ANN-DT: An Algorithm for Extraction of Decision Trees from Artificial Neural Networks

Gregor P. J. Schmitz, Chris Aldrich, and Francois S. Gouws

***Abstract*— Although artificial neural networks can represent a variety of complex systems with a high degree of accuracy, these connectionist models are difficult to interpret. This significantly limits the applicability of neural networks in practice, especially where a premium is placed on the comprehensibility or reliability of systems. A novel artificial neural-network decision tree algorithm (ANN-DT) is therefore proposed, which extracts binary decision trees from a trained neural network. The ANN-DT algorithm uses the neural network to generate outputs for samples interpolated from the training data set. In contrast to existing techniques, ANN-DT can extract rules from feedforward neural networks with continuous outputs. These rules are extracted from the neural network without making assumptions about the internal structure of the neural network or the features of the data. A novel attribute selection criterion based on a significance analysis of the variables on the neural-network output is examined. It is shown to have significant benefits in certain cases when compared with the standard criteria of minimum weighted variance over the branches. In three case studies the ANN-DT algorithm compared favorably with CART, a standard decision tree algorithm.**

***Index Terms*—Decision trees, hybrid systems, induction, neural networks, rule extraction, sensitivity analysis.**

I. Introduction

DURING the last decade interest in artificial neural networks has grown significantly, owing to their ability to represent nonlinear relationships that are difficult to model by means of other computational methods. Moreover, neural networks are easy to implement, are robust under the influence of noise, do not require *a priori* knowledge with regard to the distributions of data, and can be parallelized where rapid computation is critical.

However, but for the simplest structures, neural-network models are notoriously difficult to interpret. For example, the fact that neural networks have large degrees of freedom in the assignment of weights, can lead to a situation where two completely different sets of weights can yield nearly identical outputs. This drastically complicates the analysis and comparison of similar processes that are modeled or controlled by different neural networks. The opacity of neural networks can be seen as a major barrier to their implementation in a number of fields, such as medicine and engineering where mission critical applications demand a high degree of confidence in the behavior of relevant models.

Manuscript received January 22, 1997; revised July 28, 1998 and February 9, 1999.

The authors are with the Department of Chemical Engineering, University of Stellenbosch, Matieland 7602, South Africa.

Publisher Item Identifier S 1045-9227(99)08887-6.

In order to overcome this limitation, various attempts have previously been made to extract rules from neural networks. Most of these techniques require special training methods and architectures for neural networks, or are based on assumptions that tend to restrict the ability of the neural network to generalize the underlying relationships in the data.

A more general algorithm not subject to these limitations is therefore proposed in this paper. More specifically, this algorithm does not depend on any assumptions with regard to the structure of the neural network or the input–output data and enables the characterization of the behavior of the neural network by means of a set of heuristic rules, similar to those obtained by means of other rule induction algorithms such as ID3 [1], [2], C4.5 [3], or CART [4]. Neural networks are generally better at approximating complex relationships for problems with predominantly continuous inputs. Therefore the rules extracted from the network not only clarify the neural-network model, but in some cases are also significantly more accurate than those derived by other machine learning methods, such as the aforementioned algorithms.

II. Extraction of Rules from Neural Networks

The methods that are used to extract rules from neural networks can be categorized as *decompositional*, *pedagogical*, and *eclectic* [5], [6], based on the approach used to characterize the internal model of the network. With pedagogical techniques the neural network is treated as a "black box," i.e., rules that map inputs to outputs are extracted directly, even for multilayered neural networks. For example, Thrun's method [11], [12] of validity-interval analysis (VIA) uses linear programming to determine if a set of constraints placed on a network's activation values is consistent. Instead of approximating the activation levels of the hidden units as threshold functions, the levels are assumed to be independent of one another. Since this assumption is not always appropriate, the algorithm does not always find maximally general rules. Other methods such as those developed by Craven and Shavlik [6] and Pop *et al.* [13] are currently limited to discrete outputs. Furthermore, Craven and Shavlik's technique is implemented using either the VI analysis [12] or the KT method described by Fu [8], whereby it is implicitly assumed that the activations of the hidden layers are independent, or that they can be treated as threshold functions.

Saito and Nakano [14] have also pursued a pedagogical concept by searching for combinations of input values which activate given output units. Unfortunately, this approach resulted in a search space that grew exponentially with the

number of input variables, and limited the number of rules that could be explored. Narazaki *et al.* [15] recently followed a similar approach, where regions were defined, the boundaries of which were based on the signs of the partial derivatives of the functional relationship defined by the neural network, as well as on the class predicted by the neural network.

In contrast with pedagogical techniques, decompositional strategies [7]–[9] focus on the separate extraction of rules from each unit of the neural network. The activation of the individual neurons are approximated in terms of threshold functions, based on the assumption that the units are either maximally active or inactive. This allows each noninput unit in the neural network to be interpreted as a step function or a Boolean rule, so that the rule extraction problem is reduced to finding situations in which certain rules are valid. Apart from the risk of inaccurate representation of the neural network, the architecture and training procedures usually need to be adapted to conform more closely to this approximation. Moreover, hard thresholding of the units can make them sensitive to noise and minor variations in the data. Despite the application of remedial measures, such as soft thresholding [10], these and other similar methods constrain the distributed nondiscrete internal states of a neural network and prevent the use of many advanced network implementations.

Eclectic techniques combine elements of the two basic approaches discussed above. These methods draw inferences from the magnitudes of the weights in a neural network, as exemplified by the DEDEC [16] and BRAINNEE systems [17]. This has to be done carefully, as certain weights in a neural network can be large, but nevertheless insignificant to the final outcome, owing to cancellations of their contributions in higher layers of the network. Moreover, interdependent inputs tend to complicate this problem even further [18].

The ANN-DT (artificial neural-network decision tree) algorithm described in this paper can be seen as pedagogical, although it is not bound by the limitations of the above strategies. Two variants of the algorithm were investigated, which differ only with regard to the way in which attributes and associated split points are selected. The proposed techniques are based on the sampling of a neural-network representation of the original data, instead of attempting to extract rules from these data or interpreting the internal structure of the neural network. The idea of sampling the neural network was already introduced by Craven and Shavlik [6] for discrete inputs and outputs and was extended by Craven and Shavlik's TREPAN [19] algorithm for problems with continuous and/or discrete inputs. The TREPAN algorithm, which is limited to discrete outputs, grows nodes in a best first order [19] and uses the greedy gain ratio criterion [3] to evaluate M-of-N splits. Unlike the TREPAN algorithm, ANN-DT can be applied to data sets where both the inputs and outputs can assume discrete or continuous values. Since an earlier version of the ANN-DT algorithm restricted to discrete output data has been published previously [20], [21], this paper focuses specifically on systems with continuous output variables. Moreover it is shown that for problems in which look-ahead criteria are required, the novel significance analysis proposed in this paper can have appreciable advantages over greedy attribute selection criteria.

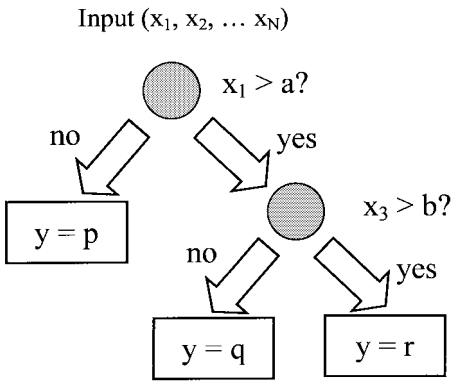


Fig. 1. A simple univariate binary decision tree.

III. ANN-DT Algorithm

As is shown in Fig. 1, the ANN-DT algorithm generates a univariate decision tree from a trained neural network. A given decision node in the tree returns "true" if the node's single specified variable exceeds a certain threshold and "false" otherwise. For each node the algorithm decides on which variable to partition the set of data, after which the threshold of that variable has to be determined. More specifically, the univariate decision tree is constructed by use of the algorithm to examine the responses of the neural network in the feature space and to conduct a sensitivity or significance analysis of the different attributes or explanatory variables pertaining to these responses.

A. Training of the Artificial Neural Networks

The ANN-DT algorithm is depicted schematically in Fig. 2. In the first stage of the procedure, a neural-network model capable of generalizing underlying trends in the data has to be constructed. In this particular investigation both multilayer perceptron [22] and radial basis function neural networks were used. The former was trained by a gradient descent method [23], in which the error was propagated backward through the network [24]. Training the radial basis function network [25], [26] entails finding a suitable set of basis nodes and weights to the output layer. Since the connecting weights of the output layer of the radial basis function neural network can be found by solution of a least squares problem, the most difficult task is finding a good set of basis functions. In this investigation an evolutionary algorithm [28] was used to find the optimal set of basis nodes, instead of more frequently used methods such as k-means clustering algorithms [27].

B. Induction of Rules from Sampled Points in the Feature Space

A systematic search through the input or feature space, as is accomplished by the methods of Fu [8], Gallant [9], Thrun [11], [12], and Saito and Nakano [14], can lead to an overwhelming number of rules for many real-world problems. Even when the feature space is properly bounded, decision boundaries generated by the neural network which are not parallel to the boundaries of the feature space can lead to an intractably large number of rules required to represent the

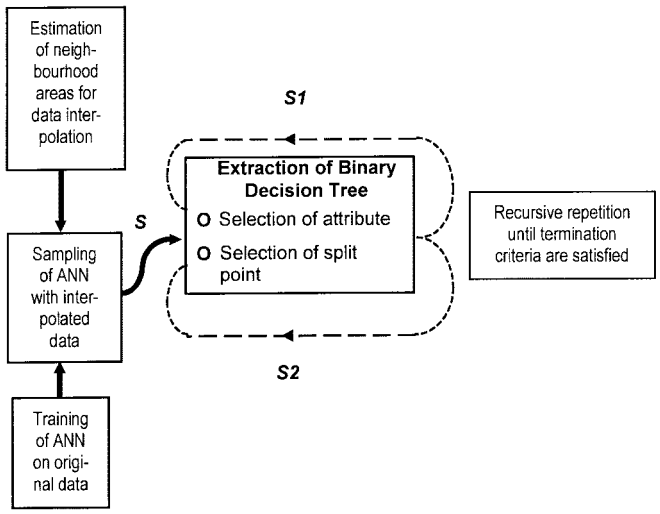


Fig. 2. A diagrammatic representation of the algorithm to extract rules from an artificial neural network.

behavior of the network to some arbitrary degree of accuracy. For example, Thrun [11] has investigated the movement of a robotic arm that has required more than 8000 rules to describe. Consequently restrictions to the depths of these kinds of searches have to be imposed. Even when the depths of these searches are restricted, the number of rules can still be large, as was suggested by a study of Saito and Nakano [14]. They have found that more than 400 rules were extracted for just one output unit in a neural network with 23 possible outputs. As a consequence it is sometimes necessary to increase the granularity of the rules in order to improve their intelligibility. This can be done by either limiting the depth to which the rules should be induced or by pruning the rules afterwards as will be discussed later.

The ANN-DT algorithm uses sampling, which can similarly lead to numerous rules depending on the number of sample points that are generated. However, it is necessary to ensure that the sampled points are restricted to those regions of the (often high-dimensional) search space on which the neural-network model is based. This is accomplished by taking the joint distributions of points in the feature space into account, not only during sampling of the network, but also when the resulting branches of the trees require pruning.

C. Interpolation of Correlated Data

If the neural network constitutes a generalized representation of a given set of data, a new artificial data set with approximately the same distribution as the original data can subsequently be generated. This is accomplished by randomly sampling the input (feature) space, and computing the target values for these sampled points by means of the trained neural network.

In order to ensure that the newly generated data are representative of the original training data set and not only of the particular neural-network model, it is essential that sampling is only allowed in the neighborhood of points or clusters present in the training data set. This can be accomplished by using a nearest neighbor method, in which the distance of a sampled point to the nearest point in the training data set is calculated. If this distance exceeds a predetermined critical value, the sampled point is deemed unlikely to be typical of the data in the training set and is discarded.

For continuous variables the normal Euclidean distance was used as a distance measure. For discrete data other measures of similarity can be defined [29], while the Euclidean distance or a metric defined by Gower [30] can also be used for mixed data. In cases where there are few discrete variables, the data sets belonging to the various possible combinations of the discrete variables can be treated separately [29].

In this investigation the critical distance was based on the distances between points in the training data set. In relatively small data sets with n data points, all $n(n-1)/2$ distances can be calculated. In larger data sets this becomes computationally expensive, in which case a random sample of a number of interpoint distances can be used. The critical distance was subsequently set equal to the average distance between the points and their respective ($k = 10$) nearest neighbors in the training data set. Each artificially generated exemplar is consequently presented to the neural network, from which a corresponding output value is computed.

D. Selection of Attribute and Threshold for Splitting

1) Weighted Variance Minimization: If a set S consists of n exemplars with n corresponding output values, an input value and a threshold are required in order to split the data set into two sets S_1 and S_2. For a least squares error measure the attribute and threshold are selected which cause the maximum decrease in the *weighted variance* V_w over the two branches [4]

$$V_w = \sum_{k=1}^{2} \frac{n_k}{n} \operatorname{Var}(O_k) \tag{1}$$

where O_k are the outputs of the data set S_k.

In this way the sum of the squared errors of the newly formed branches is reduced with each split [4]. This is the same procedure as is used in the CART algorithm when forming a regression tree.

2) Analysis of Attribute Significance: Attributes can also be selected by using an alternative method to those discussed. With this approach the significance of the various inputs on the behavior of the neural network is evaluated. Consider a neural-network model or functional relationship f between attributes (inputs) and classes (outputs) that is evaluated at a set of points S lying inside a domain D. If the magnitudes of the partial derivatives of the function with respect to the inputs are to be a measure of the significance, it is implicitly assumed that the variables can change freely and independently from one another. For the analysis of data where the influencing factors can be varied individually, this assumption is valid. However, if the measured attributes are correlated this is not appropriate as far as the system represented by the neural network is concerned, as a change in one input feature may be accompanied by a change in another covariant feature.

These interrelationships need to be taken into account by focusing on the variations of f that actually occur inside

the domain D. This can be done by taking the variation of f into account, when moving between points in S. Define the *absolute variation* $v(f)$ of the function $f(\mathbf{x})$ between the points i and j as the absolute value of the directional derivative of $f(\mathbf{x})$ integrated along a straight line between the two points. Thus

$$v_{ij}(f) = \int_{x_i}^{x_j} |\Delta f(x) \cdot u| \, dx \tag{2}$$

where $\mathbf{u}$ is the unity vector in direction $\mathbf{x}_i - \mathbf{x}_j$.

This variation can be computed between all pairs of points in S. When an attribute is insignificant to the function for the domain D, the variation in the function will be unrelated to the variation in the attribute. Note that for a function where the effect of one attribute is cancelled out by another covariant attribute [e.g., $f(\mathbf{x}) = x_1 - x_2 + \cos\{x_3\}$ and a domain in which $x_1 \approx x_2$] variations in the other attributes only (in this case x_3) will cause notable variations in $f(\mathbf{x})$. Therefore variations in the attributes with more influence (x_3) will correlate with the absolute variations in $f(\mathbf{x})$, while the variation in the attributes such as x_1 and x_2 will be uncorrelated with $v(f)$. Thus a measure of the *significance* $\sigma(f)_a$ of an attribute a for a function f over a data set S would be the correlation between the *absolute variation* of the function and the absolute variation of that attribute taken between all possible pairs of points in S

$$\begin{aligned}\sigma(f)_a &= \text{correlation } (\{v_{ij}(f)\}, \{v_{ij}(a)\}) \text{ for all pairs } i, j \\ &= \frac{\sum_{i}^{N}\sum_{j>i}^{N} \{v_{ij}(f) - \overline{v}(f)\}\{v_{ij}(a) - \overline{v}(a)\}}{\sqrt{\sum_{i}^{N}\sum_{j>i}^{N} \{v_{ij}(f) - \overline{v}(f)\}} \sqrt{\sum_{i}^{N}\sum_{j>i}^{N} \{v_{ij}(a) - \overline{v}(a)\}}}.\end{aligned} \tag{3}$$

At a given node the attribute with the maximum significance for the neural-network function over the data set of the particular node was selected. The computational complexity of the significance analysis is given in the Appendix. In cases where (3) led to excessive computations, the result is approximated using a randomly selected subset of data pairs. The threshold at which the selected attribute is split is chosen by minimizing the weighted variance. This attribute selection approach is henceforth referred to as the ANN-DT(s) variant of the algorithm.

If on the other hand the sum of the squared errors is minimized in the selection of the attribute, as well as the threshold, as was described in the previous section, the algorithm will be referred to as ANN-DT(e).

E. Stopping Criteria and Pruning

The current set of data is recursively split two ways into progressively smaller subsets. This recursion is terminated when the standard deviation or the variance is zero, or when some stopping criterion is complied with. Such stopping criteria prevent tree branches from being created where the outcome of one of the subbranches would not be significantly different from the outcome of another. These so-called prepruning methods are usually implemented for two reasons. The first is to prevent the tree from modeling noise in the data and the second is to improve tree intelligibility. Many neural-network architectures and training algorithms are available which can adequately compensate for noise in the training data. Therefore in the context of rule-extraction from neural networks prepruning techniques are primarily used to improve the intelligibility of rules.

Statistical tests are applied to the outcomes of the data contained in the two new branches. An F-test [31] can be used to ascertain whether the mean outputs of the records of each of the two subbranches are significantly different from one another. This test shows whether continued recursion would be meaningful or not. If the selected termination criterion fails at some confidence level α, the current node is converted into a terminal node. This stopping criterion can fail at a certain tree depth, even though future splits further down the tree could become statistically significant again. To prevent premature cessation of tree growth as a result of the above tests failing, these tests were only applied to nodes occurring below a certain minimum depth in the tree. Furthermore, if a branch is to contain only one data record the statistical test is also taken to have failed. This means that a single data point cannot be translated into a rule and is typical for decision trees. One more criterion was applied to prevent unnecessarily large or incomprehensible trees from being formed. The criterion was that the tree could only grow to a user-defined maximum depth.

IV. Splitting Procedures and Pruning of CART

The CART algorithm follows a splitting procedure based on the minimization of a given error measure. The least-squares regression version of the CART algorithm, used in this paper, forms a decision tree by iteratively splitting nodes on an attribute and threshold value in order to minimize the weighted variance [4] over the branches. This is the same criterion as used in the ANN-DT(e) algorithm. The variant of CART used in this paper made use of minimal error complexity pruning [32] with ten-fold cross-validation estimates [4]. This is the default pruning technique of CART.

The statistical pruning technique used by both the ANN-DT(e) and ANN-DT(s) algorithms is a greedy pruning technique compared with the more sophisticated pruning technique of the CART algorithm. The advantage of the greedy technique is that it is considerably faster. However, one can expect CART's pruning technique to do as well and sometimes better than statistical pruning. It should also be noted that any pruning techniques, including those using cross-validation (CART), can also be applied to the trees developed by the ANN-DT algorithm. Thus it would be possible to prune the trees evolved with ANN-DT(e) and ANN-DT(s) with CART's pruning technique. As with the CART algorithm, this more sophisticated pruning technique could possibly perform better on some data sets, but will also be slower. For a large data set and a large number of sample points, ten-fold cross-validation

estimates might prove to be too computationally expensive. Under these circumstances, other pruning techniques that grow the tree to full depth before applying statistical pruning mechanisms may be good alternatives.

In further experiments CART was also pruned using statistical pruning with the same setting as that used for both the ANN-DT(e) and ANN-DT(s) algorithms. It is important to note that the only difference between the ANN-DT(e) algorithm and CART using this pruning technique is the fact that ANN-DT(e) samples the neural network output. This was to make sure that in this investigation the different pruning techniques of the algorithms are not the main reason for any observed differences between ANN-DT and CART. In these and other experiments the results obtained by CART using the statistical pruning technique (χ^2 test) did not differ significantly from those obtained by CART using error complexity pruning. Therefore only the results of the latter (default CART algorithm) are considered below.

V. Illustrative Examples

The performance of both the ANN-DT(e) algorithm and the ANN-DT(s) variant were compared to CART on one artificial data set and two real-world data sets. The neural networks used for the problems considered in this paper were radial basis function neural networks trained by an evolutionary algorithm [28], as well as multilayer perceptrons with hyperbolic transfer functions and weights trained with the generalized delta learning rule with momentum [22].

Except for the problems pertaining to the artificial data (Case study 1) the ANN-DT algorithm sampled the trained neural network 20 times for every training point. In other words, for every original training point, 20 sample points were synthesized from the training data for querying the trained neural network. It should be noted that during the integration steps, the ANN-DT(s) algorithm made additional queries to the neural network in order to determine the most significant inputs. Runs were also conducted without generation of additional sampling points, i.e., based on the training data only.

During the construction of trees by the ANN-DT algorithm, specific stopping criteria and pruning methods were applied. In particular, statistical prepruning commenced at a depth of five levels with a confidence level α of 95%. The trees were kept relatively small and the maximum depth to which the trees could grow was set so that the size of the trees evolved by the ANN-DT algorithm was comparable to that of the trees developed by the CART algorithm. For all case studies other than case study 1 (discussed below) the results of the various algorithms on the validation sets were found to be fairly insensitive to changes in the above-mentioned settings.

A. Case Study 1: Sine and Cosine Curves

This is a synthetic problem with four inputs, of which three are continuous (θ, x, s) and one is discrete (ϕ). The single continuous output (y) is given by (4)

$$y = \sin(4\pi\theta - \phi) + ax + bs + c\varepsilon \tag{4}$$

with $0 \leq (\theta, x, s) \leq 1$, and the discrete variable ϕ assuming either the value zero, or $\pi/2$. ε was a random variable with a normal distribution with a zero mean and a standard deviation of unity.

A training set of 300 uniformly distributed random data points was generated with equation parameters $a = 0.3$, $b = 0.0$, and $c = 0.2$. Note that although the parameter b is zero, and the influence of variable s therefore nonexistent, the various algorithms still had to deal with what was essentially a nuisance variable. The test set consisted of 1200 points for which the outputs were computed without the error term, i.e., $c = 0$.

This is a difficult problem to solve for algorithms such as CART that use a greedy heuristic to induce the decision tree, because the data points need to be split on attribute ϕ and several times on θ (not necessarily in that order). Initially nothing is gained in terms of reducing the error from a split on ϕ and only after several splits on θ is the error reduced significantly. Algorithms employing greedy heuristics have difficulty finding good solutions under these circumstances. Other than possibly reducing the variance resulting from the error term, a split on x will reduce the variance only marginally. Moreover, the noise factor $(c\varepsilon)$ causes methods that produce rules that are supported by insufficient numbers of exemplars to overtrain. Since the initial splits do not reduce the error, premature pruning would give undesirable results. The F-test was consequently also only applied at a depth of five (as was the case with all the other case studies). Unlike the other case studies, the outcomes of the case study would be sensitive to changing this setting to for example, a depth of two or three.

A multilayer perceptron with 20 hyperbolic tangent hidden nodes was trained on the data generated by (4). The ANN-DT(e) and ANN-DT(s) algorithms were used to extract rules from the trained neural network. For five different experiments the effect of the number of sample exemplars on the quality of the extracted rules was examined. Different numbers of sample points were used to query the trained neural networks, *viz.* zero, 250, 500, 1000, and 1500 exemplars.

In another experiment the effect of noise on the performance of the algorithms was examined. The noise factor c was therefore set at different levels, specifically 0.0, 0.1, 0.2, and 0.3. In this experiment the number of sample exemplars used to query the trained neural networks was held constant at 500 exemplars.

B. Case Study 2: Abalone Data

This data set obtained from the UCI Repository of Machine Learning Databases at Irvine, consisted of eight inputs, namely the sex of the specimen (male, female, or infant), the length, the diameter, the height, the total weight, the shucked weight, the weight of the viscera, and finally the shell weight of the specimen, that were used to predict the age of the abalone [33]. At present the age of a specimen is determined by cutting the specimen's shell through the cone, staining the shell, and then counting the number of shell rings using a microscope. This is a time-consuming task that can be avoided if the specimen's

TABLE I
THE PERCENTAGE OF VARIANCE EXPLAINED ($100*R^2$) BY THE VARIOUS TECHNIQUES ON THE CASE STUDIES. BOLD NUMBERS INDICATE WHEN THE ACCURACY OF DECISION TREES OF A CERTAIN ALGORITHM IS SIGNIFICANTLY HIGHER THAN THE ACCURACIES GIVEN IN NORMAL PRINT, (i.e., EXCLUDING RESULTS FOR THE ANN, WHICH ARE INDICATED BY ITALICS). NUMBERS IN PARENTHESES ARE RESULTS OBTAINED WITHOUT ADDITIONAL SAMPLING OF THE NEURAL NETWORK (ANN)

Case Study	*ANN*	ANN-DT(e)	ANN-DT(s)	CART
1. Sin-Cos data (c = 0.3)	*86.1*	**77.6**(52.1)	**82.3(77.7)**	32.9
2. Abalone data set	*57.0*	**47.0(49.6)**	**45.5(46.5)**	40.3
3. Pine data set	*90.4*	**85.2(85.5)**	80.0(81.1)	**86.3**

TABLE II
THE FIDELITY OF THE CASE STUDIES IN TERMS OF $100*R^2$ WITH RESPECT TO THE NEURAL-NETWORK OUTPUT. THE FIDELITY VALUES OF THE CART ALGORITHM IN THE LAST COLUMN SERVE AS A BASELINE FOR COMPARISON ONLY (THEY ARE LIKELY TO BE LOW, SINCE CART AND THE NEURAL NETWORK HAVE DIFFERENT BIASES AND CART DOES NOT TRY TO DESCRIBE THE NEURAL-NETWORK MODEL). NUMBERS IN PARENTHESES ARE RESULTS OBTAINED WITHOUT ADDITIONAL SAMPLING OF THE NEURAL NETWORK (ANN)

Case Study	ANN-DT(e)	ANN-DT(s)	CART
1. Sin-Cos data (c = 0.3)	84.0	87.2	27.5
2. Abalone data set	82.47(82.21)	77.57(79.25)	76.0
3. Pine data set	92.3(93.6)	87.2(86.8)	87.93

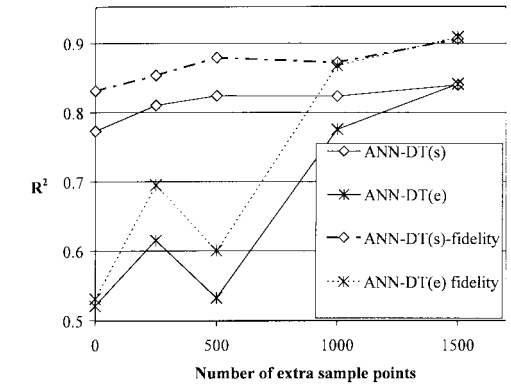
TABLE III
THE NUMBER OF LEAVES OF THE DECISION TREES EVOLVED BY THE ANN-DT(e), ANN-DT(s), AND CART ALGORITHMS. NUMBERS IN PARENTHESES ARE RESULTS OBTAINED WITHOUT ADDITIONAL SAMPLING OF THE NEURAL NETWORK (ANN)

Case Study	ANN-DT(e)	ANN-DT(s)	CART
1. Sin-Cos data (c = 0.3)	54(30)	37(30)	31
2. Abalone data set	32(32)	32(30)	27
3. Pine data set	446(307)	353(176)	543

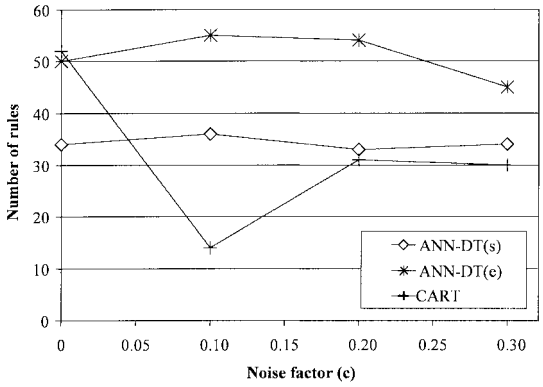
age can be predicted reliably from the more easily obtainable physical characteristics. The data consisted of 3133 training records and 1044 records for evaluation of the trained models, after removal of incomplete records. In our experiment the sex input feature was encoded as {1; 0} for type male, {0; 1} for type female and {0; 0} for the infant type. Although the output is actually discrete (the age is an integer ranging from one to 30) it was decided to treat the output as continuous and the root-mean-square error was minimized. After some preliminary experiments on the data, a multilayer perceptron with six nodes in the hidden layer was used. The neural network was trained for a total of 5000 epochs.

C. Case Study 3: Sap Flow in Pine Trees

In this data set the sap flow rate in pine trees was related to nine continuous factors, namely temperature, relative humidity, differential vapor pressure, photoactive radiation, leaf mass, height of the tree, diameter at breast height, xylem pressure potential, and the season. The data consisted of 6612 data records of which two-thirds were used for training and the remainder for testing the various algorithms. The ANN-DT algorithm was used to extract decision trees from an basis function nodes with 20 axis-parallel ellipsoidal basis function networks, trained with an evolutionary algorithm [28].



(a)



(b)

Fig. 3. (a) The accuracy on the test data versus the number of sample points beyond the training points used by the ANN-DT(s) and ANN-DT(e) algorithms to sample the trained multilayer perceptron of case study 1. The dashed line indicates the fidelity with respect to the multilayer perceptron from which the rules were extracted. The noise factor was held constant at 0.3. (b) The number of rules induced by the ANN-DT(s) and ANN-DT(e) algorithm against the number of additional neural network sampling points made in case study 1.

VI. RESULTS

The results obtained with the various algorithms as specified previously are summarized in Tables I–III. The scores of Table I are in terms of the coefficient of determination [33], i.e., $R^2 = 1 - \Sigma(y_p - y_t)^2/\Sigma(y_t - y_{avg})^2$, where y_p is the predicted value of the outcome, y_t is the target value of the outcome, and y_{avg} is the average target value of the outcomes. The corresponding fidelity to the neural network is given in Table II, while Table III contains the number of leaves in the decision trees, which is an indication of the complexity of the trees. For these binary trees the number of internal nodes is equal to one less than the number of leaves. For case study 1 the results reported on the ANN-DT algorithms in Table I are those with a noise factor of 0.3 and 1000 additional sample points to the neural network beyond the training points. Note that the additional points of the ANN-DT algorithm mentioned here, are the sample points obtained using the neural network, as described in Section III-C, that is they are generated by the algorithm and not by (4). All the algorithms were therefore presented with the same training data, and were also evaluated on the same test data.

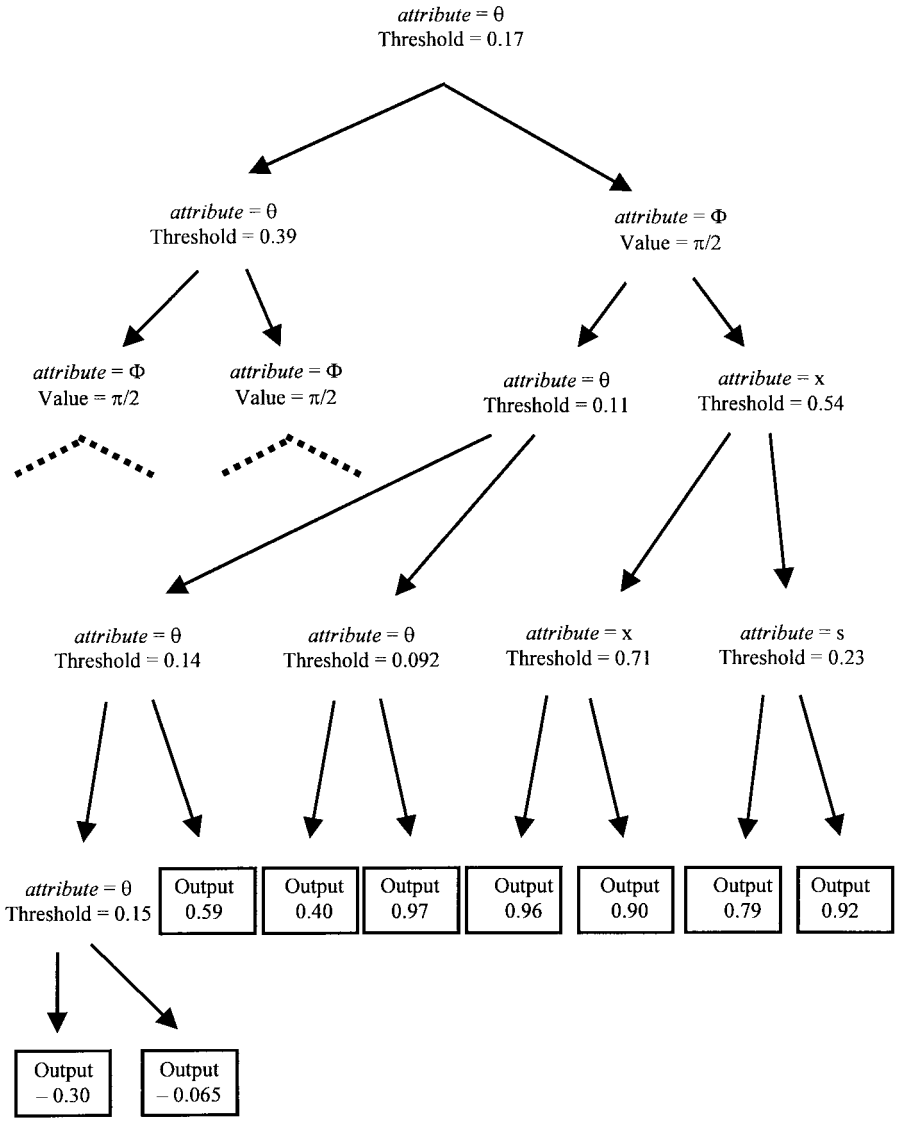


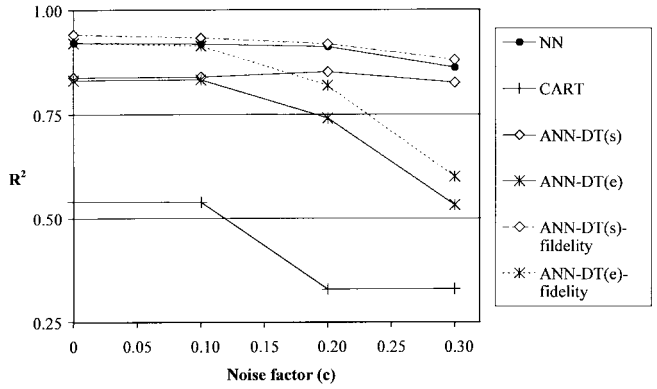
Fig. 4. The decision tree extracted by the ANN-DT(s) algorithm from the trained neural network for case study 1 with $c = 0.3$ using 1000 points to sample the neural network. If attribute a < Threshold the right subtree applies, else the left subtree is valid.

A. Case Study 1: Sine and Cosine Curves

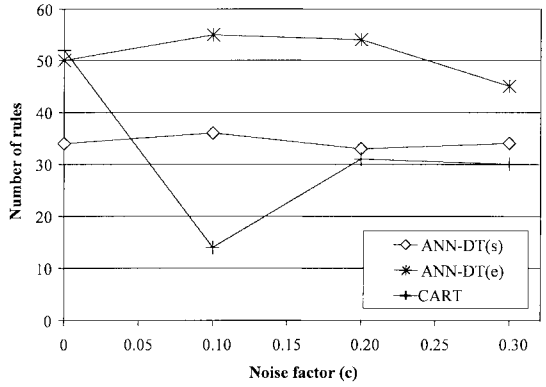
The fidelity and accuracy of ANN-DT(s) and ANN-DT(e), given in terms of the coefficient of determination (R^2), for different numbers of neural network sample points, are shown in Fig. 3(a). The number of rules obtained by the respective algorithms for the test runs is given in Fig. 3(b). The results shown in Fig. 3(a) clearly suggest that the extra sampling points make a significant contribution. Both accuracy and tree size increased with an increase in the number of sample points. Although the ANN-DT(e) algorithm failed almost completely to capture the overall trends in the data, the ANN-DT(s) algorithm yielded satisfactory results, even with few sample points.

Furthermore, even with no extra sample points beyond the training data points (in which case the neural networks only perform noise filtering) the ANN-DT(s) algorithm yielded satisfactory results, as well as performing markedly better than the ANN-DT(e) algorithm. This implies that the significance analysis enables the ANN-DT(s) method to place the attributes ϕ and θ high up in the decision tree. If one considers this data set as a whole, these attributes clearly have the greatest influence on the outcome of the neural network. For $c = 0.3$ and with 1000 sample points, ANN-DT(s) calculates $\sigma(f)_\theta$ as 0.45 and $\sigma(f)_\phi$ as 0.30 at the first split. The other two attributes each had a level of significance less than 0.02. It can be seen from the decision tree in Fig. 4 that ϕ was selected once at tree depth of one and twice at a tree depth of two.

The greedy attribute selection measure of ANN-DT(e) and CART split relatively late on the attribute ϕ, because a split on this attribute caused very little immediate gain. Although not shown in the figure, CART split the data on this attribute once at a depth of two, three, and four and ANN-DT(e) split even lower at a depth of three, four, and five. After too many splits on insignificant attributes the data were too sparse to pick up any underlying trends. This shortcoming of the greedy attribute selection measure that is used by CART cannot be compensated for at a later stage by pruning. Pruning will attempt to replace a subbranch by leaves, but will not



(a)



(b)

Fig. 5. (a) A plot of the accuracy (R^2) of the algorithms for different noise factors in the test data of case study 1. The dashed line indicates the fidelity with respect to the neural network, a multilayer perceptron from which the rules were extracted. The size of the data set with which the ANN-DT(s) and ANN-DT(e) algorithms sampled the neural network beyond the 300 training points, was held constant at 500. All the decision trees were pruned statistically with an α value of 0.05. (b) The numbers of rules induced by the algorithms in case study 4.

recalculate existing splits. Noise will make this task even harder. That is why CART's performance decreased further in the presence of noise, as can be observed from a plot of the algorithm's performance against the noise as shown in Fig. 5(a). Fig. 3(b) together with Fig. 5(b), which gives the respective number of rules obtained by each of the algorithms, reveal that the simpler trees generated by the ANN-DT(s) algorithm also perform better.

B. Data Sets 2 and 3: Abalone and Pine Data

For the Abalone problem the ANN-DT algorithms performed notably worse than the neural network, although still better than the CART algorithm. A two-sided paired t-test on the squared errors of the 1044 test points showed that this improvement was significant at a 96% confidence level for the ANN-DT(e) method, while the complexity of the trees was very similar, as can be seen from Table III. A further run with different initial states confirmed these results.

For the Pine data set, the CART algorithm had a slightly higher accuracy than the ANN-DT(e) algorithm (not significant within a 90% confidence limit). Both algorithms were significantly more accurate than the ANN-DT(s) variant.

VII. Discussion

In all the case studies, the ANN-DT algorithms successfully extracted faithful rule-based representations from the trained neural-network models. An interesting result is that the rules induced by the ANN-DT(e) algorithm are as accurate and sometimes more accurate than those induced by the CART algorithm. Similar results were found by Craven and Shavlik [19], who compared the TREPAN algorithm to classification trees induced by C4.5 [3] and ID2-3 [34]. The results indicate that for many problems inductive techniques, like C4.5 and CART, do not use all the information that is contained in the original data. A possible source of this loss of information is that the techniques split the data recursively into branches in such a way that the data to be processed in the underlying branches are isolated from another. This means that any trend that might exist between the input and the output data which is distributed over points belonging to different branches, will not be discovered by these algorithms. It also means that points not complying with this trend as a result of noise cannot be identified and a rule can arise out of these exceptions that does not generalize well.

If it is assumed that the neural network detects these trends and does not overtrain on outliers in the data, both the ANN-DT(e) and ANN-DT(s), as well as the TREPAN algorithm are evolved on data where these exceptions are already removed. Moreover, the more densely sampled points help in finding better estimates of the threshold values at which the data should be split. In contrast, the C4.5 and CART algorithms can only estimate this value to lie somewhere between two points of the subset of the original data that belongs to the branch in which the next split is to be made. This subset is much denser in the case of the algorithms extracting rules from neural networks.

Moreover, contours or class boundaries for classification problems that extend over different branches of the tree, can be estimated in regions where there is very little or no training data. This is because the neural network does not split the data and can extend such a decision boundary between the training points via interpolation. The ANN-DT algorithm can sample in these regions and produce additional rules to cover these regions. For the same reasons the ANN-DT algorithms tend to maintain a higher fidelity with respect to the neural network.

Both ANN-DT(s) and ANN-DT(e) can be applied to nonparametric models other than feedforward neural networks, without making any assumptions about the model's internal states or the nature of the data. The computational time of the ANN-DT algorithm scales linearly with the neural-network size and is only dependent on the time it takes the neural network to assign a label to a data point. However, the algorithm's computational time does suffer from the curse of dimensionality. In order to achieve a higher density of points than that of the training data, progressively more sample points are required as the dimensionality of the data increases. This problem can be reduced somewhat by initially using fewer sample points and growing the tree from a node in a best-first manner. This is performed in the TREPAN algorithm [19] by presenting the node that is most likely to increase fidelity with

sufficient samples. Naturally the number of these sample points also needs to grow exponentially with the dimensionality of the data in order to achieve the same accuracy, as once a split is made in the tree it cannot be adjusted later.

In case study 1, it was seen that the use of the significance analysis in attribute selection can hold significant advantages over the greedy variance criterion. Although single splits on attributes ϕ and θ did not cause a significant decrease in the normalized variation of the data in case study 1, changes in these attributes were nevertheless correlated with changes in the output of the neural network and therefore had high σ_f values. Provided that the neural network accurately models the input–output relationships represented by the data, the significance analysis therefore learns from the trained neural network which attributes have the most influence over the data set covered by a particular node. On the other hand, the greedy splitting criteria of the CART and ANN-DT(e) algorithms did not compensate for the periodicity of the function with respect to the attribute θ. The ANN-DT(e) algorithm could use additional sample points to obtain a satisfactory performance. In contrast, after performing initially greedy splits, the CART algorithm no longer had sufficient data points to identify the periodic functional behavior. Although case study 1 concerned a synthetic data set, such a periodic response of the output to one of the attributes with relatively sparse noisy data can also occur in real-world data sets. The fact that both ANN-DT(e) and especially ANN-DT(s) could overcome such a pitfall is at least in theory a significant benefit of the ANN-DT algorithm compared to the CART algorithm.

VIII. Conclusions

A novel approach has been developed to extract decision trees from trained feedforward neural networks, regardless of the structures of these networks. It was found that in some cases these rules were significantly more representative of the behavior of the neural network than rules extracted from the training data only.

Alternatively, the algorithm can be used as a method to extract rules from data sets. These rules appear to be of similar accuracy as those obtained with CART. In fact, in some cases a significant improvement could be obtained with the ANN-DT algorithm.

In one particular case it was demonstrated that the significance analysis of the ANN-DT(s) could correctly identify the most important attributes and build valid sets of rules. In contrast to this, a greedy error driven procedure, such as used in CART and ANN-DT(e), failed to identify the most important attributes. As a result, rules derived with CART and ANN-DT(e) were comparatively inaccurate, while the ANN-DT(e) algorithm could only find more accurate rules by using many more sample points than ANN-DT(s).

Unlike a sensitivity analysis that only considers the partial derivatives, the significance analysis proposed in this paper takes the correlational structure of the data into account. This significance analysis appears to be a suitable splitting criterion near the root of the decision tree, whereas a greedy splitting criterion would be better at splitting the lower branches of the tree.

For some case studies, additional sampling of a trained neural network resulted in appreciable improvement in the accuracy of the rules extracted from the network.

Appendix
Computation of (3)

Equation (3) is a simple correlation that is computed for the $n(n-1)$ data pairs. The correlation is quick to compute, however it can be seen that it requires knowledge of the $n(n-1)$ values of v_{ij}, one for each data pair. Each of these has to be integrated, as indicated by (2). The integral is best approximated by a finite sum. For this the following should be noted:

$$\int_a^{a+\Delta x} |\Delta f(x) \cdot u|\, dx = |f(\mathbf{a}+\Delta\mathbf{x}) - f(\mathbf{a})| \qquad \text{(A.1)}$$

(where $\mathbf{u}$ is the unity vector in direction $\Delta\mathbf{x}$), when f is a monotonic function between $\mathbf{a}$ and $(\mathbf{a}+\Delta\mathbf{x})$. If no absolute value were taken in both of the expressions in (A.1), then the top expression in (A.1) would trivially be equal to the bottom expression. When the absolute value is taken, the equality can be seen to hold if $\Delta f(\mathbf{x}) \cdot \mathbf{u}$ stays either positive or negative between $\mathbf{a}$ and $(\mathbf{a}+\Delta\mathbf{x})$. The integral in (4) can be separated into the sum of several smaller integrals, each of which can be approximated, using (A.1). In the simulation program, each of the integrals was separated into ten subintegrals. This assumes that the neural network function does not change the direction of the gradient too often between the data points. Thus in this case the computation of (2) takes ten times as long as it takes to compute a sample point of the neural network, denoted as T(NN). Therefore computation of (3) takes $\frac{1}{2}n(n-1) * 10 *$ T(NN) for n data points. In case n becomes large (in our program if $n > 250$), (3) was computed for $250 * 125$ randomly chosen pairs of points. This led to a time proportional to $250 * 1250 *$ T(NN).

References

[1] R. Davis, B. G. Buchanan, and E. Shortliffe, "Production rules as a representation for a knowledge-based consultation program," *Artificial Intell.*, vol. 8, no. 1, pp. 15–45, 1977.

[2] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[3] ——, *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann, 1993.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.* New York: Chapman and Hall, 1984.

[5] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-Based Syst.*, vol. 8, no. 6, pp. 373–383, 1995.

[6] M. W. Craven and J. W. Shavlik, "Using sampling and queries to extract rules from trained neural networks," in *Proc. 11th Int. Conf. Machine Learning*, San Francisco, CA, 1994.

[7] G. Towell and J. W. Shavlik, "Extracting refined rules from knowledge based neural networks," *Machine Learning*, vol. 13, pp. 71–101, 1993.

[8] L. M. Fu, "Rule learning by searching on adapted nets," in *Proc. 9th Nat. Conf. Artificial Intell.*, Anaheim, CA, 1991, pp. 590–595.

[9] S. I. Gallant, *Neural Network Learning and Expert Systems.* Cambridge, MA: MIT Press, 1993.

[10] I. K. Sethi, "Neural implementation of tree classifiers," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 1243–1249, 1995.

[11] S. Thrun, "Extracting provable correct rules from artificial neural networks," Institut für Informatik III Universität, Bonn, Germany, Tech. Rep. IAI-TR-93-5, 1994.

[12] _____, "Extracting rules from artificial neural networks with distributed representation," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds., 1995, vol. 7.
[13] E. Pop, R. Hayward, and J. Diederich, "RULENG: Extracting rules from a trained artificial neural network by stepwise negation," in *QUT NRC*, Dec. 1994.
[14] K. Saito and R. Nakano, "Medical diagnostic expert systems based on PDP model," in *Proc. IEEE Int. Conf. Neural Networks*, San Diego, CA, 1988, vol. 1, pp. 255–262.
[15] H. Narazaki, T. Watanabe, and M. Yamamoto, "Reorganizing knowledge in neural networks: An explanatory mechanism for neural networks in data classification problems," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, pp. 107–117, 1996.
[16] A. B. Tickle, M. Orlowski, and J. Diederich, "DEDEC: Decision detection by rule extraction from neural networks," in *QUT NRC*, Dec. 1994.
[17] S. Seisto and T. Dillon, "Automated knowledge acquisition of rules with continuously valued attributes," in *Proc. 12th Int. Conf. Expert Syst. Applicat.*, Avignon, France, May 1992, pp. 645–656.
[18] T. Masters, *Practical Neural Network Recipes in C++*. Reading, MA: Academic, 1993.
[19] M. W. Craven and J. W. Shavlik, "Extracting tree structured representations of trained networks," in *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. London, U.K.: Morgan Kaufmann/MIT Press, 1996.
[20] G. P. J. Schmitz, C. Aldrich, and F. S. Gouws, "Extracting decision trees from artificial neural networks," in *Proc. Minerals and Materials '96*, Somerset West, South Africa, 1996, vol. 1, pp. 250–257.
[21] C. Aldrich and G. P. J. Schmitz, "Characterization of process systems with decision trees extracted from neural network models," in *Proc. 5th European Congr. Intell. Syst. Soft Comput.*, Aachen, Germany, 1997, pt. 3, pp. 2163–2167.
[22] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: Macmillan, 1994.
[23] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral science," Ph.D. dissertation, Harvard Univ., Cambridge, MA, 1974.
[24] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distribution Processing: Exploration in the Microstructure of Cognition 1.* Cambridge, MA: MIT Press, 1986.
[25] M. J. D. Powel, "Radial basis functions for multivariable interpolation: A review," in *IMA Conf. Algorithms Approximation Functions and Data*, J. C. Mason and M. G. Cox, Eds. Oxford, U.K.: Oxford Univ. Press, 1985, pp. 143–167.
[26] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst. 2*, pp. 321–355, 1988.
[27] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 4, pp. 281–294, 1989.
[28] G. P. J Schmitz and C. Aldrich, "Combinatorial evolution of regression nodes in feedforward neural networks," *Neural Networks*, vol. 12, no. 1, pp. 175–189.
[29] W. J. Krzanowski and F. H. C. Marriott, *Kendall's Library of Statistics 2, Multivariate Analysis Part 2.* London, U.K.: Arnold, 1995.
[30] J. C. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, vol. 27, pp. 857–872, 1971.
[31] W. L. Hays, *Statistics.* Orlando, FL: Holt, Rinehart and Winston, 1988.
[32] J. Mingers, "An empirical comparison of pruning methods for decision tree induction," *Machine Learning*, vol. 4, pp. 227–243, 1989.
[33] W. J. Nash, T. L. Sellers, S. R. Talbot, A. J. Cawthorn, and W. B. Ford, "The population biology of abalone (Haliotis species) in Tasmania—I: Blacklip abalone (*H. rubra*) from the north coast and islands of Bass Strait," *Sea Fisheries Division*, Tech. Rep. 48 (ISSN 1034-3288), 1994.
[34] P. M. Murphy and M. J. Pazzani, "ID 2-of-3 constructive induction of M-of-N concepts for discriminators in decision trees," in *Proc. 8th Int. Machine Learning Wkshp.* San Mateo, CA: Morgan Kaufmann, 1991, pp. 183–187.

Gregor P. J. Schmitz was born in Münster, Germany, in 1972. He received the B.Sc. Hons. degree in theoretical physics in 1994 at the University of Stellenbosch, South Africa. He is currently a candidate for the Ph.D. degree in engineering science at the Department of Chemical Engineering, University of Stellenbosch.

His research interests include neural networks, nonparametric regression, process control, and fuzzy logic.

Chris Aldrich received the M.Eng. and Ph.D. degrees in chemical engineering from the University of Stellenbosch, South Africa.

He is currently a Professor in the Department of Chemical Engineering and Director of the Institute of Mineral Processing and Intelligent Systems at the University of Stellenbosch. His research interests include exploratory data analysis, neural networks, data mining, and metallurgical process systems.

Francois S. Gouws received the B.Eng. degree in chemical engineering in 1994 from the University of Stellenbosch, South Africa. Since 1995, he has been pursuing the Ph.D. degree at the same institution.

His research interests include fuzzy rule modeling, model comprehensibility, and combinatorial optimization.