Incremental Learning with Rule-Based Neural Networks[1]

C. M. Higgins R. M. Goodman
chuck@electra.caltech.edu rogo@csvax.caltech.edu

Department of Electrical Engineering, 116-81
California Institute of Technology
Pasadena, CA 91125

Abstract

A classifier for discrete-valued variable classification problems is presented. The system utilizes an information-theoretic algorithm for constructing informative rules from example data. These rules are then used to construct a neural network to perform parallel inference and posterior probability estimation. The network can be 'grown' incrementally, so that new data can be incorporated without repeating the training on previous data. It is shown that this technique performs comparably with other techniques such as back-propagation while having unique advantages in incremental learning capability, training efficiency, knowledge representation, and hardware implementation suitability.

1 Introduction

The problem of learning to classify from examples can be described as follows. Define an *attribute* as a variable which can take on a fixed number of values (not necessarily just two). Let there be a fixed number of attributes and designate one of them as the *class attribute*. Then the problem of learning to classify from examples is to learn to predict the value of the class attribute given the values of the other attributes by looking at examples of instances of all the attributes.

It is a significant advantage if such a system, having been trained on a set of examples, can later be trained on an additional set of examples without repeating the training process on the first set. We refer to this feature as incremental learning. For example, if such a system is found to have unsatisfactorily learned the representation of a class, it may be provided with more examples of the class.

This task cannot be accomplished in general by looking at only first-order correlations in the attributes; it requires higher-order information. Systems which collect *all* such higher-order correlation information were proposed by Uttley ([1],1960). Uttley's system collects an amount of information exponential in the number of attributes, much of which is not actually necessary for performing classification. Systems that randomly choose which higher-order correlations to store were among the first neural networks ([2], Rosenblatt 1962). The "hidden nodes" in a back-propagation network ([3], Rumelhart and McClelland 1989) store such higher-order information; their (arbitrarily chosen) number decides the amount of such information to store.

Rules form a natural framework for representing such information. We define a rule as a conjunction of attribute values on the left-hand (if) side and a single class attribute value on the right-hand (then) side.

IF $A = 1$ AND $B = 1$ THEN $C = 1$ WITH PROBABILITY 0.96

The number of attribute values on the left-hand side of the rule is called the rule *order*. Included with each rule is a strength which is the probability that the right-hand side is true given that the left-hand side is true.

Our approach uses an efficient algorithm to discover such rules based upon information theory. While rules obviously provide a clear method of ascertaining what has been learned, we will also show that there exists a straightforward method of constructing a neural network from the learned rules which can perform classification and posterior probability estimation. This network can be updated incrementally as desired.

[1]This work was supported in part by the Army Research Office under contract number DAAL03-89-K-0126, and in part by DARPA under contract number AFOSR-90-0199.

0-7803-0164-1/91/0000-0875$01.00©1991 IEEE

2 Information Theory for Constructing Rules

In this section, we describe the information-theoretic measure for rule 'value', and present an algorithm for discovering rules from examples.

2.1 The J-measure

The key to discovering rules from examples is a measure of the value of a rule based on the given example set. Such a measure for conjunctive rules called the *J-measure* has been developed by Smyth and Goodman[4] based upon the work of Blachman[5]. This measure tells the information content of a rule with respect to the example set.

Let a rule be defined as *if* y *then* x, where y is a conjunction of values of attributes and x is a value of the class attribute. Then the J-measure is defined as follows:

$$J(X;y) = p(y)\left(p(x|y)\log\left(\frac{p(x|y)}{p(x)}\right) + p(\bar{x}|y)\log\left(\frac{p(\bar{x}|y)}{p(\bar{x})}\right)\right)$$

The probabilities are estimated from the training set of examples. The rule with the greatest J-measure is the most informative.

2.2 The Rule Search Algorithm

Given a way to rank rules, we need a way to search the space of all possible rules in such a way as to select the best rules without covering the entire space, whose size is exponential in the number of attributes. Several search algorithms have been tried, including a constrained search of all the possible rules [ITRULE,[6]]. The following search algorithm searches a smaller subset of the space than previous algorithms by using the examples directly as templates for rules.

Given a training set of examples, an obvious way to classify is to retain all the examples and match an incoming example to be classified to an example in storage. This is equivalent to regarding the examples as very high-order specific rules. However, these rules will not match any example not explicitly contained in the training set and also model the 'noise' or randomness in the training set. Consider now if we could decide which attributes in each example to remove in order to generalize the examples to rules which cover more examples and remove the statistically insignificant 'noise' in the data set; the J-measure provides just such a way.

The algorithm for rule generation is as follows. If there are N attributes excluding the class attribute, each initial rule is of order N. For each rule independently, do the following:

1. Calculate the J-measure for the rule. Call this rule the parent-rule.
2. For each of the child-rules generated by removing a single attribute from the parent-rule, calculate the J-measure (If the parent-rule was order K, each of the K child-rules is order $K-1$).
3. Choose the rule among the parent rule and the set of child-rules with the greatest J-measure. Special cases:
   (a) If two rules have the same J-measure, choose the one with the lower order.
   (b) If two rules of the same order have the same J-measure, choose a random one.
4. If the chosen rule is not the parent rule, the chosen rule becomes a new parent rule; repeat the process starting at step 1. If the chosen rule is the parent rule, terminate.

To learn rules incrementally, perform the above algorithm on each example in the initial training set. Retain for each rule the original example which generated it. When more training data is available, use the above algorithm on the new training data. On the rules generated from the previous training set, also run the above algorithm with the change that child-rules now include those generated by *adding back each previously removed attribute*; thus child-rules now have order one more or one less than the parent rule. If the second training set was statistically similar to the first, the rules generated from the first set will

not change. However, if the second training set changes the statistics of the examples, the rules generated from the first training set may have over-generalized or over-specialized. This modification allows the rules to become either more general or more specific if necessary. J-measures in all cases are calculated with respect to the concatenation of the two training sets.

Before conversion to a neural network, duplicate rules can be removed to reduce the complexity of the network generated. However, the weight of each removed rule should be added to its duplicate which remains. Thus if a rule had five duplicates, it will have six times its original weight. This serves to preserve the statistics of the original data set.

The algorithm for the initial rule set in which rule order strictly decreases must terminate in $N+1$ iterations or less, where N is the number of attributes excluding the class attribute. This is because rule order decreases every time and cannot decrease beyond zero. It is more difficult to see a bound for the number of iterations of the incremental algorithm. Clearly, they cannot exceed 2^N, since this is the total number of rules which can be generated from any example. The number of iterations depends upon the smoothness of the J-measure search space. In our experience with the algorithm, it has never been noted to exceed $N+2$ iterations.

3 A Neural Network for Classification

Once the rules are constructed, they may be used in parallel to compute the posterior probability of each class. This approach to rule-based classification was described in detail in [7].

Let the subset of all the rules whose left-hand sides are satisfied by an example E in the training set and whose right hand side conclude $X = x_j$ be called R_j. Then we estimate the probability that $X = x_j$ as

$$\log(\hat{p}(x_j|E)) = \log(p(x_j)) + \sum_{i=1}^{|R_j|} W_{i,j} \text{ where } W_{i,j} = \log \frac{p(x_j|y_i)}{p(x_j)}$$

Once the posterior probabilities are estimated, we need only choose the largest probability to make our classification decision.

The above formula provides a simple method of constructing a neural network[8]. Consider the network of figure 1. The input layer contains one node for each attribute except the class attribute. Each node in the second layer represents a rule generated by the algorithm. Nodes in this layer are connected to the input nodes of the attributes in the left-hand side of the rule which they represent; they output a 1 if the left-hand side of the rule is satisfied. The third layer contains a node for each value of the class attribute. Each second-layer node representing rule i is connected to third-layer node j with a multiplicative weight $W_{i,j}$. The bias of each third-layer node is $-\log(p(x_j))$. Each third-layer node sums its inputs, subtracts the threshold and *exponentiates* the result. Thus, by the above formula, the output of each third-layer node is the posterior probability of the class it represents. If desired, a winner-take-all stage can be added to decide upon the most likely class.

4 Experimental Results

In this section, we show the results of experiments with the rule-based neural network to verify its effectiveness as a classifier and test the ability of the incremental algorithm.

4.1 Comparison with Other Approaches

We now compare the performance of our classifier (given the whole training set at once) with several other classification schemes. The classical first-order Bayes classifier was chosen for comparison as an example of a system which does not use higher-order information. However, this system is very simple and performs quite well for many data sets. The back-propagation network was chosen as a very successful 'black box' approach to neural network learning which uses higher-order information but is opaque to the user. Also shown for comparison is a trivial method in which the class that appears most in the examples is always picked (i.e. the most likely a priori class).

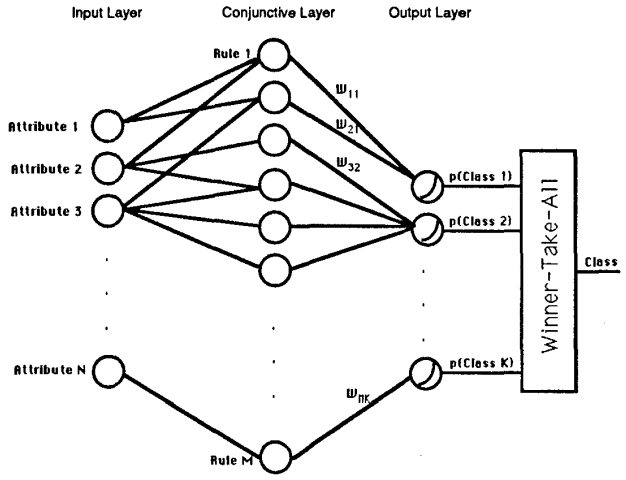

Figure 1: Rule-Based Neural Network

	Optimal Rate	Maximum a priori	Back Propagation	First-order Bayes	Rule-based Neural Network
LED	74.0%	17.0%	68.4%	68.2%	68.1%
Voting	$\sim$ 96.0%	53.3%	93.6%	91.1%	94.7%
Boolean	90.0%	67.3%	90.0%	67.5%	90.0%

Figure 2: Performance Comparison with Other Approaches

Three data sets were chosen for training. The first of these is the well-known LED digits problem in which the system must decide which digit is being shown on a 7-segment display given the value of each segment. Noise has been added to the examples so that the optimal classification rate is about 74%. The second data set consists of 435 voting records from a session of the 1984 US Congress [9]. Each example corresponds to a particular congressman and the attributes correspond to their votes on 16 different issues. The system must decide on the party affiliation of each congressman. Not surprisingly, it is possible to predict this very well. The third and final data set has been generated from the following boolean function:

$$x = (y_1 \oplus y_2) + (y_3 \cdot y_4) + (y_5 \cdot y_6)$$

To introduce noise, the class variable x has a 10% chance of being 'reversed' from its true state. Thus the optimal rate on this data set is 90%. This data set is designed to require the use of higher-order information.

The data sets were divided into disjoint training/testing sets for generating comparative results. The splits were LED: 154/846, Voting: 200/235, and Boolean: 640/640. In each case, the training examples were chosen at random from the entire data set and the testing set was the remainder. Ten random runs were averaged to obtain the results shown.

It is clear that the rule-based neural network compares with the other classification schemes. It performs about as well as the back-propagation algorithm on all data sets, and succeeds in finding the higher-order representation that the Boolean data set demands and that the first-order Bayes classifier is incapable of generating.

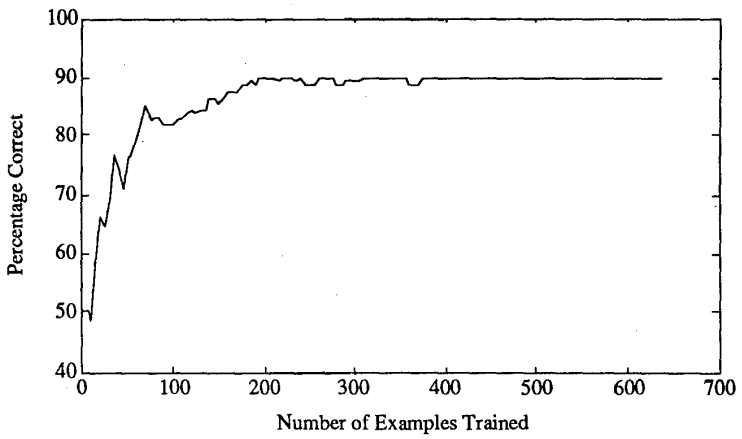

Figure 3: Incremental Performance

4.2 Performance of the Incremental Algorithm

We now show the network's performance and the evolution of its size as it is 'grown' example by example.

For this experiment the Boolean data set described in the previous section was used. While testing on 640 examples, as in the previous section, training examples were given one at a time to the learning algorithm. Classification performance and network size were checked periodically.

The plot of figure 3 shows that the classification performance quickly approaches the optimal classification rate of 90%.

We can see in figure 4 the growth of the network's conjunctive second layer as more examples are presented to it. Notice that the network quickly approaches a size of about 30 units after 200 examples, at the same time that the classification performance reaches 90%. It slowly builds to about 35 units after all 640 examples.

5 Summary and Conclusions

Since it is clear that a number of classifiers achieve near-optimal classification performance, the choice of algorithm should depend upon other factors. We have shown a neural network classifier that is comparable in classification performance with other approaches, yet has several unique advantages.

First, the classifier network may be 'grown' incrementally. The network size and complexity is proportional to the complexity of the problem and the amount that the network has learned. Performance of a network 'grown' incrementally is comparable to that of a network trained on the entire data set at once.

Second, the learning is performed with an algorithm which must terminate within a finite number of steps. This number is on the order of the number of attributes. There are no parameters of learning to 'tweak' in order to obtain the best possible performance.

Third, the rule-based nature of the network may be exploited in explaining a decision of the network. For some applications, the ability to verify the reasoning behind a decision is crucial.

Finally, the algorithm is well suited for hardware implementation. Since each rule is formed independently of all other rules, the rules can be learned in parallel. Currently in progress is a parallel VLSI implementation of the incremental algorithm with binary attributes.

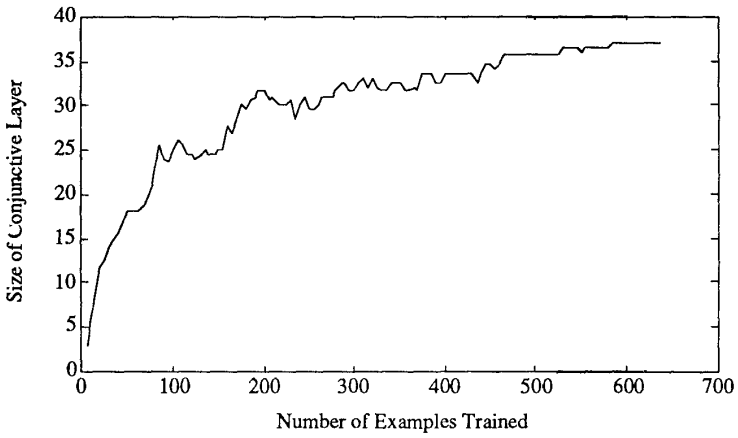

Figure 4: Growing the Network Incrementally

References

[1] A. M. Uttley, 'The Design of Conditional Probability Computers,' *Information and Control* 2, 1-24 (1959).

[2] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Washington, DC: Spartan Books, 1962.

[3] D. Rumelhart, G. Hinton, and R. Williams. 'Learning Internal Representations by Error Propagation.' In D. Rumelhart and J. McLelland, editors, *Parallel Distributed Processing* 1. MIT Press, Cambridge, Mass., 1989.

[4] P. Smyth and R. Goodman, 'An Information-Theoretic Approach to Rule Induction in Databases,' to appear in *IEEE Transactions on Knowledge and Data Engineering.*

[5] N. M. Blachman, 'The amount of information that y gives about X,' *IEEE Transactions on Information Theory*, vol. IT-14(1), 27-31, 1968.

[6] P. Smyth and R. Goodman, 'The Induction of Probabilistic Rule-Sets: the ITRULE Algorithm,' *Proceedings of the 1989 International Workshop on Machine Learning,* Morgan Kaufmann: Palo Alto, CA, pp. 781-787, 1989.

[7] R. Goodman, C. Higgins, and P. Smyth, 'A Hybrid Rule-Based/Bayesian Classifier,' *Proceedings of the 1990 European Conference on Artificial Intelligence*, Stockholm, Sweden, August 1990.

[8] R. Goodman , C. Higgins, J. Miller, and P. Smyth, 'A Rule-Based Approach to Neural Network Classifiers,' *Proceedings of INNC 90 Paris*, International Neural Network Conference, Palais des Congres, Paris, France, July 9-13, 1990.

[9] *Congressional Quarterly Almanac, 98th Congress,* Second session 1984, Congressional Quarterly Inc.: Washington, D.C., 1985.