. These are the parameters found best for the architecture. The training set used consists of 702 examples and the validation set is of 192 examples. As can be seen from Figure 3, rprop

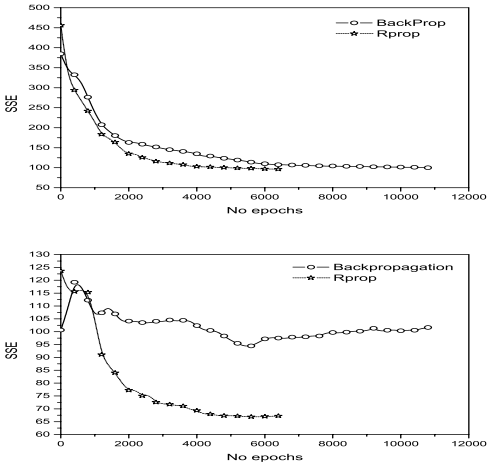

Fig. 3. Comparison of backpropagation and rprop over the training and validation sets for xor-10 problem in (10-25-2) network. $\eta = 0.1, \beta = 0.1$ for backpropagation and $\Delta_0 = 0.01, \Delta_{max} = 15$ for rprop and $\alpha = 3$ for weight decay

performs better than backpropagation both on training and validation sets and much better especially on the validation set. Figure 4a and Figure 4b show the training and validation SSE curves for backpropagation and rprop respectively for the same architectural and parameter setups as defined above. It is to be noted that for backpropagation, the validation curve doesn't resemble the training curve and is more oscillating. In contrast, the validation curve for the rprop follows the trajectory of the training curve. This helps us in generalizing better than that of backpropagation. Also to be noted from Figure 4b is that the validation error for backpropagation is much larger than that of rprop indicating that backpropagation is trying to over-fit the training data.

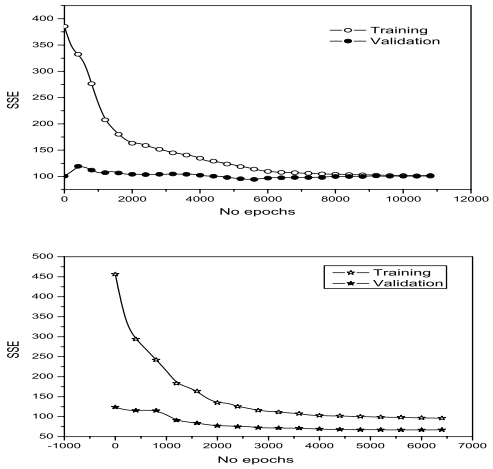

Fig. 4. SSE curves for the training and validation sets for backpropagation for xor-10 problem in (10-25-2) network. $\eta = 0.1, \beta = 0.1$ for backpropagation and $\Delta_0 = 0.01, \Delta_{max} = 15$ for rprop and $\alpha = 3$ for weight decay

1) Inductive Bias: We have tested the KBANN on different weight initializations for the links obtained by domain theory. We have run the experiments for weights ranging from 0.1 to 6.0 in steps of 0.1. Figure 5a and Figure 5b show the SSE curves for different weight initializations for backpropagation and rprop respectively for different parameters on the training set. The depicted curves are for the best parameters among the ones tried out. Only a few of the weight values between 0.0 and 6.0 are shown. It can be noted that the error rates are less for weight values in the range of 0.8 in contrast to a weight value of 4.0 as suggested in Towell et al [6]. So, the initialization of weights is largely dependent on the application domain. Also the error rates for backpropagation is more sensitive to the initial weight values as compared to rprop. Figure 5c and Figure 5d show the SSE curves for different weight initializations for backpropagation and rprop for different parameters on the validation set. An important point to note is that the validation curve for backpropagation doesn't follow the trajectory of its training data whereas that of rprop does.

2) Weight decay: Figure 6a and Figure 6b show the performance for training and validation sets respectively for xor-10 problem with 10-25-2 network when weight decay is applied on rprop algorithm. The α value is set to 3.0. It is to be noted that the performance of KBANN is better when regularization methods (weight decay) are used. Also, we have observed that for large initial weight values to links obtained from domain theory, the parameter α is less than that for small initial weights. This indicates preference of the network to have small weights.

VI. Conclusions

We have studied ways of improving the performance of KBANN. We concentrated on two major aspects viz., that of the inductive bias and training methods. Most of the hybrid

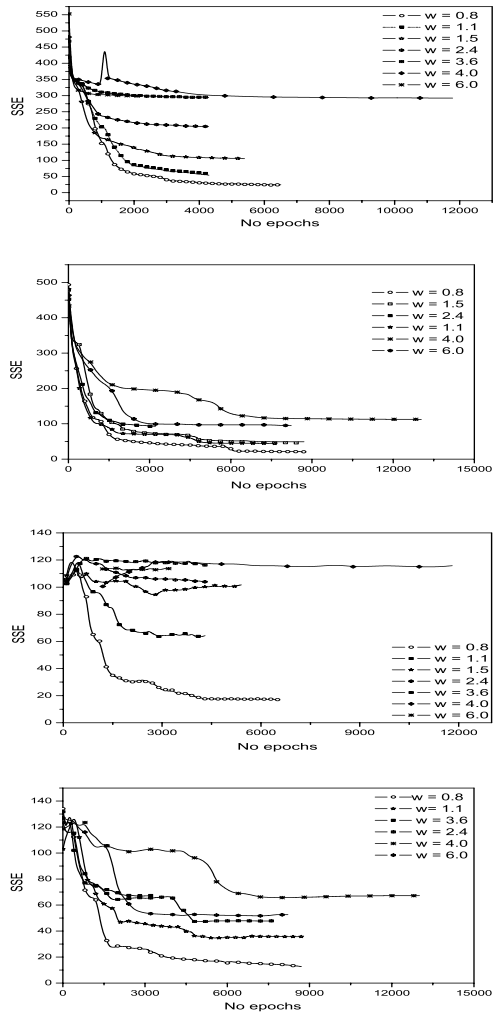

Fig. 5. Comparison of SSE for different weight initializations for backpropagation and rprop for xor-10 classification using 10-25-2 network architecture on training and validation sets

systems proposed (with the exception of RAPTURE [8]) use the standard backpropagation for training. However, the training times are found to be pretty high in those cases. We explored ideas of using advanced adaptive learning techniques for KBANN. In particular, we used the Rprop (Resilient Backpropagation) [11]. Our results indicate that usage of adaptive learning techniques reduce the generalization error and training time. Also, a bias value of $\omega = 4.0$ as suggested by Towell et al [6] for links obtained by domain theories doesn't always apply. We have tested the KBANN system by altering the initial weights between 0.1 and 6.0 and have found that the bias value is dependent on the problem domain. Noted also is the improvement in the performance of KBANN by using regularization methods viz., weight decay during training. Our results show that KBANN has a preference for small weight values. Small weight values has high significance even from the sample complexity perspective of neural networks as it has

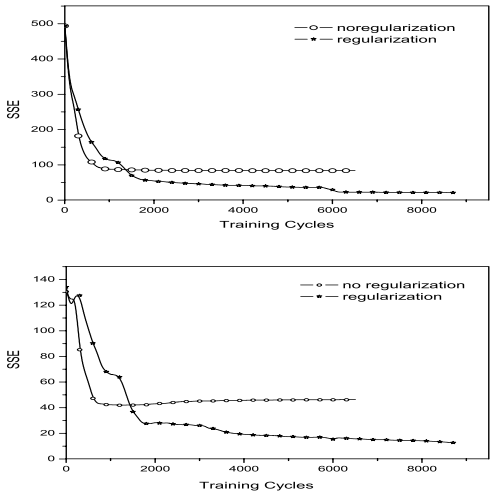

Fig. 6. SSE for training set for xor-10 problem with 10-25-2 network with and without weight decay using rprop

been shown recently that the sample complexity depends both on the number and size of the weights [12].

REFERENCES

[1] R. S. Michalski, "Inferentical theory of learning: Developing foundations for multistrategy learning," *Machine Learning:A Multistrategy Approach*, vol. 4, 1993.

[2] L. M. Fu and L. C. Fu, "Mapping rule-based systems into neural architecture," *Knowledge-Based Systems*, pp. 48–56, 1989.

[3] L. Fu, "Knowledge-based connectionism for revising domain theories," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 1, pp. 173–182, 1993.

[4] S. I. Gallant, "Connectionist expert systems," *Commun. ACM*, vol. 31, no. 2, pp. 152–169, 1988.

[5] C. Glover, M. Silliman, M. Walker, and P. Spelt, "Hybrid neural network and rule-based pattern recognition system capable of self-modification," in *Proceedings of SPIE, Application of Artificial Intelligence VIII*, 1990, pp. 290–300.

[6] G. G. Towell and J. W. Shavlik, "Knowledge based artificial neural networks," *Artificial Intelligence*, vol. 70(1-2), pp. 119–165, 1994.

[7] R. Maclin and J. W. Shavlik, "Using knowledge-based neural networks to improve algorithms: Refining the Chou-Fasman algorithm for protein folding," *Machine Learning*, vol. 11, pp. 195–215, 1993.

[8] J. J. Mahoney and R. J. Mooney, "Combining connectionist and symbolic learning to refine certainty factor rule-bases," *Connection Science*, vol. 5, no. 3-4, pp. 339–364, 1993.

[9] T. M. Mitchell, *Machine Learning*. Tata Mc.Graw Hill, 1997.

[10] M. D. Jardins and D. F. Gordon, "Evaluation and selection of biases in machine learning," *Machine Learning*, vol. 20, pp. 1–17, 1995.

[11] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms," *International Journal of Computer Standards and Interfaces; Special Issue on Neural Networks*, vol. 16, no. 3, pp. 265–275, 1994.

[12] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[13] C. Blake and C. Merz, "UCI repository of machine learning databases," 1998, University of California, Irvine, Dept. of Information and Computer Sciences. [Online]. Available: "http://www.ics.uci.edu/~mlearn/MLRepository.html"

[14] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1992.