

Feedforward neural network construction using cross-validation

Rudy Setiono

School of Computing

National University of Singapore

3 Science Drive 2, Singapore 117543

Email: rudys@comp.nus.edu.sg

Abstract

This paper presents an algorithm which constructs feedforward neural networks with a single hidden layer for pattern classification. The algorithm starts with a small number of hidden units in the network and adds more hidden units as needed to improve the network's predictive accuracy. To determine when to stop adding new hidden units, the algorithm makes use of a subset of the available training samples for cross-validation. New hidden units are added to the network only if they improve the classification accuracy of the network on the training samples and on the cross-validation samples. Extensive experimental results show that the algorithm is effective in obtaining networks with predictive accuracy rates that are better than those obtained by state of the art decision tree methods.

Keywords: Feedforward neural networks, network construction, decision tree methods, conjugate gradient method.

1 Introduction

When applying the neural network approach to solve a pattern classification problem, one is faced with the question of what kind of network topology is most suitable for this problem. To address this question of model selection, in the past years numerous algorithms which automatically construct neural networks have appeared in the literature. The algorithms include the cascade correlation algorithm (Fahlman and Lebiere 1990), the tiling algorithm (Mezard and Nadal 1989), the self-organizing neural network (Tenorio and Lee 1990), the MPyramid-real and the MTiling-real algorithms (Parekh *et al.* 2000). For a given problem, these algorithms will generally build networks with many layers of processing units.

The upstart algorithm (Frean 1990) is an algorithm which builds a network with layers of linear threshold units to learn boolean classification on patterns of binary variables. The generated network, however, can be converted into a network with one hidden layer by placing all the generated units in a new hidden layer and connecting them to a newly created output unit. Networks with a single hidden layer have been shown to be universal function approximators (Hornik *et al.* 1989). It is also relatively easier to extract symbolic rules from such networks by applying the decompositional approach (Tickle *et al.* 1998) than to extract rules from networks with many layers of hidden units. Among the algorithms which construct networks with a single hidden layer are the DNC (Dynamic Node Creation) method (Ash 1989), FNNCA (Feedforward Neural Network Construction Algorithm) (Setiono 1996) and CARVE (Constructive Algorithm for Real-Valued Examples) (Young and Downs 1998). These algorithms create a feedforward neural network by sequentially adding hidden units to its hidden layer. The initial number of hidden units in the network is usually one or two. The network is trained to minimize its classification errors on the training samples. If the trained network does not achieve the required accuracy rate, one or more hidden units is added to the network and the network is retrained. The process is repeated until a network that

correctly classifies all the input samples or meets some other prespecified stopping criteria has been constructed.

FNNCA and CARVE algorithms produce networks that correctly classify all the training samples, while DNC may terminate without obtaining such networks. For general problems, however, it is usually not desirable to run any network construction algorithm until all the training samples have been correctly classified. Insisting on 100% correct classification on the training samples usually means generating networks with too many parameters. Such networks can be expected to overfit the data and do not generalize well. Theoretical results (Baum and Haussler 1989) and empirical results (Setiono and Liu 1997; Treadgold and Gedeon 1999) indicate that smaller networks generalize better than larger ones.

In this paper, we present a neural network construction algorithm that makes use of cross-validation or hold-out samples to determine when to stop adding hidden units to the network. Our approach is similar to that taken by Treadgold and Gedeon (1999). New nodes are added to the growing network only if they improve the accuracy of the network on the training and cross-validation samples. By checking the network accuracy on these samples, the requirement to specify a preset minimum accuracy rate for the growing network is eliminated.

The network constructed by the proposed algorithm is the standard feedforward neural network with a single hidden layer. The growing network is trained using the conjugate gradient minimization algorithm. A very important criterion when choosing an optimization algorithm to minimize the errors of the growing network is the convergence speed of the algorithm, especially when large problems with many inputs and/or outputs are to be solved. The conjugate gradient algorithm converges faster than first-order methods such as back-propagation method and its memory storage requirement is much less than second-order methods such as the Newton's method.

While there are numerous network construction algorithms described in the literature,

most of them have been tested only on a small number of data sets and their performance have not been compared against other methods for classification. We test our algorithm on 32 publicly available data sets and compare our results with those obtained by state-of-the-art methods which generate decision trees. We find that neural networks give statistically better predictive accuracy than the decision trees for many of the test problems.

The outline of this paper is as follows. The Neural Network Construction with Cross-validation Samples (N2C2S) algorithm is described in Section 2 of this paper. The results of our experiments are presented in Section 3. Finally, we conclude the paper in Section 4.

2 The N2C2S algorithm

We assume that we have P data samples $(\mathbf{x}_p, \mathbf{y}_p), p = 1, 2, \dots, P$ where input $\mathbf{x}_p \in \mathbb{R}^N$ and target $\mathbf{y}_p \in [0, 1]^M$, N is the dimensionality of the input data, and M is the number of classes. The proposed algorithm requires that this data set be split randomly into two disjoint subsets: the training (\mathcal{T}) and the cross-validation (\mathcal{C}) sets. The samples in \mathcal{T} determine the optimal weights of the connections of the growing network, while the samples in \mathcal{C} help to determine the topology of the final network.

The outline of the network construction algorithm is as follows.

Algorithm N2C2S (Neural Network Construction with Cross-validation Samples)

Input:

\mathcal{T} : A set of training samples $(\mathbf{x}_p, \mathbf{y}_p), p = 1, 2, \dots, P_1$, and

\mathcal{C} : A set of cross-validation samples $(\mathbf{x}_p, \mathbf{y}_p), p = 1, 2, \dots, P_2$.

Objective: A feedforward neural network with good generalization capability.

Step 1. Let \mathcal{N}_1 be a network with N input units, M output units, and H hidden units.

Step 2. Initialize the connection weights of \mathcal{N}_1 randomly and train this network to minimize an error function. Let the accuracy rates of the trained networks on the sets \mathcal{T} and \mathcal{C} be $\mathcal{A}_{\mathcal{T}_1}$ and $\mathcal{A}_{\mathcal{C}_1}$, respectively.

Step 3. Let \mathcal{N}_2 be a network with N input units, M output units, and $H + h$ hidden units.

Step 4. Set the weights of the connections to and from the first H hidden units of \mathcal{N}_2 to the optimal weights of \mathcal{N}_1 and set the remaining connection weights of \mathcal{N}_2 randomly. Train \mathcal{N}_2 and let its accuracy rates on the sets \mathcal{T} and \mathcal{C} be $\mathcal{A}_{\mathcal{T}_2}$ and $\mathcal{A}_{\mathcal{C}_2}$, respectively.

Step 5a. If $(\mathcal{A}_{\mathcal{T}_2} + \mathcal{A}_{\mathcal{C}_2}) > (\mathcal{A}_{\mathcal{T}_1} + \mathcal{A}_{\mathcal{C}_1})$, then

1. Set $H := H + h$.
2. Let $\mathcal{N}_1 := \mathcal{N}_2$, $\mathcal{A}_{\mathcal{T}_1} := \mathcal{A}_{\mathcal{T}_2}$, $\mathcal{A}_{\mathcal{C}_1} := \mathcal{A}_{\mathcal{C}_2}$.
3. If $H < MaxH$, go to Step 3.

Step 5b. Else

1. Train a new network \mathcal{N}_3 with $H + h$ hidden units and with all its initial weights assigned randomly and let its accuracy rates on the sets \mathcal{T} and \mathcal{C} be $\mathcal{A}_{\mathcal{T}_3}$ and $\mathcal{A}_{\mathcal{C}_3}$, respectively.
2. If $(\mathcal{A}_{\mathcal{T}_3} + \mathcal{A}_{\mathcal{C}_3}) > (\mathcal{A}_{\mathcal{T}_1} + \mathcal{A}_{\mathcal{C}_1})$, then
 - (a) Set $H := H + h$.
 - (b) Let $\mathcal{N}_1 := \mathcal{N}_3$, $\mathcal{A}_{\mathcal{T}_1} := \mathcal{A}_{\mathcal{T}_3}$, $\mathcal{A}_{\mathcal{C}_1} := \mathcal{A}_{\mathcal{C}_3}$.
 - (c) If $H < MaxH$, go to Step 3.

Step 6 Output \mathcal{N}_1 as the final constructed network.

The algorithm adds nodes to the growing network as long as the addition of the nodes improves the accuracy of the network. As a measure of accuracy, we use the total classification accuracy of the network on the training and cross-validation sets. For some problems,

it would be reasonable to check the accuracy of the growing network on the cross-validation samples only. However, for small data sets with many classes and/or uneven distribution of the classes, we find that using only the relatively few samples available for cross-validation leads to final networks that generalize poorly. The addition of new hidden units usually results in a higher network accuracy on the training samples. However, when there are already sufficient hidden units in the network, this increase is usually accompanied by a decrease in the cross-validation accuracy. The new hidden units are added to the growing network only if they improve the accuracy of the network on both sets of samples or if the increase in the training accuracy is greater than the decrease in the cross-validation accuracy.

Two networks with different number of hidden units \mathcal{N}_1 and \mathcal{N}_2 are trained and their accuracy rates are compared so that we do not have to determine a prespecified accuracy level to terminate the network construction algorithm. The difficulty associated with a fixed prespecified accuracy level is obvious. When the required accuracy on the training samples is set too high, the network will have too many hidden units and it will overfit the data. This, in turn, usually leads to poor prediction of new unseen samples. On the other hand, if the prespecified accuracy level is set too low, the algorithm would terminate prematurely with a small network that cannot predict well when given new samples. When the accuracy of the growing network on a set of cross-validation samples is checked, we usually do not know the level of accuracy that we can expect. Hence, the proposed algorithm attempts to overcome this difficulty by comparing the accuracy of two successive networks generated along the way.

When h new hidden units are added to the trained network \mathcal{N}_1 in Step 3 to form the larger network \mathcal{N}_2 , the optimal weights of network \mathcal{N}_1 are used as the initial weights for the connections to and from the first H hidden units of \mathcal{N}_2 . By making use of the optimal weights of the smaller network, we hope to reduce the training time of the new network. A third network \mathcal{N}_3 with the same number of hidden units as \mathcal{N}_2 is trained when the accuracy

of \mathcal{N}_2 is not better than the accuracy of the smaller network \mathcal{N}_1 in Step 5b. The accuracy of \mathcal{N}_2 may not be better than the accuracy of \mathcal{N}_1 because the conjugate gradient algorithm used to minimize the error function is trapped at a local minimum point of the weight space. In order to reduce the chances of getting trapped at such a local minimum, the algorithm trains \mathcal{N}_3 which has been initialized with a completely random set of weights. The construction algorithm terminates only if this new network does not predict better than the smaller network \mathcal{N}_1 either.

We explain now the error function that is minimized during the network construction process. Let \mathbf{w} be the weights of the network connections from the input units to the hidden units and \mathbf{v} be the weights of the connections from the hidden units to the output units. The error measure that we have adopted is the standard sum of squared errors $E(\mathbf{w}, \mathbf{v})$ augmented with a penalty term $\theta(\mathbf{w}, \mathbf{v})$:

$$E(\mathbf{w}, \mathbf{v}) = \sum_{p=1}^{P_1} \sum_{\ell=1}^M (\tilde{y}_{p\ell} - y_{p\ell})^2 + \theta(\mathbf{w}, \mathbf{v}) \quad (1)$$

The penalty term $\theta(\mathbf{w}, \mathbf{v})$ is defined as follows (Setiono 1997):

$$\theta(\mathbf{w}, \mathbf{v}) = \epsilon_1 \left(\sum_{i=1}^H \sum_{j=1}^N \frac{\beta w_{ij}^2}{1 + \beta w_{ij}^2} + \sum_{i=1}^H \sum_{\ell=1}^M \frac{\beta v_{\ell i}^2}{1 + \beta v_{\ell i}^2} \right) + \epsilon_2 \left(\sum_{i=1}^H \sum_{j=1}^N w_{ij}^2 + \sum_{i=1}^H \sum_{\ell=1}^M v_{\ell i}^2 \right) \quad (2)$$

where

- $y_{p\ell} \in [0, 1]$ is the target value for input \mathbf{x}_p at output unit ℓ ,
- $\tilde{y}_{p\ell}$ is the predicted value for input \mathbf{x}_p at output unit ℓ ,
- $\epsilon_1, \epsilon_2, \beta$ are positive penalty parameters,
- w_{ij} is the weight of the connections from input unit j to hidden unit i ,
- $v_{\ell i}$ is the weight of the connection from hidden unit i to the output unit ℓ ,

- the hidden unit activation value A_{ip} for input \mathbf{x}_p and the predicted value $\tilde{\mathbf{y}}_p$ for this input are computed as follows:

$$A_{ip} = \tanh\left(\sum_{j=1}^N w_{ij}x_{jp}\right), \quad (3)$$

$$\tilde{y}_{p\ell} = \sum_{i=1}^H v_{\ell i}A_{ip}. \quad (4)$$

The transfer function that we have used to compute the hidden unit activation of the network is the hyperbolic tangent function $\tanh(\xi) = (e^\xi - e^{-\xi})/(e^\xi + e^{-\xi})$.

The penalty term $\theta(\mathbf{w}, \mathbf{v})$ is added to the error function to increase the chances of constructing the smallest networks with the best generalization ability. Extensive empirical experiments indicate that the addition of a penalty term during network construction improves the generalization ability of the resulting networks (Treadgold and Gedeon 1999). Among the different penalty terms employed, the cubic penalty term was found to be the most beneficial for network generalization. Instead of the cubic penalty term, we have opted to incorporate the penalty term $\theta(\mathbf{w}, \mathbf{v})$ that we have used in conjunction with a neural network pruning algorithm (Setiono 1997). Employing this penalty term enabled us to obtain pruned networks that are smaller than the networks obtained by other network pruning algorithms.

A local minimum of the error function $E(\mathbf{w}, \mathbf{v})$ can be obtained by applying any nonlinear optimization methods such as the gradient descent method or a higher order method. In our implementation, we have used the conjugate gradient algorithm CONMIN implemented by Shanno and Phua (1976, 1980). Their implementation of the conjugate gradient method with line search ensures that after every iteration of the method, the error function value will decrease. This is a property of the method that is not possessed by the standard backpropagation method with a fixed learning rate. While the conjugate gradient method does not converge as fast as the quasi-Newton methods, its memory storage requirement is less. For an unconstrained minimization problem with n variables, the conjugate gradient

option of CONMIN requires $7n+2$ double-precision words of working storage, while a variant of the quasi-Newton methods, the BFGS method, requires $n^2/2 + 11n/2$ words (Shanno and Phua 1980). As some data sets can have both high-dimensional inputs and many classes, the total number of weights in the growing network easily exceeds one thousand, making the BFGS method prohibitively expensive in terms of memory storage requirement.

3 Experimental results

Thirty-two classification problems were solved using the proposed algorithm. These are real-world classification problems with mixed discrete and continuous attributes. The data sets for these problems are available from the website of the Machine Learning Research Group at the Department of Computer Science, University of Waikato ¹. They are also available from the UCI Machine Learning Repository (Blake and Merz 1998). The size of the data sets and the characteristics of the input attributes of the data are summarized in Table 1.

The performance of many new network learning and construction algorithms has been evaluated on only a relatively few problems. This is despite some criticism about this practice (Prechelt 1996). We have selected the classification problems listed in Table 1 because the data sets are available publicly and the results from experiments using decision tree methods are also available (Frank *et al.* 1997). Hence performance comparison can be made between the neural network approach and the decision tree methods.

For each data set, the experimental setting was as follows:

1. Ten-fold cross-validation scheme: we split each data set randomly into 10 subsets of equal size. Eight subsets were used for training, one subset was used for cross-validating, and one subset for measuring the predictive accuracy of the final constructed network. This procedure was performed 10 times so that each subset was tested once.

¹<http://www.cs.waikato.ac.nz/~ml/weka/index.html>

Test results were averaged over 10 ten-fold cross-validation runs. Data splitting was done without sampling stratification.

2. The same set of penalty parameter values in the penalty term (Eqn. 2) was used: $\epsilon_1 = 0.5$, $\epsilon_2 = 0.05$ and $\beta = 0.1$.
3. The starting number of hidden units was 2. The numbers of input units and output units are shown in Table 1. The number of input units includes one unit with a constant input value of one to implement hidden unit bias. The number of output units corresponds to the number of classes in the data. The *winner-takes-all* strategy for prediction is used.
4. Two hidden units, instead of one, were added at one time to the growing network (Step 3 of the algorithm). This is to reduce the training time for problems that require large number of hidden units. The maximum number hidden units allowed $MaxH$ was 20.
5. During network training, the conjugate gradient algorithm was terminated (i.e. the network was assumed to have been trained) if the relative decrease in the error function value (Eqn. 1) after two consecutive iterations was less than 10^{-5} or the maximum number of function evaluations was reached. This number was set to 2500.
6. One network input unit was assigned to each continuous attribute in the data set. Nominal attributes were binary coded. A nominal attribute with D possible values was assigned D network inputs. A binary attribute with no missing value required one network input unit, while a binary attribute with missing value required 2 input units (See 8 below on how the missing values were handled).
7. Continuous attribute values were scaled to range in the interval $[0, 1]$, while binary-encoded attribute values were either 0 or 0.2. We found that the 0/0.2 encoding produced better generalization than the usual 0/1 encoding.

8. A missing continuous attribute value was replaced by the average of the non-missing values. A missing discrete attribute value was assigned the value “unknown” and the corresponding components of the input \mathbf{x} were set to the zero.

The experimental results are presented in Table 2. In this table, we show the average number of hidden units and the average accuracy rates of the constructed networks on the training data set, the cross-validation data set and the test set. The average number of hidden units ranged from 2.46 for the labor data set to 19.44 for the soybean data set. Most of the networks for the latter data set contain the maximum 20 hidden units. It might be possible to improve the overall predictive accuracy of these networks if we had allowed them to have more than 20 hidden units. This was not done, however, because we attempted to solve all the problems using the same set of values for all the parameters in the algorithm. A general picture of the network accuracy on the training data set, cross-validation data set and the test set can be seen from the figures in the table. The accuracy on the training data set is higher than the accuracy on the cross-validation set, which is in turn higher than the predictive accuracy on the test set.

How accurate are the network predictions compared to those from other methods? To answer this question, we reproduce the results obtained by Frank et al. (1997) from their method M5'. M5' is a reimplementation of M5 algorithm developed by Quinlan for learning from data sets with continuous classes (Quinlan 1992). These algorithm generate binary decision trees with linear regression function at the leaf nodes. Since both algorithms are designed for predictions of continuous classes, they can be used for classification problems by applying them to approximate the conditional class probability function. For a problem having N classes, N model trees are generated. Each of these trees approximates the conditional probability of a given sample belonging to class $C_i, i = 1, 2, \dots, N$. The class of the input sample is determined by the model tree which gives the highest probability value.

Frank et al. compared their results to those of C5.0, a successor to Quinlan's widely used C4.5 (Quinlan 1993) decision tree method for classification. Table 3 summarizes the results from the neural networks, M5' and C5.0. The results of M5' and C5.0 are the averages and standard deviations from 10 runs of ten fold cross-validation experiments. We computed the standard error of the difference between the averages of the neural networks and those of the other method. The t statistic for testing the null hypothesis that the two means are equal was then obtained and a two-tailed test was conducted. If the null hypothesis was rejected at the significance level of $\alpha = .01$, we checked which method has the higher average accuracy rate. In Table 2, a neural network win is denoted by a bullet (\bullet), while a neural network loss is denoted by a diamond (\diamond). Averages with no significant difference are left unmarked.

Table 4 provides the summary of the comparison. For exactly half of the 32 data sets tested, neural networks outperform C5.0. For the other 16 problems, C5.0 is more accurate than neural networks on 7 data sets. On the rest of the data sets, there are no significant differences in the accuracy rates of the two methods. Compared to M5', neural networks are significantly better on 13 data sets and significantly worse on only 7 data sets. There are no significant differences in the predictive accuracy of the two methods on remaining 12 data sets.

4 Conclusion and discussion

We have presented an algorithm for constructing feedforward neural networks using cross-validation samples. This algorithm eliminates the difficulty associated with having to determine the number of hidden units before network training begins. While there are numerous network construction algorithms in the literature, most of them have been tested on only a small number of data sets. We tested and compared the results of our network construction

algorithm on 32 publicly available data sets with mixed continuous and discrete attributes. Our results show that compared to the decision tree methods, neural networks can provide better predictions for many of these data sets.

There are several parameters that need to be set in the network construction algorithm such as the penalty parameters, the starting number of hidden units, the maximum number of hidden units, and the maximum number of iterations/epochs allowed during training/retraining of the growing network. Fine tuning the values of these parameters for individual problem is very likely to improve the accuracy of the constructed networks. With the goal of having a single set of parameter values, we have fixed the parameter values as described in the previous section. They are found after an extensive experimentation to find a combination that gives the best overall performance on the 32 test data sets. The results show that with this fixed set of parameters values, the constructed networks can outperform state-of-the-art methods that generate decision trees.

Future work to improve the accuracy of the constructed networks includes

1. checking for possible removal of hidden units from the constructed networks. An algorithm which adds and then removes the hidden units was shown to be effective in reducing the overall training time of backpropagation neural networks (Hirose et al. 1991). The effect of such removal on the network's generalization, however, remains to be investigated.
2. incorporation of a pruning algorithm to remove redundant attributes of the data.
3. modification of the algorithm so that it will be able to select different penalty terms and/or different values for the penalty parameters automatically depending on the characteristics of the data set.
4. better handling of data samples with missing attribute values.

Acknowledgment

This work was done while the author was spending his sabbatical leave at the Computational Intelligence Lab, University of Louisville, Kentucky. He is grateful to Professor Jacek M. Zurada of the Department of Electrical and Computer Engineering, University of Louisville for providing office space and super-computer access.

References

- Ash, T. 1989. Dynamic node creation in backpropagation networks. *Connection Science* **1**(4), 365–375.
- Baum, E. and Haussler, D. 1989. What size net gives valid generalization? *Neural Computation* **1**, 151–160.
- Blake, C.L. and Merz, C.J. 1998. UCI Repository of machine learning databases (<http://www.ics.uci.edu/~mlearn/MLRepository.html>). Irvine, CA: University of California, Department of Information and Computer Science.
- Fahlman, S.E. and Lebiere, C. 1990. The cascade-correlation learning architecture, in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky, Ed., pp. 524–532. Morgan Kaufmann, San Mateo, CA.
- Frank, E., Wang, Y., Inglis, S., Holmes, G., and Witten, I.H. 1997. Using model trees for classification. *Machine Learning* **32**(1), 63–76.
- Frean, M. 1990. The upstart algorithm: a method for constructing and training feedforward neural networks. *Neural Computation* **2**, 198–209.
- Hirose, Y., Yamashita, K. and Hijiya, S. 1991. Back-propagation algorithm which varies the number of hidden units. *Neural Networks* **4**, 61–66.

- Hornik, K., Stinchcombe, M. and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* **2**, 359–366.
- Mezard, M. and Nadal, J.P. 1989. Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A* **22**(12), 2191–2203.
- Parekh, R., Yang, J., and Honavar, V. 2000. Self-organizing network for optimum supervised learning. *IEEE Trans. on Neural Networks* **11**(2), 436–451.
- Prechelt, L. 1996. A quantitative study of experimental evaluation of neural network learning algorithms. *Neural Networks* **9**, 457–462.
- Quinlan, R. 1992. Learning with continuous classes. In *Proc. of the Australian Joint Conference on Artificial Intelligence*, World Scientific Publishing, Singapore, pp. 343–348.
- Quinlan, R. 1993. *C4.5: Programs for machine learning*. Morgan Kaufman, San Mateo, CA.
- Setiono, R. 1996. A neural network construction algorithm which maximizes the likelihood function. *Connection Science* **7**(2), 147–166.
- Setiono, R. 1997. A penalty-function approach for pruning feedforward neural networks. *Neural Computation* **9**(1), 185–204.
- Setiono, R. and Liu, H. 1997. Neural network feature selector. *IEEE Transactions on Neural Networks* **8**(3), 654–662.
- Shanno, D.F. and Phua, K.H. 1976. Algorithm 500. Minimization of unconstrained multivariate functions. *ACM Transactions on Mathematical Software* **2**(1), 87–94.
- Shanno, D.F. and Phua, K.H. 1980. Remark on Algorithm 500. *ACM Transactions on Mathematical Software* **6**(4), 618–622.

- Tenorio, M.F. and Lee, W. 1990. Self-organizing network for optimum supervised learning. *IEEE Transactions on Neural Networks* **1** (1), 100–110.
- Tickle, A.B., Andrews, R., Golea, M., and Diederich, J. 1998. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *IEEE Transactions on Neural Networks* **9**(6), 1057–1068.
- Treadgold, N.K. and Gedeon, T.D. 1999. “Exploring constructive cascade networks. *IEEE Transactions on Neural Networks* **10**(6), 1335–1350.
- Young, S. and Downs, T. 1998. CARVE - A constructive algorithm for real-valued examples. *IEEE Transactions on Neural Networks* **9**(6), 1180–1190.

Table 1: Data sets used in the experiments

Data set	Size	Missing Values (%)	Attributes			Neural Network	
			Continuous	Binary	Nominal	Inputs	Outputs
anneal	898	0.0	6	14	18	82	5
audiology	226	2.0	0	61	8	95	24
australian	690	0.6	6	4	5	45	2
autos	205	1.1	15	4	6	73	6
balance-scale	625	0.0	4	0	0	5	3
breast-cancer	286	0.3	0	3	6	51	2
breast-w	699	0.3	9	0	0	10	2
german	1000	0.0	6	3	4	62	2
glass	214	0.0	9	0	0	10	6
glass (G2)	163	0.0	9	0	0	10	2
heart-c	302	0.2	6	3	4	23	2
heart-h	294	20.4	6	3	4	25	2
heart-statlog	270	0.0	13	0	0	14	2
hepatitis	155	5.6	6	13	0	30	2
horse-colic	368	23.8	7	2	13	62	2
hypothyroid	3772	5.5	7	20	2	34	4
ionosphere	351	0.0	33	1	0	35	2
iris	150	0.0	4	0	0	5	3
kr-vs-kp	3196	0.0	0	34	2	41	2
labor	57	3.9	8	3	5	30	2
lymphography	148	0.0	0	9	6	39	4
pima-indians	768	0.0	8	0	0	9	2
primary-tumor	339	3.9	0	14	3	37	21
segment	2310	0.0	19	0	0	20	7
sick	3772	5.5	7	20	2	33	2
sonar	208	0.0	60	0	0	61	2
soybean	683	9.8	0	16	19	135	19
vehicle	846	0.0	18	0	0	19	4
vote	435	5.6	0	16	0	33	2
vowel	990	0.0	10	2	1	28	11
waveform-noise	5000	0.0	40	0	0	41	3
zoo	101	0.0	1	15	0	17	7

Table 2: The number of hidden units and the accuracy rates of the neural networks constructed by the N2C2S algorithm. Figures shown are the averages and the standard deviations from 10 ten-fold cross-validation runs.

Data set	Hidden units	Training acc. (%)	Cross-val acc. (%)	Test acc. (%)
anneal	6.58 ± 0.29	99.65 ± 0.06	99.45 ± 0.24	99.41 ± 0.17
audiology	17.18 ± 1.21	95.51 ± 1.14	80.47 ± 1.65	79.50 ± 1.61
australian	7.22 ± 1.38	92.26 ± 0.71	86.90 ± 0.62	84.83 ± 0.71
autos	8.38 ± 1.01	97.78 ± 0.57	78.73 ± 2.23	72.31 ± 1.83
balance-scale	10.12 ± 1.03	93.76 ± 0.51	93.38 ± 0.54	92.32 ± 0.89
breast-cancer	6.06 ± 0.84	91.86 ± 0.96	74.46 ± 2.24	67.80 ± 2.17
breast-w	3.40 ± 0.51	97.47 ± 0.06	96.94 ± 0.27	96.58 ± 0.24
german	9.54 ± 1.43	96.56 ± 1.27	74.00 ± 1.14	70.10 ± 1.65
glass	9.58 ± 1.41	79.88 ± 1.63	71.72 ± 1.10	65.55 ± 3.00
glass (G2)	7.06 ± 0.84	89.98 ± 1.10	81.90 ± 3.43	77.91 ± 2.50
heart-c	5.60 ± 1.13	92.36 ± 0.98	85.60 ± 0.74	82.47 ± 1.95
heart-h	5.94 ± 1.14	90.17 ± 0.92	84.96 ± 1.20	81.63 ± 1.60
heart-statlog	5.52 ± 0.99	94.08 ± 1.35	83.44 ± 1.08	77.56 ± 1.00
hepatitis	5.66 ± 1.16	97.49 ± 1.22	86.75 ± 2.08	81.94 ± 3.34
horse-colic	4.88 ± 0.38	98.47 ± 0.52	83.20 ± 1.27	78.97 ± 1.29
hypothyroid	10.02 ± 1.52	97.24 ± 0.11	96.95 ± 0.11	96.82 ± 0.11
ionosphere	4.20 ± 0.87	99.20 ± 0.15	92.65 ± 1.16	89.52 ± 2.26
iris	3.34 ± 0.49	98.31 ± 0.14	97.40 ± 0.49	96.60 ± 0.66
kr-vs-kp	7.20 ± 0.71	99.57 ± 0.11	99.40 ± 0.22	99.28 ± 0.18
labor	2.46 ± 0.40	99.98 ± 0.07	92.63 ± 2.11	91.90 ± 4.10
lymphography	6.34 ± 0.84	96.92 ± 0.47	85.22 ± 1.82	82.97 ± 1.97
pima-indians	5.88 ± 0.99	80.43 ± 0.38	78.49 ± 0.62	76.04 ± 0.86
primary-tumor	11.38 ± 1.38	62.09 ± 1.41	50.28 ± 0.96	45.97 ± 1.37
segment	14.68 ± 0.70	95.99 ± 0.26	95.52 ± 0.21	95.19 ± 0.43
sick	4.80 ± 0.57	98.03 ± 0.03	97.72 ± 0.09	97.56 ± 0.11
sonar	5.18 ± 0.48	99.92 ± 0.14	82.42 ± 2.44	76.51 ± 1.56
soybean	19.44 ± 0.70	96.37 ± 0.96	93.28 ± 0.67	92.97 ± 0.72
vehicle	13.26 ± 1.13	91.15 ± 0.43	86.17 ± 0.82	84.29 ± 0.79
vote	4.42 ± 0.61	98.55 ± 0.13	96.64 ± 0.48	96.09 ± 0.48
vowel	17.44 ± 1.38	94.25 ± 1.23	90.56 ± 1.17	88.86 ± 1.46
wave	10.82 ± 2.61	89.71 ± 0.89	86.63 ± 0.39	85.66 ± 0.25
zoo	12.12 ± 0.10	100.00 ± 0.00	94.75 ± 1.42	94.34 ± 1.46

Table 3: Comparison of the accuracy rates of neural networks, C5.0 and M5'. A bullet (●) indicates that the accuracy of neural networks is significantly higher than that of the other method, while a diamond (◇) indicates that the accuracy of the neural networks is significantly lower.

Data set	Neural networks	C5.0	M5'
anneal	99.41 ± 0.17	98.7 ± 0.3 ●	98.8 ± 0.2 ●
audiology	79.50 ± 1.61	76.5 ± 1.4 ●	76.7 ± 1.0 ●
australian	84.83 ± 0.71	85.3 ± 0.5	85.8 ± 0.9
autos	72.31 ± 1.83	80.0 ± 2.5 ◇	74.4 ± 1.9
balance-scale	92.32 ± 0.89	77.6 ± 1.0 ●	86.4 ± 0.7 ●
breast-cancer	67.80 ± 2.17	73.3 ± 1.6 ◇	69.6 ± 2.3
breast-w	96.58 ± 0.24	94.5 ± 0.3 ●	95.3 ± 0.3 ●
german	70.10 ± 1.65	71.2 ± 1.0	72.9 ± 0.7 ◇
glass	65.55 ± 3.00	67.5 ± 2.6	70.5 ± 2.8 ◇
glass (G2)	77.91 ± 2.50	78.7 ± 2.1	81.8 ± 2.2 ◇
heart-c	82.47 ± 1.95	76.8 ± 1.4 ●	80.9 ± 1.4
heart-h	81.63 ± 1.60	79.8 ± 0.9 ●	79.0 ± 0.8 ●
heart-statlog	77.56 ± 1.00	78.7 ± 1.4	82.2 ± 1.0 ◇
hepatitis	81.94 ± 3.34	79.3 ± 1.2	81.9 ± 2.2
horse-colic	78.97 ± 1.29	85.3 ± 0.6 ◇	84.6 ± 0.7 ◇
hypothyroid	96.82 ± 0.11	99.5 ± 0.0 ◇	96.6 ± 0.1 ●
ionosphere	89.52 ± 2.26	88.9 ± 1.6	89.7 ± 1.2
iris	96.60 ± 0.66	94.5 ± 0.7 ●	94.7 ± 0.7 ●
kr-vs-kp	99.28 ± 0.18	99.5 ± 0.1 ◇	99.4 ± 0.1
labor	91.90 ± 4.10	78.1 ± 4.8 ●	79.7 ± 4.6 ●
lymphography	82.97 ± 1.97	75.4 ± 2.8 ●	79.8 ± 1.4 ●
pima-indians	76.04 ± 0.86	74.5 ± 1.2 ●	76.2 ± 0.8
primary-tumor	45.97 ± 1.37	41.8 ± 1.3 ●	45.1 ± 1.6
segment	95.19 ± 0.43	96.8 ± 0.2 ◇	97.0 ± 0.2 ◇
sick	97.56 ± 0.11	98.8 ± 0.1 ◇	98.3 ± 0.1 ◇
sonar	76.51 ± 1.56	74.7 ± 2.8	78.5 ± 3.4
soybean	92.97 ± 0.72	91.3 ± 0.5 ●	92.5 ± 0.5
vehicle	84.29 ± 0.79	72.9 ± 1.2 ●	76.5 ± 1.3 ●
vote	96.09 ± 0.48	96.3 ± 0.6	96.2 ± 0.3
vowel	88.86 ± 1.46	79.8 ± 1.3 ●	81.7 ± 1.1 ●
wave	85.66 ± 0.25	75.4 ± 0.5 ●	82.0 ± 0.2 ●
zoo	94.34 ± 1.46	91.8 ± 1.1 ●	92.1 ± 1.3 ●

Table 4: Summary of the results from neural networks compared to those from C5.0 and M5'.

NN versus	Wins (●)	Ties	Losses (◇)
C5.0	16	9	7
M5'	13	12	7