Neural-Network Feature Selector

Rudy Setiono and Huan Liu

Department of Information Systems and Computer Science

National University of Singapore

Kent Ridge, Singapore 119260

Republic of Singapore

Email:{rudys,liuh}@iscs.nus.sg

Abstract

Feature selection is an integral part of most learning algorithms. Due to the existence of irrelevant and redundant attributes, by selecting only the relevant attributes of the data, higher predictive accuracy can be expected from a machine learning method. In this paper, we propose the use of a three-layer feedforward neural network to select those input attributes that are most useful for discriminating classes in a given set of input patterns. A network pruning algorithm is the foundation of the proposed algorithm. By adding a penalty term to the error function of the network, redundant network connections can be distinguished from those relevant ones by their small weights when the network training process has been completed. A simple criterion to remove an attribute based on the accuracy rate of the network is developed. The network is retrained after removal of an attribute, and the selection process is repeated until no attribute meets the criterion for removal. Our experimental results suggest that the proposed method works very well on a wide variety of classification problems.

Keywords: Feedforward neural network, backpropagation, cross entropy, penalty term, network pruning, feature selection.

I. INTRODUCTION

The problem of pattern recognition can be divided into two stages, feature extraction and classification [1]. Feature extraction refers to the process of finding a mapping that reduces the dimensionality of the patterns. A special case of feature extraction is feature selection. If the characteristics of the patterns are defined by a set of N attributes, we attempt to find a subset of these attributes that are relevant for classification by feature selection. The importance of feature selection is well known. By excluding redundant/irrelevant attributes from the classification process, a classifier with higher generalization capability, i.e., better predictive accuracy on new/unseen patterns can often be found. The dimensionality of patterns with attributes that are highly correlated may be reduced with little or no loss of information.

Hence, by collecting only values of the relevant attributes, the cost of future data collection may also be cut down.

The process of feature selection is often incorporated into the classification process. Classification algorithms that build decision trees such as ID3 [2] and CART [3] select the most suitable attribute at each branching node to grow the decision trees. ID3 finds a branching attribute by computing the information gains of all unselected attributes. The information gained by branching on a particular attribute depends on the number of patterns in each class for each distinct value of the attribute. A different measure, node impurity, is adopted by CART as the criterion for node splitting. A node that mixes patterns of all classes with equal probability has the highest node impurity, while a node that contains patterns of only one class has the lowest node impurity. The attribute for node splitting is chosen such that the resulting descendant nodes have lower node impurity. In general, decision tree methods split a node by using only a single attribute. In some cases, however, it will be more sensible to make use of combinations of attributes.

The approach to feature selection taken by the algorithm proposed in this paper is the opposite of those of CART and ID3. Instead of selecting one attribute at a time, we start with the whole set of attributes and remove the irrelevant attributes one by one. A three-layer feedforward neural network is used as the tool to determine which attributes are to be discarded. The network is trained with the complete set of attributes as input. For each attribute $\mathcal{A}_i$ in the network, we compute the accuracy of the network with all the weights of the connections associated with this attribute set to zero. The attribute that gives the smallest decrease in the network accuracy is removed. The network is then retrained and the process is repeated. To facilitate the process of identifying the irrelevant attributes, the network is trained to minimize an augmented error function. This error function consists of two components. The first component is a measure of network accuracy and the second component is a measure of the network complexity. The accuracy of the network is measured using the cross-entropy error function, while the complexity of the network is measured by a penalty term. A network weight with a small magnitude incurs almost no penalty, while a weight that falls in a certain allowable range incurs an almost constant penalty. The penalty of a large weight that falls outside this interval increases as a quadratic function of its magnitude. The details of this function are presented in Section II. In Section III of this paper, we describe our feature selection algorithm. Experimental results are reported in

Section IV. Finally, a conclusion is given in Section V.

II. NEURAL NETWORK TRAINING

Let us consider the standard fully connected three-layer network depicted in Figure 1. The error measure that we minimize during the training process is the cross-entropy function[4, 5]:

$$F(w,v) = -\left(\sum_{i=1}^{k}\sum_{p=1}^{\mathcal{C}} t_p^i \log S_p^i + (1-t_p^i)\log(1-S_p^i)\right) \tag{1}$$

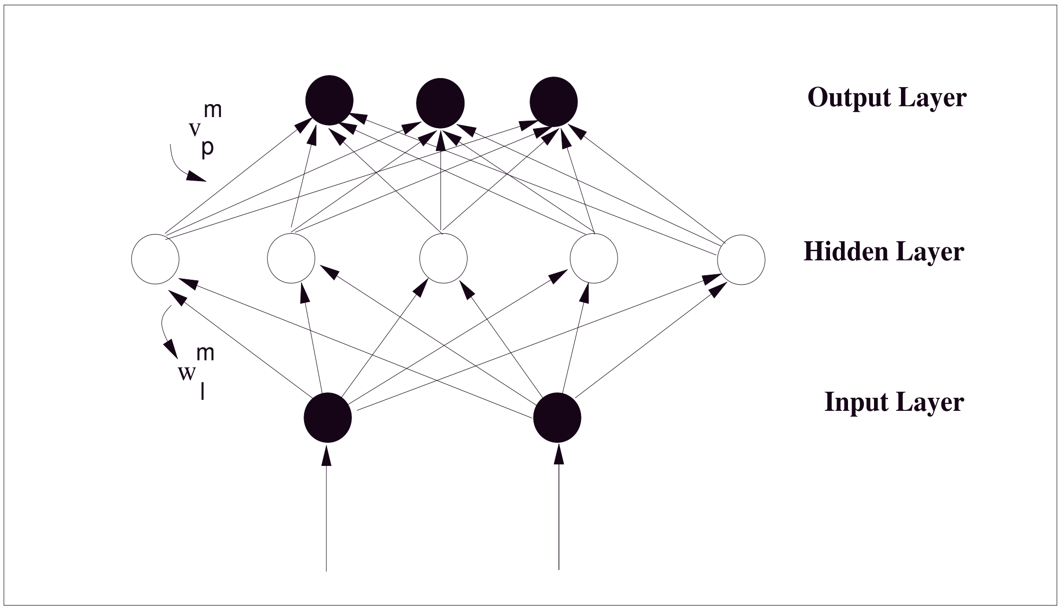

Figure 1: Fully connected feedforward neural network with 5 hidden units and 3 output units.

where

- k is the number of patterns.
- $t_p^i = 0$ or 1 is the target value for pattern x^i at output unit p, $p = 1, 2, \ldots, C$.
- $\mathcal{C}$ is the number of output units.
- S_p^i is the output of the network at unit p.

$$S_p^i = \sigma\left(\sum_{m=1}^{h} \delta\left((x^i)^T w^m\right) v_p^m\right) \tag{2}$$

- x^i is an n-dimensional input pattern, $i = 1, 2, \ldots, k$.
- w^m is an n-dimensional vector of weights for the arcs connecting the input layer and the m-th hidden unit, $m = 1, 2, \ldots, h$.
- v^m is a $\mathcal{C}$-dimensional vector for the arc connecting the m-th hidden unit and the output layer.
- The output unit activation function is the sigmoid function $\sigma(y) = 1/(1 + e^{-y})$.
- The hidden unit activation function is the hyperbolic tangent function $\delta(y) = (e^y - e^{-y})/(e^y + e^{-y})$.

Relevant and irrelevant inputs are distinguished by the strength of their connections from the input layer to the hidden layer in the network. The network is trained such that the connections from the irrelevant inputs to the hidden layer have small magnitude. These connections can be removed from the network without much effect on the network accuracy. Since we are interested in finding the smallest subset of the attributes that still represents the characteristics of the patterns, it is important that the network be trained such that only those connections from the necessary inputs have large magnitude. To achieve this goal, a penalty term is added for each connection from the input layer to the hidden layer of the network.

The penalty function that we use is

$$f(w) = \epsilon_1 \beta w^2 / (1 + \beta w^2) + \epsilon_2 w^2. \tag{3}$$

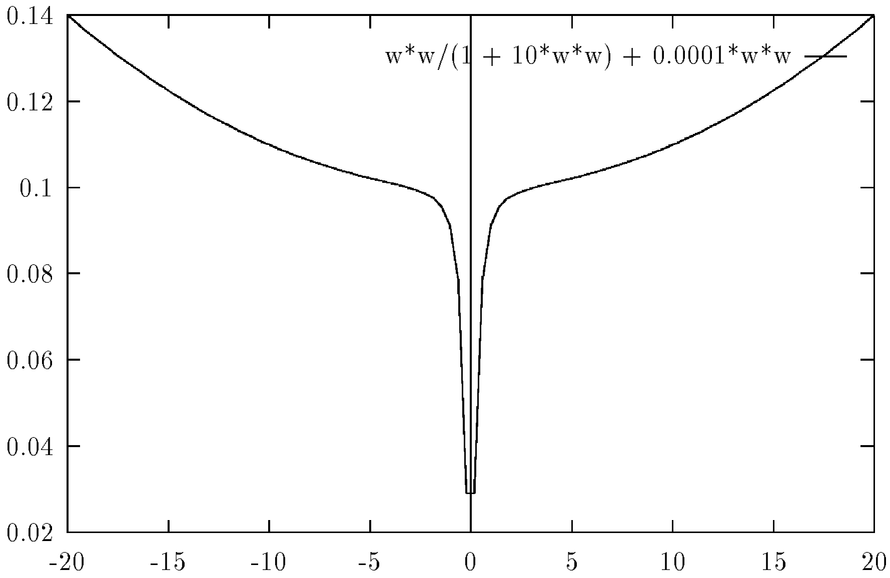

Figure 2: Plot of the function $f(w)$ with $\epsilon_1 = 10^{-1}, \epsilon_2 = 10^{-4}$, and $\beta = 10$.

The plot of the function $f(w)$ with $\epsilon_1 = 10^{-1}, \epsilon_2 = 10^{-4}$, and $\beta = 10$ is shown in Figure 2. The values of the parameters ϵ_1, ϵ_2, and β determine the range over which the value of the function $f(w)$ is approximately equal to ϵ_1. For example, for any weight w whose magnitude is in the interval $(0.95, 10.04)$, the value of the function $f(w)$ is within 10 % of ϵ_1. A weight with small magnitude is encouraged to converge to zero as reflected by the steep drop in the function value near zero. On the other hand, the weights of the network are prevented from taking values that are too large as reflected by the quadratic component of the penalty function which becomes dominant for large values of w. Previously, we have used this penalty function to prune individual connections from a neural network. The detailed description of the network pruning algorithm can be found in [6].

Adding the penalty function to the cross-entropy error function (1), we obtain the following function that is to be minimized during network training:

$$\theta(w, v) = -\left(\sum_{i=1}^{k} \sum_{p=1}^{\mathcal{C}} t_p^i \log S_p^i + (1 - t_p^i) \log(1 - S_p^i) \right) + P(w), \tag{4}$$

where

$$P(w) = \epsilon_1 \left(\sum_{m=1}^{h} \sum_{\ell=1}^{n} \frac{\beta (w_\ell^m)^2}{1 + \beta (w_\ell^m)^2} \right) + \epsilon_2 \left(\sum_{m=1}^{h} \sum_{\ell=1}^{n} (w_\ell^m)^2 \right).$$

In order to reduce the computation time, instead of the standard backpropagation algorithm [7], a variant of the quasi-Newton method is applied to find a local minimum point of the function $\theta(w, v)$. The quasi-Newton algorithm that we use is the BFGS (Broyden-Fletcher-Shanno-Goldfarb) method, which has been shown to be very effective for neural network training by many researchers [8, 9]. At each iteration of the BFGS algorithm, a positive definite matrix that is an approximation of the inverse of the Hessian of the function is computed. A descent direction is obtained by multiplying this matrix with the negative of the gradient of the function at current point. A step-size is computed via an inexact line-search algorithm, and a new approximate solution is obtained by moving along the descent direction with this step-size. Using an appropriately computed step-size, it can be guaranteed that the function value always decreases from one approximate solution to the next. This is one of the main differences between neural network training using the BFGS method and the backpropagation method with a fixed learning rate. The details of the BFGS algorithm for unconstrained minimization can be found in [10].

III. FEATURE SELECTION

Features are selected for removal based on their saliency. Several saliency measures are reported by Belue and Bauer [11]. These measures of saliency of an attribute involve the derivative of the network error function, or the weights of the network, or both. In order to obtain a confidence interval for the mean value of the saliency of the attributes, the network needs to be retrained repeatedly starting from different random weights. It is suggested that the network be trained at least 30 times in order to find a reliable mean and standard deviation of the saliency measure. As training a network by backpropagation algorithm can be very slow, the requirement that the network be trained many times makes their proposed scheme computationally unappealing.

To facilitate the process of identifying the salient features needed for classification, we train the network to minimize the augmented error function (4). With the use of an augmented error function, it is expected that after training, the network will have connections with large magnitude only for those inputs that are needed to represent the patterns' characteristics for classification. Connections with small weights can be excluded from the network with little effect on the network accuracy, that is, either the accuracy is preserved, or if the accuracy drops, it can be recovered by retraining the network. Instead of using a saliency measure that is a function of the network weights, we use a very simple criterion to determine which attribute is to be excluded from the network. This criterion is the network accuracy on the training dataset. Given a trained network with the set of attributes $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N\}$ as its input, we compute the accuracy of the networks having one less attribute, i.e., the set $\mathcal{A} - \{\mathcal{A}_k\}$, for each $k = 1, 2, \ldots N$ is the input attribute set. The accuracy rates are computed by simply setting the connection weights from input attribute $\mathcal{A}_k$ of the trained network to zero. The accuracy rates of these networks are then ranked. Starting with the network having the highest accuracy, the set of attributes to be retained is searched. The steps of the algorithm is outlined below.

Neural-network feature selection algorithm

1. Let $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N\}$ be the set of all input attributes. Separate the patterns into two sets: the training set $\mathcal{S}_1$ and the cross-validation set $\mathcal{S}_2$. Let $\Delta\mathcal{R}$ be the allowable maximum decrease in accuracy rate on the set $\mathcal{S}_2$ and let $\epsilon_1(k)$ and $\epsilon_2(k)$ be the penalty parameters (cf. Eqn. 4) for the connections from input $\mathcal{A}_k$ to the hidden layer, for all $k = 1, 2 \ldots N$.

2. Train network $\mathcal{N}$ to minimize the augmented error function $\theta(w, v)$ with the set $\mathcal{A}$ as input such that it achieves a minimum required accuracy rate on the set $\mathcal{S}_1$. Let $\mathcal{R}^2$ be the accuracy of the network on the set $\mathcal{S}_2$.

3. For all $k = 1, 2, \ldots N$, let $\mathcal{N}_k$ be the network whose weights are set as follows:

   (a) From all inputs except for $\mathcal{A}_k$, set the weights of $\mathcal{N}_k$ equal to the weights of $\mathcal{N}$.

   (b) Set the weights from input $\mathcal{A}_k$ to zero.

   Compute $\mathcal{R}^1_k$ and $\mathcal{R}^2_k$, the accuracy rates of network $\mathcal{N}_k$ on the sets $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively.

4. Rank the networks $\mathcal{N}_k$ according to their accuracy rates: $\mathcal{R}^1_{r(1)} \geq \mathcal{R}^1_{r(2)} \geq \ldots \mathcal{R}^1_{r(N)}$. Let $\mathcal{R}^1_{ave}$ be the average of these rates.

   (a) Set $k = 1$.

   (b) Retrain the network $\mathcal{N}_{r(k)}$.

   (c) Let $\delta = \left(\mathcal{R}^2 - \mathcal{R}^2_{r(k)}\right) / \mathcal{R}^2$.

(d) If $\delta \leq \Delta\mathcal{R}$, then

- Update the penalty parameters for all attributes $j \neq r(k)$:
  - For each input attribute $\mathcal{A}_j$ with network accuracy rate $\mathcal{R}_j^1 \geq \mathcal{R}_{ave}^1$, set $\epsilon_1(j) := 1.1\epsilon_1(j)$ and $\epsilon_2(j) := 1.1\epsilon_2(j)$.
  - For each input attribute $\mathcal{A}_j$ with network accuracy rate $\mathcal{R}_j^1 < \mathcal{R}_{ave}^1$, set $\epsilon_1(j) := \epsilon_1(j)/1.1$ and $\epsilon_2(j) := \epsilon_2(j)/1.1$.
- Reset the input attribute set to $\mathcal{A} - \{\mathcal{A}_{r(k)}\}$, and set $N := N - 1$.
- Set $\mathcal{R}^2 := \max\{\mathcal{R}^2, \mathcal{R}^2_{r(k)}\}$.
- Go to step 3.

(e) If $k < N$, set $k := k + 1$ and go to Step 4(b).
Else stop.

The available patterns for training are divided into two sets, $\mathcal{S}_1$ and $\mathcal{S}_2$. The set $\mathcal{S}_1$ consists of patterns that are actually used to obtain the weights of the neural networks. The set $\mathcal{S}_2$ consists of patterns that are used for cross-validation. By checking the accuracy of the networks on the set $\mathcal{S}_2$, the algorithm decides whether to continue or to stop removing more attributes. The algorithm keeps the best accuracy rate $\mathcal{R}^2$ of the networks on this set. If there is still an attribute that can be removed such that the relative accuracy rate on $\mathcal{S}_2$ does not drop by more than $\Delta\mathcal{R}$, then this attribute will be removed. If there is no such attribute can be found among the inputs, the algorithm terminates.

At the start of the algorithm, the values of the penalty parameters $\epsilon_1(k)$ and $\epsilon_2(k)$ are set equal for all k, since it is not yet known which are the relevant attributes and which are not. After the network is trained, the relative importance of each attribute can be inferred from the accuracy rates of all the networks $\mathcal{N}_k$ having one less attributes. A high accuracy rate of $\mathcal{N}_k$ suggests that the attribute $\mathcal{A}_k$ can be removed from the attribute set. Step 4(d) of the algorithm updates the values of the penalty parameters for all the remaining attributes based on the accuracy of the networks. If the accuracy rate of network $\mathcal{N}_k$ is higher than the average, then the penalty parameters for the network connections from input attribute $\mathcal{A}_k$ are multiplied by a factor 1.1. It is expected that with larger penalty parameters, the connections from this input attribute will have smaller magnitudes after the network is retrained, and therefore the attribute can be removed in the next round of the algorithm. On the other

TABLE I
ATTRIBUTES OF THE MONKS PROBLEMS

A_1:	head_shape	$\in$	round, square, octagon;	A_2:	body_shape	$\in$	round, square, octagon;
A_3:	is_smiling	$\in$	yes, no;	A_4:	holding	$\in$	sword, balloon, flag;
A_5:	jacket_color	$\in$	red, yellow, green, blue;	A_6:	has_tie	$\in$	yes, no.

hand, a below-average accuracy rate of the network $\mathcal{N}_k$ indicates that the attribute $\mathcal{A}_k$ is important for classification. For all such attributes, the penalty parameters are divided by a factor of 1.1.

It is not possible to select the initial values for $\epsilon_1(k)$ and $\epsilon_2(k)$ that work best for all problems. However, through our experiments, we found that the starting values $\epsilon_1(k) = 10^{-1}$ and $\epsilon_2(k) = 10^{-4}$ for all original input attributes $\mathcal{A}_k$ gave very good results for the many problems we tested. These values were used to obtain all the experimental results reported in the next section. The maximum allowable decrease $\Delta\mathcal{R}$ in the accuracy rate on the cross-validation set $\mathcal{S}_2$ was set to 3%.

IV. EXPERIMENTAL RESULTS

We report our experimental results to select features for classification problems in this section. The problems that we selected have been widely tested by other researchers and they include both real-world problems and artificially created problems. Two sets of artificial problems were tested. They were the monks problems [12] and a subset of the data mining problems generated by Agrawal *et al.* [13] of IBM Almaden Research Center. Four real-world problems were also tested, they originated in diverse fields: breast cancer diagnosis, the US Congressional voting records, diabetes diagnosis, and sonar returns classification. Data for these four problems as well as for the monks problems can be obtained via anonymous ftp from the University of California-Irvine repository [14].

A. The monks problems.

The monks problems [12] are an artificial robot domain, in which robots are described by six different attributes (Table I):

The learning tasks of the three monks problems are of binary classification, each of them is given by the following logical description of a class.

- Problem Monks 1: (head_shape = body_shape) or (jacket_color = red). From 432 possible samples, 112 were randomly selected for the training set, 12 for cross-validation, and all 432 for testing.

- Problem Monks 2: Exactly two of the six attributes have their first value. From 432 samples, 152 were selected randomly for the training set, 17 for cross-validation, and all 432 for testing.

- Problem Monks 3: (Jacket_color is green and holding a sword) or (jacket_color is not blue and body_shape is no octagon). From 432 samples, 122 were selected randomly for training and among them there were 5% misclassifications, i.e., noise in the training set. Twenty samples were selected for cross-validation, and all 432 samples formed the testing set.

In order to demonstrate the effectiveness of the feature selection algorithm, we use each possible values of the 6 attributes as a single new attribute. For example, the attribute head_shape which can be either round, square, or octagon, is represented by three new attributes. The three attributes are head_shape=round, head_shape=square and head_shape=octagon. For any pattern, two of these three attributes must have a value of 0, while the third attribute has a value of 1. This representation of the original 6 attributes enables us to select not only the relevant attributes, but also to discover which particular values of these attributes are useful for classification.

For each problem, thirty neural networks with 12 hidden units and 17 input units were trained starting from different initial random weights. The results of the experiments are summarized in Table II. In this table, we give the average accuracy of the networks on the training and testing datasets with and without feature selection. Standard deviations are given in parentheses. The average function evaluation reflects the cost of selecting the relevant features. In our implementation of the BFGS algorithm, each time the value of the error function is computed, its gradient is also computed. Hence, the average number of function evaluations is also the average number of gradient evaluations. The P-value is computed to check if there is any significant increase in the accuracy of the networks with selected input features compared to the networks with the whole set of attributes as input. A smaller P-value indicates a more significant increase.

TABLE II

RESULTS FOR THE THREE MONKS PROBLEMS. FIGURES IN PARENTHESES INDICATE STANDARD DEVIATIONS.

Monks 1		
	With all features	**With selected features**
Ave. no. of features	17 (0.00)	5.07 (0.37)
Ave. acc. on training set (%)	100.00 (0.00)	100.00 (0.00)
Ave. acc. on testing set (%)	99.71 (0.67)	100.00 (0.00)
Ave. function evaluations	360.37 (114.76)	
P-value (testing set acc.)	0.09	
Monks 2		
	With all features	**With selected features**
Ave. no. of features	17 (0.00)	6.23 (0.43)
Ave. acc. on training set (%)	100.00 (0.00)	100.00 (0.00)
Ave. acc. on testing set (%)	98.78 (2.34)	99.54 (0.99)
Ave. function evaluations	538.63 (117.02)	
P-value (testing set acc.)	0.05	
Monks 3		
	With all features	**With selected features**
Ave. no. of features	17 (0.00)	3.87 (1.78)
Ave. acc. on training set (%)	100.00 (0.00)	94.23 (0.79)
Ave. acc. on testing set (%)	93.55 (1.41)	98.41 (1.66)
Ave. function evaluations	826.70 (212.86)	
P-value (testing set acc.)	$< 10^{-5}$	

TABLE III
ATTRIBUTES OF THE IBM DATASETS

Attribute	Description	Value
salary	salary	uniformly distributed from 20,000 to 150,000
commission	commission	if salary $\geq$ 75000 $\rightarrow$ commission = 0
		else uniformly distributed from 10000 to 75000.
age	age	uniformly distributed from 20 to 80.
elevel	education level	uniformly distributed from $[0, 1, \ldots, 4]$.
car	make of the car	uniformly distributed from $[1, 2, \ldots 20]$.
zipcode	zip code of the town	uniformly chosen from 9 available zipcodes.
hvalue	value of the house	uniformly distributed from $0.5k10000$ to $1.5k1000000$
		where $k \in \{0 \ldots 9\}$ depends on zipcode.
hyears	years house owned	uniformly distributed from $[1, 2, \ldots, 30]$.
loan	total amount of loan	uniformly distributed from 1 to 500000.

The figures in Table II show that feature selection improves significantly the predictive accuracy of the networks. Very good results are obtained for all three problems. For the Monks 1 problem, all 30 networks with selected input attributes are capable of classifying all testing patterns correctly. Twenty nine networks have the minimum 5 input attributes and the remaining one has 7 input attributes. For the Monks 2 problem, 23 networks have the minimum 6 attributes and the remaining 7 networks have 7 attributes.

For the Monks 3 problem, most networks have either 2 or 5 input attributes. The maximum number of attributes a network has is 9. All twelve networks with 5 input attributes achieve 100 % accuracy rate on the testing dataset. All eleven networks with 2 input attributes have an accuracy rates of 93.44 % and 97.22 % on the training dataset and the testing dataset, respectively. The 97.22 % accuracy rate is the same as that reported by Thrun *et al.* [12]. It is worth noting that, despite the presence of 6 mislabeled training patterns, 14 of the 30 networks with selected attributes have a perfect 100 % accuracy rate on the testing dataset. None of the 30 networks with all input attributes has such accuracy.

B. IBM datasets.

These datasets had been generated by Agrawal *et al.* [13] to test their database mining algorithm $\mathcal{CDP}$. Every pattern of the datasets consists of nine attributes given in Table III. One network input unit was assigned for each of these attributes.

Ten binary classification functions were developed using these attributes. We selected 3

TABLE IV
IBM CLASSIFICATION FUNCTIONS

Function 1	
Group A:	$((\mathbf{age} < 40) \wedge (50000 \leq \mathbf{salary} \leq 100000))\vee$
	$((40 \leq \mathbf{age} < 60) \wedge (75000 \leq \mathbf{salary} \leq 125000))\vee$
	$((\mathbf{age} \geq 60) \wedge (25000 \leq \mathbf{salary} \leq 75000))$.
Group B:	Otherwise.
Function 2	
Group A:	$((\mathbf{age} < 40)\wedge$
	$(((\mathbf{elevel} \in [0\ldots2]\ ?\ (25000 \leq \mathbf{salary} \leq 75000)) : (50000 \leq \mathbf{salary} \leq 100000))))\vee$
	$((40 \leq \mathbf{age} < 60)\wedge$
	$(((\mathbf{elevel} \in [1\ldots3]\ ?\ (50000 \leq \mathbf{salary} \leq 100000)) : (75000 \leq \mathbf{salary} \leq 125000))))\vee$
	$((\mathbf{age} \geq 60)\wedge$
	$(((\mathbf{elevel} \in [2\ldots4]\ ?\ (50000 \leq \mathbf{salary} \leq 100000)) : (25000 \leq \mathbf{salary} \leq 75000))))$
Group B:	Otherwise.
Function 3	
Group A:	$\mathbf{disposable} > 0$, where
	$\mathbf{disposable} = (0.67 \times (\mathbf{salary} + \mathbf{commission}) - 5000 \times \mathbf{elevel} - 0.2 \times \mathbf{loan} - 10000)$
Group B:	Otherwise.

of the 10 functions to test our feature selection algorithm. The 3 functions selected involved different kinds of input attributes with different complexities. The definitions of the functions are given in Table IV.

Similar to the three Monks problems, we also trained 30 networks each with 12 hidden units for Functions 1, 2, and 3. The number of input units was 9. The number of patterns used for training, cross validation, and testing was 800, 200 and 1000, respectively. These patterns were randomly generated according to the distributions described in Table III. Following Agrawal *et al.* [13], we also included a perturbation factor as one of the parameters of the random data generator. The perturbation factor was set at 5 percent. The results of the feature selection algorithm are presented in Table V.

The results of the algorithm on the three functions also show significant improvement in the accuracy of the networks with selected input features. The effectiveness the feature selection algorithm is shown by the small number of attributes selected. For all three functions, the average number of attributes selected by the algorithm is close to the minimum number

TABLE V

RESULTS FOR THE IBM CLASSIFICATION FUNCTIONS

Function 1		
	With all features	**With selected features**
Ave. no. of features	9 (0.00)	2.47 (0.51)
Ave. acc. on training set (%)	98.39 (1.59)	94.20 (1.65)
Ave. acc. on testing set (%)	88.86 (2.49)	93.13 (1.86)
Ave. function evaluations	2733.23 (927.90)	
P-value (testing set acc.)	$< 10^{-5}$	
Function 2		
	With all features	**With selected features**
Ave. no. of features	9 (0.00)	3.10 (0.92)
Ave. acc. on training set (%)	95.73 (2.52)	90.03 (2.46)
Ave. acc. on testing set (%)	83.41 (1.74)	87.16 (1.85)
Ave. function evaluations	2921.13 (945.88)	
P-value (testing set acc.)	$< 10^{-5}$	
Function 3		
	With all features	**With selected features**
Ave. no. of features	9 (0.00)	4.00 (0.00)
Ave. acc. on training set (%)	100.00 (0.00)	99.75 (0.12)
Ave. acc. on testing set (%)	98.09 (0.34)	98.20 (0.35)
Ave. function evaluations	1209.80 (317.05)	
P-value (testing set acc.)	0.11	

necessary. For function 3, all 30 networks select the correct set of 4 attributes that defined the function. For function 1, the average number of selected features is 2.47 which is about 25 % higher than the the number of attributes involved in the function. Sixteen networks selected the correct set of input attributes, i.e., age and salary. The remaining 14 networks, however, still have the attribute commission as the third input. For function 2, 23 of the networks select the correct set of relevant input attributes. The number of attributes of the remaining 7 networks is either 2, 4, 5 or 7.

C. Real-world datasets.

Four real-world datasets are chosen to test our feature selection algorithm. These datasets are described below.

1. The University of Wisconsin Breast Cancer Diagnosis Dataset.

   The Wisconsin Breast Cancer Data (WBCD) is a large data set that consists of 699 patterns of which 458 are benign samples and 241 are malignant samples. Each of these patterns consists of nine measurements taken from fine needle aspirates from a patient's breast [15]. The measurements were graded 1 to 10 at the time of sample collection, with 1 being the closest to benign and 10 the most anaplastic. A linear programming based method for pattern separation called the Multisurface Method has been proposed by Mangasarian [16]. A computer program that implements this method for the WBCD has been in use at the University of Wisconsin Hospital since 1990 [17]. For our experiment, 315 samples were randomly selected for training, 35 samples were selected for cross-validation, and 349 for testing.

2. United States Congressional Voting Records Dataset.

   The dataset consists of the voting records of 435 congressmen on 16 major issues in the 98th Congress. The votes are classified into one of the three different types of votes: yea, nay, and unknown. The classification problem is to predict the party affiliation of each congressman, which is either democrat or republican. We selected 197 patterns for training randomly, 21 patterns for cross-validation, and 217 patterns for testing. Schlimmer [18] reported getting an accuracy rate of 90%-95% on this dataset.

3. Pima Indians Diabetes Dataset.

   The dataset consists of 768 samples taken from patients who may show signs of diabetes.

Each sample is described by 8 attributes, 1 attribute has discrete values and the rest have continuous values. The training set consists of 345 randomly selected samples, the cross-validation set consists of 39 samples, and the testing set consists of the remaining 384 samples. Applying the ADAP algorithm trained on 576 samples, Smith *et al.* [19] achieved an accuracy rate of 76 % on the remaining 192 samples.

4. Sonar Targets Dataset.

   The sonar returns classification dataset [20] consists of 208 sonar returns, each of which is represented by 60 real numbers between 0.0 and 1.0. The task is to distinguish between returns from a metal cylinder and those from a cylindrically shaped rock. Using 104 returns as training samples and 104 returns as testing samples, Gorman and Sejnowski trained backpropagation networks with hidden units ranging from 0 to 24 [20]. The accuracy rates reported on the testing set ranged between 73.1% and 90.4%. For our experiment, we have used 90 returns for training, 14 returns for cross-validation, and 104 returns for testing.

For each of the four datasets, 30 neural networks with 12 hidden units were trained. The results are summarized in Table VI. The performance of the networks with selected input features is consistently better than the performance of the networks with all attributes as input. Although the percentage differences in the accuracy of the networks on the testing sets are small, except for the breast cancer dataset, they are statistically significant. The results of the experiments on these datasets show that, for these problems it is possible to achieve similar or better predictive accuracy with much less number of attributes. The number of selected input attributes ranges from about one-third of the original set of attributes for the breast cancer dataset to less than 7 % for the sonar returns dataset.

V. SUMMARY

Feature selection is a very important aspect of solving the problem of pattern classification. Many collected datasets contain attributes that are redundant or irrelevant. The advantages of using only the relevant features of the data for classification are many. First, by reducing data-overfitting, a classifier with better predictive accuracy can be obtained. Second, by identifying the relevant features, the cost of future data collection can be reduced. Third, by excluding the irrelevant attributes, a simpler classifier can be obtained and the time required

TABLE VI
RESULTS FOR 4 REAL-WORLD DATASETS

Wisconsin Breast Cancer Dataset		
	With all features	**With selected features**
Ave. no. of features	9 (0.00)	2.70 (1.02)
Ave. acc. on training set (%)	100.00 (0.00)	98.05 (1.31)
Ave. acc. on testing set (%)	93.94 (0.94)	94.15 (1.00)
Ave. function evaluations	1401.73 (569.70)	
P-value (testing set acc.)	0.20	
US Congressional Voting Records		
	With all features	**With selected features**
Ave. no. of features	16 (0.00)	2.03 (0.18)
Ave. acc. on training set (%)	100.00 (0.00)	95.63 (0.43)
Ave. acc. on testing set (%)	92.00 (0.96)	94.79 (1.60)
Ave. function evaluations	818.43 (194.24)	
P-value (testing set acc.)	$< 10^{-5}$	
Pima Indians Diabetes Dataset		
	With all features	**With selected features**
Ave. no. of features	8 (0.00)	2.03 (0.18)
Ave. acc. on training set (%)	93.59 (2.77)	74.02 (6.00)
Ave. acc. on testing set (%)	71.03 (1.74)	74.29 (3.25)
Ave. function evaluations	4749.93 (2289.54)	
P-value (testing set acc.)	$< 10^{-5}$	
Sonar Targets Dataset		
	With all features	**With selected features**
Ave. no. of features	60 (0.00)	3.87 (0.82)
Ave. acc. on training set (%)	100.00 (0.00)	98.33 (3.01)
Ave. acc. on testing set (%)	92.34 (1.17)	93.81 (2.82)
Ave. function evaluations	2915.50 (338.28)	
P-value (testing set acc.)	0.0041	

to classify new patterns can be reduced.

We have presented our algorithm for feature selection using neural networks. The two main components of the algorithm are an augmented error function for neural network training and an efficient quasi-Newton algorithm for minimizing the error function. When a network is trained to minimize the augmented error function, relevant and irrelevant attributes are distinguished by the magnitude of the weights of their respective connections from the input layer to the hidden layer. Irrelevant attributes are identified by the small magnitude of their connection weights. The quasi-Newton method is used to speed up both the training and retraining of the network. We have tested the algorithm on 6 artificially generated and 4 real-world datasets. For the real-world datasets, the algorithm removed a large number of attributes from the original attribute sets and improved the predictive accuracy of the neural networks. For the artificial datasets, the algorithm picked only the relevant input attributes in most of the experiments conducted. Our algorithm provides a different approach than those of decision tree methods and has been shown to be very effective for selecting the relevant input attributes for classification purpose.

REFERENCES

[1] T.Y. Young and T.W. Calvert, *Classification, Estimation and Pattern Recognition.* Elsevier, New York, 1974.

[2] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, California, 1993.

[3] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, "Classification and regression trees," Wadsworth & Brooks, California, 1984.

[4] K.J. Lang and M.J. Witbrock, "Learning to tell two spirals apart", in *Proc. of the 1988 Connectionist Model Summer School*, D. Touretzky, G. Hinton & T. Sejnowski (Eds), Morgan Kaufmann, San Mateo, CA., 1988, pp. 52-59.

[5] A. van Ooyen and B. Nienhuis, "Improving the convergence of the backpropagation algorithm", *Neural Networks*, vol. 5, pp. 465-471, 1992.

[6] R. Setiono, "A penalty-function approach for pruning feedforward neural networks", *Neural Computation*, forthcoming.

[7] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing*, volume 1. The MIT Press, Cambridge, Mass., 1986.

[8] R.L. Watrous, "Learning algorithms for connectionist networks: applied gradient methods for nonlinear optimization," in *Proc. IEEE First International Conference on Neural Networks*, San Diego, 1987, pp. 619-627.

[9] R. Setiono, "A neural network construction algorithm which maximizes the likelihood function," *Connection Science*, vol. 7, no. 2, pp. 147-166, 1995.

[10] J.E. Dennis Jr. and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[11] L.M. Belue and K.W. Bauer, Jr., "Determining input features for multilayer perceptron," *Neurocomputing*, vol. 7, no. 2, pp. 111-122, 1995.

[12] S.B. Thrun *et al.* , "The MONK's Problems - A performance comparison of different learning algorithms", Department of Computer Science, Carnegie Mellon University, CMU-CS-91-197, 1991.

[13] R. Agrawal, T. Imielinski and A. Swami. "Database mining: A performance perspective", *IEEE Trans. on Knowledge and Data Engineering*, vol. 5, no. 6, Dec. 1993, pp. 914-925.

[14] P.M. Murphy and D.W. Aha, UCI Repository of machine learning databases, Machine-readable data repository, Irvine, CA. University of California, Dept. of Information and Computer Science, 1992.

[15] W.H. Wolberg and O.L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc. of the National Academy of Sciences*, vol. 87, pp. 9193-9196, 1990.

[16] O.L. Mangasarian, "Multisurface method of pattern separation," *IEEE Transactions on Information Theory,* vol. 14, no. 6, pp. 801-807, 1968.

[17] O.L. Mangasarian, R. Setiono and W.H. Wolberg, "Pattern recognition via linear programming: theory and application to medical diagnosis," in: T.F. Coleman and Y. Li. eds., *Large-scale Numerical Optimization,* (SIAM, Philadelphia, 1990), pp. 22-30.

[18] J.C. Schlimmer, "Concept acquisition through representational adjustment," Ph.D. thesis, Dept. of Information and Computer Science, University of California, Irvine, CA., 1987.

[19] J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," in *Proc. of the Symposium on Computer Applications and Medical Care*, IEEE Computer Society Press, 1988, pp. 261-265.

[20] R.P. Gorman and T.J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar target," *Neural Networks*, vol. 1, pp. 75-89, 1988.