A Knowledge-Based Approach to Supervised Incremental Learning

LiMin Fu[1], Hui-hunag Hsu[2] and Jose C. Principe[3]

[1] Department of Computer and Information Sciences
[2,3] Department of Electrical Engineering
University of Florida, Gainesville, FL 32611, USA

Abstract

How to learn new knowledge without forgetting old knowledge is a key issue in designing an incremental-learning neural network. In this paper, we present a rule-based connectionist approach in which old knowledge is preserved by bounding weight modifications. In addition, some heuristics are developed for avoiding overtraining of the network and adding new hidden units. The feasibility of this approach is demonstrated for classification problems including the iris and the promoter domains.

1 Introduction

An incremental learning system updates its hypotheses as a new instance arrives without reexamining old instances. In other words, an incremental-learning system learns Y based on X, then learns Z based on Y, and so on. Such a learning strategy is both spatially and temporally economical since it need not store and reprocess old instances. It is especially crucial for a learning system which continually receives input and must process it in a real-time manner. Most (maybe all) of the neural networks with incremental learning capability developed to date do not exploit existing domain knowledge. In this paper, we present a knowledge-based approach to incremental learning and report our experimental results.

2 Incremental Learning Strategies

2.1 The Rule-Based Approach

A rule-based inference (or problem solving) system can be mapped into a neural network architecture as follows. First, data attributes or variables are assigned input units (nodes), target concepts or final hypotheses are assigned output units, and intermediate concepts or hypotheses are assigned hidden units. Then, the initial domain rules determine how the attributes and concepts link and how the links are weighted (Fu 1993). The neural network so built is referred to as a *rule-based connectionist network*.

Each rule consists of a premise (antecedent) and a consequent. In the network configuration, the premise is assigned a hidden unit (called a conjunction unit), each condition corresponds to an assigned attribute or concept node, and the consequent corresponds to an assigned concept node. Each condition node is connected to the premise node which in turn connected to the consequent node. Under such construction, the rule strength corresponds to the weight associated with the connection from the premise node to the consequent node. In addition to knowledge-based connections, we may add some hypothetical connections to increase the learning capacity of the network.

2.2 Bounded Weight Modification

When knowledge is represented by a weight vector in a multidimensional space, as in the connectionist framework, the knowledge vector can often be moved in a range while preserving its truth. The limit of this range is called the *validity bound* of knowledge. When knowledge involves some uncertainty, we further define a range of uncertainty, whose boundary lies beyond the validity bound and is called the *uncertainty bound*. The purpose of learning is to shrink the uncertainty range until its bound coincides with the validity bound. Hereinafter, the bound always refers to the uncertainty bound. A bound on weight modification is imposed in order that previous network knowledge is preserved while there is room for refining the knowledge.

In the rule-based connectionist network, two kinds of weights can be recognized: weights associated with in-connections and weights associated with out-connections of conjunction units. The former weights measure the importance of each condition in the rule's premise, while the latter weights correspond to rule strengths. The incoming weight vector of a conjunction unit encodes the corresponding rule pattern, for which directionality is more important than magnitude, since it serves as a basis function in the network. By contrast, magnitude is more important than directionality for rule strengths, which weigh values produced by different rules.

Instead of a scalar weight, we have a weight interval if a bound is imposed. The semantics of this bound depends on the role of the associated weight. For rule strengths, a weight interval means a conditional belief interval, which represents the range of uncertainty on the fact to be concluded given the premise. In the case of the weight vector encoding a rule pattern, the bounds on individual components create a geometrical space, which represents the range of uncertainty for directionality.

We choose a larger bound for weight modification if the network knowledge is more uncertain. As learning proceeds, the bounded weights should converge asymptotically on precise knowledge. A bound can be defined centering around the initial knowledge. Thus search for weights on a new instance is confined to the neighborhood of the initial knowledge. Alternatively, we can define a dynamic bound centering around the current knowledge. In this case, the bound may shift away from the initial knowledge. This latter approach is more useful when the data statistics are not stationary or when the initial knowledge is not accurate.

Two different learning strategies can be formulated: fast learning and slow learning. In fast learning, weights are adjusted iteratively on each new instance to a full extent as long as the defined bound is not exceeded. In slow learning, at most one weight change is allowed for each new instance. The goal of slow learning is to achieve gradual convergence upon a minimum rate of misclassifications, and is not intended to rectify every misclassification at once. Fast learning is more unreliable since a single instance does not provide adequate constraints for the learning process. Hence, only slow learning is currently adopted.

When the weight modification is bounded, it is likely that adjustments of different weights may be cut off at different proportions. As a result, the network weight vector (i.e., the collection of all weights as a vector) may not move in the steepest descent during error minimization. This problem is dealt with by introducing a scaling factor s which scales down all weight adjustments so that all of them are within bounds. The learning rule is thus

$$\Delta W_{ji}(k) = s(k)\eta\delta_j(k)O_i(k) \tag{1}$$

where W_{ji} is the weight from unit i to unit j, η $(0 < \eta < 1)$ is a trial-independent learning rate, δ_j is the error gradient at unit j, O_i is the activation level at unit i, and the parameter k denotes the kth iteration. Suppose the bound on weight adjustment for an instance is B such that

$$|\sum_k \Delta W_{ji}(k)| < B$$

The scaling factor s for the nth iteration is set by

$$s(n) = \min_{j,i}[1, (B - |\sum_{k=1}^{n-1} \Delta W_{ji}(k)|)/\eta\delta_j(n)O_i(n)] \tag{2}$$

The bound B is based on prior knowledge or determined empirically. In the experiments conducted, we used the CF-based activation function and set the bound B to 0.01.

2.3 Learning Heuristics

To avoid over-training, weights are not adjusted if the newly seen instance is well-covered by the knowledge of the net. For this purpose, we define ***output margin*** as the difference between the largest and the second largest outputs in the case of multiple output units, or as the difference between the output and 0.5 in the case of a single output unit. A larger margin is interpreted as a higher certainty for classifying the instance.

Table 1. A domain theory for iris.

R1	If petal-length ≤ 2.7	then setosa
R2	If petal-length > 2.7 & ≤ 5.0, and petal-width > 0.7 & ≤ 1.6	then versicolor
R3	If petal-length > 5.0	then virginica
R4	If septal-width ≤ 2.8, and petal-width > 1.6	then virginica
R5	If septal-width > 2.8 & ≤ 3.1, and petal-width > 1.6	then virginica
R6	If septal-width > 3.1, and petal-length > 2.7 & ≤ 5.0	then versicolor

As a heuristic, only when the margin is below a certain threshold, the network weights are allowed to be adapted.

If the network cannot correctly classify an instance by modifying its weights within bounds, a hidden unit will be added to implement that instance as a specific rule. However, it is improper to add a hidden unit for any instance which is strongly contradictory to the knowledge of the net. This is the case when the net gives a strong support for a class different from the one labeling the instance. Thus, no hidden unit will be added if the margin is greater than a predefined threshold.

3 The Learning Procedure

Learning proceeds in the following steps:

1. Feed in the current instance and forward the activation to the output units.
2. Calculate the output margin.
3. Check the current output and the desired output. If the instance is misclassified, go to step 5.
4. If the output margin is less than a predefined threshold (e.g., 0.25), the net learns this instance once by backpropagation (Rumelhart, Hinton, and Williams 1986). Go to step 1.
5. Train the net by backpropagation with the weights bounded until:
   a. the instance can be correctly classified, or
   b. MSE is less than a predefined value (e.g., 0.001), or
   c. a certain number of iterations is reached.
6. If the instance can be correctly classified after training, restore the weights obtained by the first iteration of learning and go to step 1. Otherwise, restore old weights (the weights before learning).
7. If the output margin is below a predefined value, add one hidden unit and related connections to the net. Go to step 1.

4 Experimental Results

Two problem domains were used to evaluate the learning system. The first domain was the classification of iris flowers into three classes: setosa, versicolor, and virginica. There are 150 instances, 50 for each class. Each instance has four attributes: septal length, septal width, petal length, and petal width. The second domain was the recognition of promoters in DNA nucleotide strings. There are 53 positive instances and 53 negative instances. Each instance is composed of 57 sequential nucleotides. Fifty nucleotides before (minus) and six following (plus) the site where transcription initiates. And each nucleotide has four possible base types: A (Adenine), G (Guanine), C (Cytosine), and T (Thymine).

Each attribute of all iris instances was discretized into three levels. Six rules (in Table 1) were used as the initial knowledge for the iris classification problem. Because there are two inconsistent instances due to the discretization process, the best possible performance in terms of classification accuracy is 98.7 percent. In the promoter domain, we used the 14 rules given in Towell, Shavlik, and Noordewier (1990).

Table 2. Iris Domain: Test of Incremental Learning Capability

Cases	Classification Accuracy			Number of Hidden Units	
	initial	best	final	initial	final
1 instance	33.3%	97.3%	97.3%	1	9
1 rule	33.3%	98.7%	97.3%	1	9
2 rules	66.0%	96.7%	96.0%	2	10
3 rules	93.3%	98.0%	98.0%	3	8
4 rules	93.3%	98.7%	98.7%	4	6
5 rules	96.0%	98.7%	98.7%	5	7
6 rules	93.3%	98.7%	98.0%	6	9
MLP 12-3-3 (15 instances)	94.0%	98.0%	97.3%	3	5

Table 3. Promoter Domain: Test of Incremental Learning Capability

Cases	Classification Accuracy			Number of Hidden Units	
	initial	best	final	initial	final
14 rules (20 instances)	84.9%	96.2%	96.2%	14	14
14 rules (52 instances)	95.3%	99.1%	99.1%	14	14
MLP 228-14-1 (20 instances)	78.3%	96.2%	96.2%	14	14
MLP 228-14-1 (52 instances)	91.5%	99.1%	99.1%	14	14

We evaluated the performance of an incremental learning system in the respects of memorization of old knowledge and generalization to unseen instances. Suppose n instances are available to test our system. We test the network against the n instances when the network is learning the kth instance. The result reflects how well the network remembers $k-1$ instances and how well it generalizes to $n-k$ instances. As k approaches n, the test on generalization capability is deemphasized.

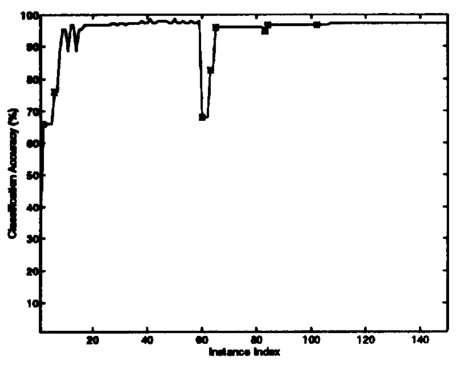
Initial knowledge exists in three forms:

- Initial rules which are directly mapped into a neural network without initial training
- Initial rules which are directly mapped into a neural network, which is then trained on a certain number of initial instances. In the tables 2 and 3, this case is shown by putting the number of initial instances in parentheses under the number of initial rules.
- Initial instances used to train a fully connected neural network before incremental learning starts.
- An initial single instance which is directly mapped into a neural network without initial training

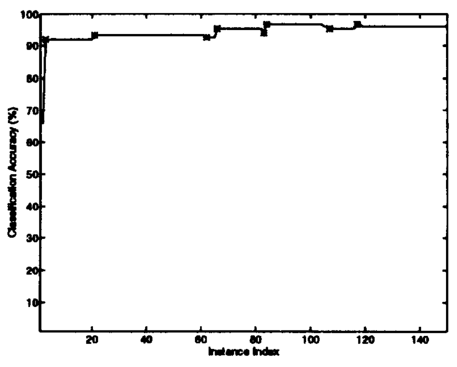
The test results are summarized in Tables 2 and 3, and Figure 1. Notice that in the iris domain, R6 is not quite correct, and therefore, both the initial and the final classification accuracies of the five-rule case are better than those of the six-rule case.

Figure 1 shows the learning curves of some cases in terms of classification accuracy. Oscillation occurs to a different extent, but in all cases, final convergence is achieved. In Figure 1(a) (the case of one rule in the iris domain), there is a big dip at instance #60, which is caused by mistakenly adding a hidden unit for an ambiguous instance due to weak initial knowledge of this case. A similar observation is made in one-instance case.

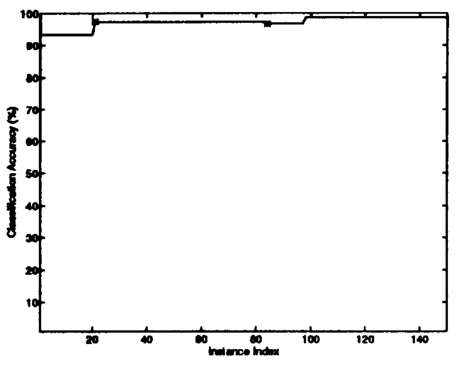
Oscillation in the learning curve is caused by the learning of new instances. Although weight modifications are bounded, learning of new instances might still cost some old knowledge. So, there is a compromise between keeping old knowledge and learning new instances.



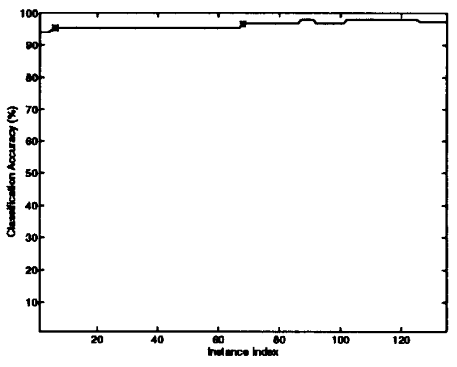
(a) Iris: One Rule



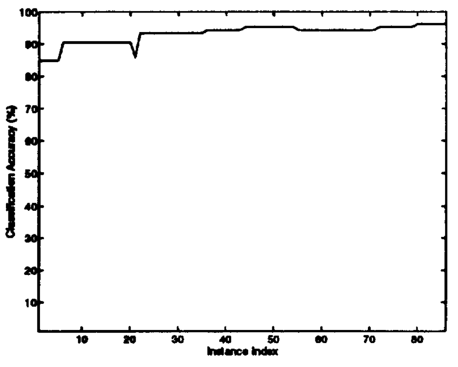
(b) Iris: Two Rules



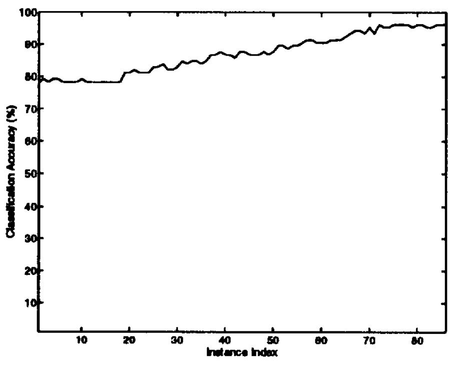
(c) Iris: Four Rules



(d) Iris: MLP 12-3-3 (15 instances)



(e) Promoter: Fourteen Rules (20 instances)



(f) Promoter: MLP 228-14-1 (20 instances)

Figure 1: Learning curves in terms of classification accuracy. " * " marks where hidden units are added.

5 Comparison with Related Work

Our system has the following advantages in comparison with related work. It differs from the ART network and its derivatives for supervised learning (Carpenter et al. 1992; Lee and Lai 1993) mainly in that it does not have to rely on clustering for incremental learning. It differs from the cascade correlation network (Fahlman and Lebiere 1990) in that it need not grow many hidden layers. It differs from the probabilistic neural network and its derivatives (Specht 1990) in that it does not rely on case-based memory. Finally, it differs from all other incremental learning neural networks (e.g., Chen and Soo 1993) in that it can fully exploit existing rule-based knowledge and thus minimizes its need for test instances.

6 Conclusion

Several conclusions can be drawn from current experimental results on incremental learning:

- Incremental backpropagation learning is a feasible learning strategy for classification problems especially if the network possesses sufficient initial knowledge. The network may start with null knowledge and achieve a satisfactory result. But, in general, the more initial knowledge, the better the result.
- Both the quantity and the quality of initial knowledge will influence the learning outcome. Incorrect initial knowledge is worse than no knowledge.
- A desired learning curve is one with increasing smoothness over time. A bad learning curve is characterized by oscillation, which may imply insufficient initial knowledge or too large the learning rate or improper adding of hidden units.

7 References

1. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., and Rosen, D.B., 1992. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3, pp. 698-713.

2. Chen, H.W., and Soo, V.W. 1993. Design of adaptive and incremental feed-forward neural networks. In *Proceedings of IEEE International Conference on Neural Networks (ICNN-93)* (San Francisco, CA), pp. 479-484.

3. Fahlman, S.E., and Lebiere, C. 1990. The cascade-correlation learning architecture. Tech. Rep. CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

4. Fu, L.M. 1993. Knowledge-Based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4), pp. 173-182.

5. Lee, H.M., and Lai, C.S. 1993. Supervised fuzzy ART: Training of a neural network for pattern classification via combining supervised and unsupervised learning. In *Proceedings of IEEE International Conference on Neural Networks (ICNN-93)* (San Francisco, CA), pp. 323-328.

6. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. Learning internal representation by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1. MIT press, Cambridge, MA.

7. Specht, D.F. 1990. Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. *IEEE Transactions on Neural Networks*, 1(1), pp. 111-121.

8. Towell, G.G., Shavlik, J.W., and Noordewier, M.O. 1990. 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.