Experimental Evaluation of Performance Improvements in Abductive Network Classifiers with Problem Decomposition

R. E. Abdel-Aal
Center for Applied Physical Sciences, Research Institute,
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

Abstract:

Problem decomposition and divide-and-conquer strategies have been proposed to improve the performance and realization of neural network solutions for complex problems. This paper reports on an experimental evaluation of performance gains brought about by problem decomposition for abductive network classifiers that classify four noisy waveform patterns having two waveform types (sine/cosine) and two different frequencies. Two-stage problem decomposition improves overall classification accuracy from 87.2% to 99%. Problem decomposition classifiers were found to be much more tolerant to model simplification and reduction in the training set size compared to monolithic solutions. This allows trading off some of the large gain in classification performance for some other advantages that may be quite desirable in some applications, such as simpler models that execute faster and are easier to implement, smaller training sets, and shorter training times. A problem decomposition classifier is more accurate than a monolithic classifier in spite of the former being five times simpler, executing over two times faster, requiring one fifth of the training data, and synthesized in one eleventh of the training time. Performance is comparable with a neural network solution using the same decomposition method and significantly superior to an abductive network committee approach.

Index Terms: Classifiers, Abductive Networks, Neural Networks, Machine Learning, Problem Decomposition, Divide and Conquer, Classification Accuracy, Modular Networks, Gaussian Noise, Network Committee.

Dr. R. E. Abdel-Aal,
P. O. Box 1759,
KFUPM, Dhahran 31261
Saudi Arabia
e-mail: radwan@kfupm.edu.sa
Phone: +966 3 860 4320, Fax: +966 3 860 4281

I. INTRODUCTION

In spite of being demonstrated on many small-sized problems, artificial neural networks do not scale up well [22]. Their performance deteriorates rapidly with the increase in the network size due to the larger increase in the training time and connection complexity. Classification represents an important application area of neutral networks that suffers from such limitations. The divide-and-conquer approach has been proposed to improve the performance and realization of neural network solutions to real life problems through problem decomposition. Instead of tackling the whole complex problem in one go, the problem is divided into a number of simpler, more manageable sub-problems, each of which can be solved by a separate network. The resulting network modules are simpler than a single (monolithic) network that attempts to solve the problem as a whole, and therefore would generalize better, thus improving prediction performance. Simpler networks also train faster, with the possibility of training in parallel to further reduce training time. They would also be easier to realize physically as VLSI circuits where practical limitations exist on the number of connections associated with a node [28]. Resulting smaller modular networks reduce the requirement on training sample size, which is useful in handling high-dimensionality data as in remote sensing applications [32]. Problem decomposition has been applied to phoneme classification from acoustic spectra, leading to an order of magnitude reduction in neural network training time for comparable performance [28]. The technique has greatly simplified the training of neural networks required to steer a tractor-trailer truck to dock while backing up through decomposing the control problem into a number of smaller subtasks [19]. A 2-level modular neural networks approach for classifying the auditory brainstem response has proved simpler, trained faster, and yielded higher recognition rates as compared to non-modular networks [33]. More recently, problem decomposition has been used in finding real roots of polynomials [15], predicting crosstalk in VLSI circuits [17], and in automatic face detection [9]. A number of approaches exist for decomposing a complex problem into a set of simpler ones. In the manual approach, decomposition is performed by the designer prior to training based on knowledge of the problem,

e.g. [17,19]. For multi-class classification problems, manual class decomposition is a straight forward approach, where for example, a K-class classifier can be replaced by K two-class modules, each trained to recognize one class from its complement [5]. Techniques have also been described for performing the decomposition automatically during training without requiring any prior knowledge on the decomposition of the problem, e.g. [14]. Another closely related approach for modular networks is that of network committees or ensembles [l8], where individual modules cooperate to improve performance. With this approach, a number of classifiers are used in parallel and their outputs combined to produce the final committee output. Individual classifier outputs are often combined using simple measures such as majority voting or weighted averaging, without involving the input vector of attributes [20]. Alternatively, a gating network uses the input vector to determine the optimum weighting factors for each case to be classified [31]. In the stacked generalization approach, the combiner takes the form of another higher-level network trained on the outputs of individual members to generate the committee classification output [13]. When member classifiers are independent, the resulting diversity in the decision making process is expected to enhance generalization performance, thus improving the accuracy, robustness, and reliability of classification. With neural network committees, increased diversity among individual members is attempted through training on different parts of the dataset [25] or different input features [29], or through using different network architectures [27], different learning paradigms [26], or different training parameters [8]. Resampling methods, such as bootstrap sampling, have been used to increase independence among training subsets for individual committee members. They form the basis for the bagging [7] and boosting [12] ensembling methods. Techniques for automatically enhancing negative correlation between individual members of a network committee have been described [34].

Abductive or polynomial networks [24] based on the self-organizing group method of data handling (GMDH) [10] offer an alternative machine learning approach that has been somewhat neglected in the literature. Compared to neural networks, the method offers the advantages of

automated model synthesis requiring little or no user intervention, faster convergence during model synthesis without the problems of getting stuck in local minima, automatic selection of relevant input variables, and automatic configuration of model structures [4]. With the model represented as a hierarchy of polynomial expressions, resulting analytical model relationships can provide insight into the modeled phenomena, highlight contributions of various inputs, and allow comparison with previously used empirical models. Training stops automatically to avoid over-fitting by using a proven regularization criterion that penalizes model complexity and does not require a separate cross validation data set. Abductive networks have been used for modeling, prediction, and classification in a variety of applications, e.g. [1,2,16,21,30].

Although modular solutions have been described in many forms using neural networks, such techniques have not been applied to abductive networks. With the two machine learning approaches being distinctly different in many ways, an investigation is warranted into the use of modular approaches with abductive network. This paper reports on an experimental investigation into performance gains with abductive network classifiers through problem decomposition for a typical application of classifying noisy waveforms. Following an overview of GMDH and abductive network modeling, the classification problem of discriminating is described. Monolithic classifiers using various combinations of model complexity and training set size are presented and their performance evaluated. Two-stage problem decomposition is then introduced and performance is compared with that of the monolithic approach. Results are compared with neural network modular solutions developed on the same data, and with an abductive network committee approach. Various scenarios are considered for trading off some of the significant improvement in classification accuracy brought about by problem decomposition for other benefits such as simpler models, faster training, and smaller training sets.

II. GMDH and AIM ABDUCTIVE NETWORKS

AIM (abductory inductive mechanism) [3] is a supervised inductive machine-learning tool for automatically synthesizing abductive network models from a database of inputs and outputs representing a training set of NT solved examples. As a GMDH algorithm, the tool can automatically synthesize adequate models that embody the inherent structure of complex and highly nonlinear systems. The automation of model synthesis not only lessens the burden on the analyst but also safeguards the model generated from influence by human biases and misjudgments. The GMDH approach is a formalized paradigm for iterated (multi-phase) polynomial regression capable of producing a high-degree polynomial model in effective predictors. The process is 'evolutionary' in nature, using initially simple (myopic) regression relationships to derive more accurate representations in the next iteration. To prevent exponential growth and limit model complexity, the algorithm selects only relationships having good predicting powers within each phase. Iteration is stopped when the new generation regression equations start to have poorer prediction performance than those of the previous generation, at which point the model starts to become overspecialized and therefore unlikely to perform well with new data. The algorithm has three main elements: representation, selection, and stopping. It applies abduction heuristics for making decisions concerning some or all of these three aspects.

To illustrate these steps for the classical GMDH approach, consider an estimation database of n_e observations (rows) and $m+1$ columns for m independent variables ($x_1, x_2, ..., x_m$) and one dependent variable y. In the first iteration we assume that our predictors are the actual input variables. The initial rough prediction equations are derived by taking each pair of input variables (x_i, x_j ; $i,j = 1,2,...,m$) together with the output y and computing the quadratic regression polynomial [10]:

$$y = A + B\,x_i + C\,x_j + D\,x_i^2 + E\,x_j^2 + F\,x_i\,x_j \qquad (1)$$

Each of the resulting $m(m-1)/2$ polynomials is evaluated using data for the pair of x variables used

to generate it, thus producing new estimation variables (z_1, z_2, ..., $z_{m(m-1)/2}$) which would be expected to describe *y* better than the original variables. The resulting *z* variables are screened according to some selection criterion and only those having good predicting power are kept. The original GMDH algorithm employs an additional and independent selection set of n_s observations for this purpose and uses the regularity selection criterion based on the root mean squared error r_k over that data set, where

$$r_k^{\,2} = \sum_{\ell=1}^{n_s} (y_\ell - z_{k\ell})^2 \Big/ \sum_{\ell=1}^{n_s} y_\ell^{\,2} \quad ;k = 1,2,\ldots,m(m-1)/2 \tag{2}$$

Only those polynomials (and associated *z* variables) that have r_k below a prescribed limit are kept and the minimum value, r_{min}, obtained for r_k is also saved. The selected *z* variables represent a new database for repeating the estimation and selection steps in the next iteration to derive a set of higher-level variables. At each iteration, r_{min} is compared with its previous value and the process is continued as long as r_{min} decreases or until a given complexity is reached. An increasing r_{min} is an indication of the model becoming overly complex, thus over-fitting the estimation data and performing poorly in predicting the new selection data. Keeping model complexity checked is an important aspect of GMDH-based algorithms, which keep an eye on the final objective of constructing the model, i.e. using it with new data previously unseen during training. The best model for this purpose is that providing the shortest description for the data available [6]. Computationally, the resulting GMDH model can be seen as a layered network of partial quadratic descriptor polynomials, each layer representing the results of an iteration.

A number of GMDH methods have been proposed which operate on the whole training data set thus avoiding the use of a dedicated selection set. The adaptive learning network (ALN) approach, AIM being an example, uses the predicted squared error (PSE) criterion [6] for selection and stopping to avoid model overfitting, thus eliminating the problem of determining when to stop training in neural networks. The criterion minimizes the expected squared error that would be obtained when the network is used for predicting new data. AIM expresses the PSE error as:

$$PSE = FSE + CPM\,(2K/n)\sigma_p^{\,2} \quad (3)$$

Where *FSE* is the fitting squared error on the training data, *CPM* is a complexity penalty multiplier selected by the user, *K* is the number of model coefficients, *n* is the number of samples in the training set, and σ_p^2 is a prior estimate for the variance of the error obtained with the unknown model. This estimate does not depend on the model being evaluated and is usually taken as half the variance of the dependent variable *y* [6]. As the model becomes more complex relative to the size of the training set, the second term increases linearly while the first term decreases. *PSE* goes through a minimum at the optimum model size that strikes a balance between accuracy and simplicity (exactness and generality). The user may optionally control this trade-off using the *CPM* parameter. Larger values than the default value of 1 lead to simpler models that are less accurate but may generalize well with previously unseen data, while lower values produce more complex networks that may overfit the training data and degrade actual prediction performance.

AIM builds networks consisting of various types of polynomial functional elements. The network size, element types, connectivity, and coefficients for the optimum model are automatically determined using well-proven optimization criteria, thus reducing the need for user intervention compared to neural networks. This simplifies model development and reduces the learning/development time and effort. The models take the form of layered feed-forward abductive networks of functional elements (nodes) [3], see Fig. 1. Elements in the first layer operate on various combinations of the independent input variables (*x*'s) and the element in the final layer produces the predicted output for the dependent variable *y*. In addition to the main layers of the network, an input layer of normalizers convert the input variables into an internal representation as *Z* scores with zero mean and unity variance, and an output unitizer unit restores the results to the original problem space.

The used version of AIM supports the following main functional elements:

(i) A white element which consists of a constant plus the linear weighted sum of all outputs of the previous layer, i.e.:

"White" Output $= w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$ (4)

where $x_1, x_2,..., x_n$ are the inputs to the element and $w_0, w_1, ..., w_n$ are the element weights.

(ii) Single, double, and triple elements which implement a third-degree polynomial expression with all possible cross-terms for one, two, and three inputs respectively, for example,

"Double" Output $= w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + w_6x_1^3 + w_7x_2^3$ (5)

III. THE CLASSIFICATION PROBLEM

We considered the classification problem of identifying four waveform patterns buried in noise. Two patterns are sine waves with two different frequencies while the other two represent cosine waves having the same two frequencies as the sine waves. A noisy sinusoid was simulated as:

$$y(i) = A\sin\frac{2\pi i}{T} + e(i); i = 1,2,...,50 \quad (6)$$

where A is the amplitude and T is the period. Samples are uniformly spaced in time and the sample spacing is taken as 1 second for simplicity. All four patterns have zero phase and the same nominal amplitude value $A = 5$. The higher frequency pattern had a period $T = T1 = 25$ while the lower frequency pattern had $T = T2 = 30$. This applies also for the two cosine patterns with the cos function replacing the sin function. Classes 1 and 2 are sine waves with periods *T1* and *T2*, respectively, while classes 3 and 4 are cosine waves with periods *T1* and *T2*, respectively. The additive noise component *e(i)* represents samples of a white gaussian noise process with zero mean and standard deviation σ. The value of σ is chosen to produce the required signal to noise ratio, *SNR*, given in decibels by:

$$SNR = 10\log\frac{A^2}{2\sigma^2} \quad (7)$$

All work described here used *SNR* = 0 dBs, which gives $\sigma = 3.536$ for $A = 5$. Each waveform record consists of 50 samples which constitute the input variables to AIM. Table 1 lists values for two separability measures for all possible pair combinations of the four waveform classes, as well as an estimate for the upper bound on the Bayes error. Each class is represented by 500 waveform

records. For Gaussian distributions, the Bhattacharyya Distance, *B*, between two classes is given by [11]:

$$B = \frac{1}{8}(M_2 - M_1)^T \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1}(M_2 - M_1) + \frac{1}{2}\ln \frac{\left|\frac{\Sigma_1 + \Sigma_2}{2}\right|}{\sqrt{|\Sigma_1|}\sqrt{|\Sigma_2|}} \tag{8}$$

where M_1 and M_2 are the mean vectors and Σ_1 and Σ_2 are the covariance matrices for the two classes, respectively. With the two classes being equiprobable, an upper bound on the Bayes error is given by [11]

$$\varepsilon_u = 0.5\, e^{-B} \tag{9}$$

A J-measure of class separability based on scatter matrices is given by [11]:

$$J_4 = \frac{tr(S_b)}{tr(S_w)} \tag{10}$$

where S_b is the between-class covariance matrix and S_w is the within-class covariance matrix. For equiprobable classes, S_w is the average of the two covariance matrices for the two classes and S_b is given by:

$$S_b = \frac{1}{2}\left[(M_1 - M_{12})(M_1 - M_{12})^T + (M_2 - M_{12})(M_2 - M_{12})^T\right] \tag{11}$$

where M_{12} is the global mean of the combined distribution of the two classes,

$$M_{12} = \frac{M_1 + M_2}{2} \tag{12}$$

Both measures show classes 1 and 4 to have the smallest separability, with 1.14% being the percentage upper bound on error probability. Separability decreases and the classification error probability increases at lower values of the signal to noise ratio.

IV. THE MONOLOTHIC CLASSIFIER

For the monolithic solution, a single abductive network was synthesized to identify all four waveform patterns in one go through training on 2000 examples (NT = 2000) consisting of 500

examples of each of the four patterns. Records for the AIM training/evaluation database were derived by appending to each simulated waveform record the corresponding known pattern classification. A typical complete data record is represented as:

Inputs:	Output:
Values of the Waveform Samples	Pattern Classification (Categorical Variable)
y(1) y(2) y(3) … y(50)	1 (= Sine, T = 25) 2 (= Sine, T = 30) 3 (= Cosine, T = 25) 4 (= Cosine, T = 30)

A. Default Model

The top row in Table 2 shows the abductive network model generated using the default value of 1 for the CPM complexity parameter. Here yi indicates *y(i),* the ith time sample of the waveform. The classifier is a 4-layer network comprising four triple elements and one white element and operates on 42 different samples out of the 50 samples of the input waveform. Large numbers for the model layers and selected inputs characterize complex models and increase model execution (classification) time. For comparison with other model configurations, we consider training times relative to the training time for this monolithic network with CPM = 1, which is taken as unity. The number of different waveform samples selected by AIM as model inputs and the relative training times are also shown in the table. The network classification performance was evaluated on a total of 1000 waveform records (250 of each of the four patterns) representing new data previously unseen during training. The computed real output from the classifier was converted into a categorical predicted classification output (1, 2, 3, or 4) by rounding according to the following rule:

IF computed output < 1.5 THEN Predicted Classification = 1 (Sine, T = 25)
IF 1.5 ≤ computed output < 2.5 THEN Predicted Classification = 2 (Sine, T = 30)
IF 2.5 ≤ computed output < 3.5 THEN Predicted Classification = 3 (Cosine, T = 25)
IF 3.5 ≤ computed output THEN Predicted Classification = 4 (Cosine, T = 30) (13)

Table 3 shows the results of classifying the four patterns of the evaluation data set in the form of

percentage values for waveform (sine/cosine), period (*T1/T2)*, and overall classification accuracies. The period classification accuracy was derived only for those waveforms (sines and cosines) that were correctly classified. Therefore, the overall classification accuracy is the product of the other two parameters. Shown also in the table are the mean and standard deviation values for the absolute error between the predicted output (before rounding) and the known classification output. The top row of Table 3 indicates an overall classification accuracy of 87.2% for the default monolithic model.

B. Simpler Model

We enforced the synthesis of a simpler classifier by setting the CPM parameter to 5 instead of the default value of 1. Simpler network structures are easier to implement and have the advantages of reduced training time, faster classification, and simpler model description. Table 2 shows the resulting classifier which consists of only 3 layers with the first layer consisting of a simpler White element as compared to the Triple element for the default model. The simpler network uses only 21 waveform samples as opposed to 42 samples used by the previous model and has a relative training time of 0.78. Performance of this classifier on the evaluation set is shown in Table 3, which gives the Sine/Cosine, period, and overall classification accuracies as 92.6%, 86.4%, and 80%, respectively. Such simplification of the monolithic classifier degrades all three measures of classification accuracy.

C. Smaller Training Set

We have also investigated the effect of reducing the size of the training set on classifier performance. In some applications, e.g. medical diagnostics, training examples may be scarce due to the difficulty or high cost involved in obtaining solved examples. Instead of training on 2000 waveform records (500 of each pattern) as described earlier, we restricted training to only 400 records (100 of each pattern), with the same evaluation set of 1000 records (250 of each pattern) used to evaluate the models. This presented a grater challenge to the resulting classifiers, as they

would be evaluated on a population 2.5 times larger than the training population. Table 2 shows the classifier obtained using the smaller training set with the default complexity of CPM = 1. Although the model still uses 4 layers, it is simpler than that of the default model in the top row of the table. The first layer is just a White element and only 31 waveform samples are used as opposed to 42 samples before. A simpler model is expected since it is easier to reconcile input-output variations within a smaller training set. The relative training time drops to 0.29. Table 3 shows the classification performance and gives the Sine/Cosine, period, and overall classification accuracies as 91.2%, 90.6%, and 82.7%, respectively. The overall classification accuracy exceeds that of the simpler model trained on the full training set (NT = 2000, CPM = 5).

D. Simpler model with a Smaller Training Set

Table 2 shows the classifier obtained using the smaller training set with CPM set to 5 for a simpler model. This leads to a significant drop in model complexity, with the classifier consisting of only two elements organized in two layers and using only four waveform samples as inputs. The relative training time drops to 0.23. Table 3 shows the classification performance and gives the Sine/Cosine, period, and overall classification accuracies as 89.6%, 65.7%, and 58.9%, respectively. While performance remains reasonably adequate for the simpler task of sine/cosine discrimination, over-simplification of the model causes significantly poorer period recognition and overall classification. It is obvious that the monolithic learning of the classification problem at hand is too complex to be supported adequately with the level of model simplification attempted (NT = 400, CPM = 5). The monolithic approach does not allow us to take advantage of desirable benefits of model simplification (e.g. smaller training sets, reduced training times, simpler and smaller model networks, and faster classification) as this adversely affects classification performance.

We use the z statistic to test the statistical significance of the difference in performance levels exhibited by two model configurations. The statistic tests the hypothesis that the means of the

output absolute error for two error distributions are equal, and therefore belong to the same distribution, and is expressed as [23]:

$$z = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma^2{}_1}{n_1} + \frac{\sigma^2{}_2}{n_2}}} \tag{14}$$

where μ_i, σ_i, and n_i are the error mean and standard deviation, and the sample size for the *i*th error distribution, respectively, and $i \in 1,2$. At the 95% confidence level ($\alpha = 0.05$), the hypothesis is accepted for $-1.96 < z < 1.96$. Performing this test on the absolute error data in Table 3 (with $n_1 = n_2 = 1000$) between data for the top row and data for each of the remaining three rows, the hypothesis is rejected in all three cases. Therefore, degradations in classification performance by the default model at the top row of the table (NT=2000, CPM=1) attributed to model changes introduced in the remaining three rows are all statistically significant. Such changes reduce model complexity or the training set size, or both, by a factor of 5.

V. CLASSIFICATION WITH PROBLEM DECOMPOSITION

We adopt a manual (explicit) approach based on knowledge of the problem to decompose the classification of the four waveform patterns into a set of simpler tasks. A straight-forward class decomposition approach would use four separate 2-class classifier modules, each trained to separate each of the four patterns from its complement [5]. This method has the advantage that all four modules are trained and interrogated independently in parallel, which simplifies development and operation, and speeds up both training and classification. A serious limitation is the gross imbalance in the composition of the training sets for individual modules. Examples representing a class would be only one third of those representing its complement, which is expected to slow down training and reduce classification accuracy [5]. To avoid such limitations we use a hierarchical 2-stage decomposition where the waveform type is first determined as either sine or cosine and then the waveform is classified as having one of the two waveform periods (*T1* or *T2*). By simplifying the learning tasks one hopes that, compared to the monolithic approach, the

resulting models would perform more accurately and that simpler models would still give adequate performance. In place of a single monolithic network, the decomposition classifier requires three network modules as shown in Fig. 2. Network Net1 performs the initial sine/cosine waveform type classification and networks Net2 and Net3 perform the period classification for the classified sine and cosine waveforms, respectively. Net2 is activated only when Net1 detects a sine waveform, while Net3 is activated only for patterns classified as cosines by Net1. To compare classification performance with and without problem decomposition, the same data sets used for training and evaluating the monolithic classifiers (without decomposition) were used to train and evaluate decomposition classifier. Net1 was trained on all the 2000 training records, but with those records identified only as either sines or cosines. Net2 was trained on the 1000 sine records in the training set, with those records identified as having a period *T1* or *T2*. Similarly, Net3 was trained on the cosines of the training data. All three classifier modules are thus trained using balanced datasets, where the classes being classified are equally represented. A typical data record for the training/evaluation of Net1 is represented as:

Inputs:	Output:
Values of the Waveform Samples	Waveform Type (Categorical Variable)
y(1) y(2) y(3) … y(50)	1 (= Sine) 2 (= Cosine)

Records for use with Net2 would take the form:

Values of the Waveform Samples (all sines)	Waveform Period (Categorical Variable)
y(1) y(2) y(3) … y(50)	1 (Period = *T1* = 25) 2 (Period = *T2* = 30)

Similarly for network Net3 for cosines.

The three dedicated classifier networks synthesized with NT = 2000 and CPM = 1 are shown in top row of Table 4. As each network solves a simpler problem than that of the monolithic solution, the modules are considerably simpler than the corresponding monolithic network in the top row of Table 2. The only functional elements used are the White and the Single elements, and the period

classifier for cosines uses only 3 layers. This should considerably simplify the mathematical expression for the models as compared to the corresponding monolithic network dominated by the more complex Triple element. The sine/cosine classifier uses the largest number of waveform samples for inputs (= 34) while the two period classifiers for sines and cosines use only 29 inputs each. It is interesting to note that in all cases represented in Table 4, the period classifiers for sines and cosines use an identical number of waveform samples. The number of inputs used by the individual networks is always lower than the number of inputs used by the monolithic network in Table 2. Relative to the training time for the monolithic network, the training times for Net1, Net2, and Net3 are given in Table 4 as 0.65, 0.31, and 0.30, respectively. Since training of the three decomposition networks can take place simultaneously, total training time is determined by the longest of the three times, i.e. 0.65. This shows that problem decomposition can effectively reduce training time by a factor of 1.54.

As shown in Fig. 2, classification with problem decomposition takes place in two stages executed sequentially, namely determining the waveform type (sine/cosine) and then recognizing the period type (*T1/T2*) for the waveform type determined. Maximum execution time for a complete classification is the sum of the execution time of the sine/cosine network and the largest of the two execution times for the period classifiers for the sines and the cosines. A rough indicator of the classification execution time is the number of layers in the network, although this does not account for variations in the complexity of functional elements in each layer. For the modular network in the top row of Table 4, network depth corresponding to the maximum classification execution time is 4 + 4 = 8. Compared to the depth of 4 for the corresponding monolithic classifier in Table 2, this suggests that the decomposition solution classifies at half the speed of the monolithic classifier. In practice, however, the simpler function elements and the smaller number of network inputs improve the classification speed of the decomposition classifier.

Performance of the decomposition classifier was measured using the same evaluation set of 1000

waveform records (250 of each of the four patterns). First, the sine/cosine classifier (Net1) was evaluated on the full set but with the waveform records identified only as either sines (500 records) or cosines (500 records). The computed real output from the classifier was converted into a categorical predicted classification output (1 or 2) by rounding according to the following rule:

IF computed output < 1.5 THEN Predicted Classification = 1 (Sine)
IF 1.5 ≤ computed output THEN Predicted Classification = 2 (Cosine) (15)

Net2 and Net3 were evaluated on the waveforms classified by Net1 as sines and cosines, respectively. In both cases, the computed real output from the classifier was converted into a categorical predicted classification output (1 or 2) by rounding according to the following rule:

IF computed output < 1.5 THEN Predicted Classification = 1 (Period = *T1* = 25)
IF 1.5 ≤ computed output THEN Predicted Classification = 2 (Period = *T2* = 30) (16)

Table 5 shows the evaluation results for the modular approach. Only 5 records of each of the sine and cosine patterns were misclassified by Net1. Net2 and Net3 performed the period (*T1/T2*) classification with 100% accuracy for all the waveform records correctly classified by Net1 (495 sine and 495 cosine records, respectively). In practice, Net2 and Net3 would also receive cosine records misclassified as sines and sine records misclassified as cosines, respectively. This was taken into account when calculating the values shown in Table 5 for the mean and standard deviation of the absolute error between the predicted output (before rounding) and the known classification output. Comparison between the results in Tables 3 and 5 shows improvement in all aspects of classification performance with problem decomposition, with the overall classification accuracy rising from 87.2% to 99%. Performing the z statistic test on the absolute error data in the top row of both Tables 3 and 5 shows that the error reduction attributed to problem decomposition is statistically significant.

VI. POSSIBLE TRADE-OFFS FOR PROBLEM DECOMPOSITION GAINS

Results given in Section V indicate that problem decomposition leads to a significant improvement

in classification accuracy and some reduction in training time, at the expense of some possible increase in the classification execution time. In many situations, classification accuracy is the prime concern and improvements therein would be most welcome even if obtained with more computationally demanding training or somewhat slower classification. Abundance of low cost, high speed computing power that can be used in parallel alleviates such limitations for most practical applications. However, in some demanding online applications we may be willing to sacrifice some classification accuracy for improved classification speed. Another improvement area that would be welcome in many disciplines is reducing the size of the data set required to train the classifier. We have investigated a number of scenarios for trading-off some of the improvement in prediction accuracy gained from problem decomposition for other benefits, such as further model simplification, reductions in training time and computational requirements, improvement in classification speed, and reduction in the size of the training data set required.

A. Simpler Modular Classifiers

With a full training set (NT = 2000), we enforced the synthesis of simpler networks for the decomposition solution by setting the CPM parameter to 5 instead of the default value of 1. The expected simpler network structures have the advantages of reduced computational requirements and faster classification. Table 4 shows the models obtained for Net1, Net2, and Net3. Compared with the corresponding networks using CPM = 1 (top row in the same table), it is clear that the modules became much simpler. For example, the period classifier for cosines is now only a 2-layer network operating on just 10 waveform samples. For CPM = 1, the corresponding network has 3 layers and operates on 29 samples. With the training times for Net1, Net2, and Net3 shown in the table, the relative overall total training time for the decomposition solution networks (if performed in parallel) is 0.57, indicating a reduction in training time by a factor of 1.75 from the monolithic case with CPM = 1. Table 5 shows the values of the three classification performance parameters as 98.6%, 99.7%, and 98.3%. These values are very close to those for the decomposition case with CPM = 1, which demonstrates that problem decomposition allows significant model simplification

while maintaining adequate classification accuracy. This has not been the case with the monolithic approach, where model simplification reduced the overall classification accuracy from 87.2% to 80%, see Table 3.

B. Smaller Training Set

Another area where improved classification accuracy through problem decomposition can be traded off is reducing the size of the training set required. This should prove useful in situations where training examples are scarce due to the difficulty or high cost of obtaining solved examples. Instead of training on 2000 waveform records (500 of each pattern) as described in Section V, we restricted the training to 400 records (100 of each pattern). However, the evaluation set remained the same, 1000 records (250 of each pattern). Table 4 shows the models obtained for Net1, Net2, and Net3 at the default model complexity (CPM = 1). Compared with the networks obtained using the full training set of 2000, it is clear that networks became much simpler. For example, the sine/cosine classifier is now a 3-layer network operating on 19 waveform samples as compared to a 4-layer network operating on 34 samples. With the training times for Net1, Net2, and Net3 shown, the relative overall total training time for the decomposition solution networks (if performed in parallel) is 0.20, indicating a reduction in training time by a factor of 5 compared to the monolithic case with CPM = 1. Table 5 shows the classification performance parameters as 98.7%, 99.6%, and 98.3%. These are almost identical to those with NT = 2000 and CPM = 1, which demonstrates that a 5-fold reduction in the size of the training set is quite affordable with problem decomposition. This has not been the case with the monolithic approach, where model simplification reduced the overall classification accuracy from 87.2% to 82.7%, see Table 3.

C. Simpler Modular Classifiers with a Smaller Training Set

Further gains in model simplicity and reduction in classification time, training time and the training data required can be obtained using simpler classifiers (CPM = 5) that are trained on the smaller training set (NT = 400). Table 4 shows the models obtained for Net1, Net2, and Net3. Compared

with other rows in the table, the networks are the simplest obtained so far, each being a single functional element operating on just 3 samples of the waveform. The most complex is the sine/cosine classifier that uses a Triple element, while the period classifiers for both the sines and the cosines use a linear White element. Classification time would be the fastest obtained so far, being determined by the execution time of only two layers, as opposed to eight layers for the decomposition approach with NT = 2000 and CPM = 1 (top row of Table 4). With the training times for Net1, Net2, and Net3 shown in Table 4, the relative overall total training time for the decomposition solution networks (if performed in parallel) is 0.09, indicating a reduction in training time by a factor of 11 compared to the monolithic case with NT = 2000 and CPM = 1. The simpler networks allow simple analytical expressions to be derived for the classifier. As an example, Fig. 3 shows details of the equations for the function elements of the sine/cosine Net1 classifier. Substituting symbolically for these equations gives the following overall relationship for the classifier output in terms of the relevant waveform samples:

$$\begin{aligned}\text{Output} = {} & -0.04331\, y(9) - 0.040745\, y(10) - 0.047598\, y(12) + 0.0016083\, y(9)\, y(10) \\ & - 0.00013939\, y(9)\, y(12) - 0.0011048\, y(10)\, y(12) + 0.0011623\, y(9)\, y(10)\, y(12) \\ & + 1.4798 \end{aligned} \tag{17}$$

Sine/cosine classification is then performed by rounding the output according to the rule in Equation (15). For example, with $A = 5$ and $T = 25$, a noiseless sine has y(9) = 3.852566, y(10) = 2.938926, and y(12) = 0.626666, while a noiseless cosine has y(9) = -3.187120, y(10) = -4.045085, and y(12) = -4.960574. Substituting for both cases in Equation (17) gives the computed output as 1.19 and 1.94, respectively, leading to correct classification in both cases.

Table 5 shows the three classification performance parameters as 93.1%, 96%, and 89.4%. While these are significantly poorer than all other problem decomposition results in Table 5, they are still better than nearly all the monolithic results in Table 3. In particular, the overall classification accuracy is still better than that for the default monolithic solution (NT = 2000, CPM = 1). This

demonstrates that problem decomposition makes it affordable to combine significant model simplification with considerable reduction in the size of the training data without seriously affecting classification performance. This was not possible with the monolithic approach, where the overall classification accuracy deteriorated sharply from 87.2% to the unacceptably low value of 58.9%, see Table 3. Tests using the z statistic shows that performance for each row in Table 5 (with problem decomposition) is significantly superior to that of the corresponding row in Table 3 for the monolithic case. Moreover, performance of each of the problem decomposition models in rows 2 and 3 of Table 5 is significantly superior to that of the default monolithic model (NT=2000, CPM=1) in the top row of Table 3.

VII. COMPARISON WITH OTHER TECHNIQUES

We have compared the performance of the abductive problem decomposition approach proposed with another abductive network method for improving classification accuracy based on network committees [18]. A committee of five abductive networks was used. The five member networks were trained using CPM = 1 on different and equal subsets of the training set of 2000 records. Therefore for each network, NT = 400 and each waveform class was represented with 100 cases. The committee was evaluated using the same evaluation set of 1000 cases. Overall classification accuracies for individual member networks were 82.7%, 86.9%, 82.3%, 85.1%, and 82.1%. The final committee output was determined by simple averaging of the raw outputs of the five individual member networks, followed by rounding according to Equation (13). Overall classification accuracy for the committee was 88.4%. Table 6 lists performance figures for the committee approach as well as the monolithic and problem decomposition approaches for comparison. Compared an improvement of 11.8 percentage points in the overall classification accuracy by problem decomposition, the network committee gives a much smaller improvement of only 1.2 points. In the latter case, change in the mean absolute error is not statistically significant. Moreover, error reduction by problem decomposition over the committee approach is statistically

significant.

Comparison was also made with both monolithic and problem decomposition neural network models developed and evaluated on the same data used for the abductive models. The backpopagation neural network models were developed using the PathFinder neural network software for Windows. Networks were trained trained and evaluated using the same data used with the corresponding abductive models. 20% of the training data were used for cross validation. All neural models had one hidden layer containing 6 neurons with a sigmoid transfer function. Performance results are shown in the last two rows of Table 6. The monolithic neural model outperforms the monolithic abductive model, with the overall classification accuracy being 93.9% and 87.2%, respectively, and reduction in the mean absolute error is statistically significant. However, problem decomposition solutions using the two techniques give nearly the same performance, and variations in the mean absolute errors are not statistically significant. In both cases, the period classifiers give 100% classification accuracy for waveforms correctly classified by the sine/cosine classifier stage.

VIII. DISCUSSION

Table 7 summarizes the results obtained by listing data on model complexity, training time, and overall classification accuracy. Model complexity is described in terms of the CPM parameter, the number of waveform samples used as classifier inputs, and the model depth in layers to execute a classification. Training time is relative to that of the default monolithic classifier (NT=2000, CPM =1). With problem decomposition, training of the three classifier modules is assumed to take place in parallel, and training time is taken as the longest of the three classifier modules. Data in the table indicates that for the same size of the training set and CPM value, problem decomposition always results in significantly higher classification accuracies and lower training times. These improvements are attributed to the simpler modular networks for the problem decomposition solution, which generalize better, train faster, and allow training to be performed in parallel. Such improvements are more significant with simpler models obtained with smaller training sets, higher

CPM values, or both. For example, with NT = 400 and CPM = 5, problem decomposition increases classification accuracy by 51.8% and reduces training time by 60.9% as compared to only 13.5% and 35%, respectively for NT = 2000 and CPM = 1. With task simplification brought about by problem decomposition, the approach is much more tolerant to model simplification as compared to the monolithic approach. A five-fold reduction in training set size combined with a five-fold increase in the CPM parameter decrease classification accuracy of the problem decomposition classifier by only 9.7% as compared to 32.4% for the monolithic classifier. This forms the basis for trading off improved classification accuracy benefits due to problem decomposition for other advantages such as further model simplification (with faster training and faster classification) and reduction in the amount of training data required which can be very useful in many situations. Forcing the synthesis of simpler models is far easier to implement with abductive networks (specifying a higher CPM value) as compared to neural networks. Serial execution of modules in a problem decomposition classifier implies slower classification, as suggested by the larger values for the classifier depth in Table 7. However, layers of the decomposition classifier modules are made up of much simpler functional elements operating on fewer waveform inputs, which speeds up execution compared to the corresponding monolithic classifier. Moreover, this problem becomes less severe with simpler models. For example, both the monolithic and problem decomposition solutions for NT = 400 and CPM = 5 have the same depth of 2 layers and because of its simpler function elements, the problem decomposition classifier would actually execute faster. The problem decomposition technique described gives superior performance to that of a an abductive network committee and a comparable performance to that of a neural network scheme developed on the same data using the same approach to problem decomposition

ACKNOWLEDGEMENTS

The author wishes to acknowledge the support of the Research Institute of King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

REFERENCES

[1] R. E. Abdel-Aal, Automatic fitting of Gaussian peaks using abductive machine learning, IEEE Transactions on Nuclear Science 45 (1998) 1-16.

[2] R. E. Abdel-Aal and M. Raashid, Using abductive machine learning for online vibration monitoring of turbo molecular pumps, Shock and Vibration 6 (1999) 253-265.

[3] AbTech Corporation, Charlottesville, VA, USA, AIM User's Manual, 1990.

[4] A. P. Alves da Silva, U. P. Rodrigues, A. J. Rocha Reis, and L. S. Moulin, NeuroDem - a neural network based short term demand forecaster, IEEE Power Tech. Conf., Porto, Portugal, 2001.

[5] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, Efficient classification for multiclass problems using modular neural networks**,** IEEE Transactions on Neural Networks 6 (1995) 117-124.

[6] A. R. Barron, Predicted squared error: A criterion for automatic model selection. Self-Organizing, in S. J. Farlow, ed., Self-Organizing Methods in Modeling: GMDH Type Algorithms, (Marcel-Dekker, New York, 1984) 87-103.

[7] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123-140.

[8] D. Edelman, P. Davy, and Y. L. Chung**,** Using neural network prediction to arbitrage the Australian All-Ordinaries Index**,** Third International Conference on Knowledge-Based Intelligent Information Engineering Systems (1999) 166–169.

[9] H. M. El-Bakry, Face detection using neural networks and image decomposition, Proceedings of the International Joint Conference on Neural Networks (2002) 1045 –1050.

[10] S. J. Farlow, The GMDH algorithm, in S. J. Farlow, ed., Self-Organizing Methods in Modeling: GMDH Type Algorithms, (Marcel-Dekker, New York, 1984) 1-24.

[11] K. Fukunaga, Introduction to Statistical Pattern recognition, (Morgan Kaufmann, San Mateo, CA, 1990).

[12] W. Y. Goh, C. P. Lim, and K. K. Peh, Predicting drug dissolution profiles with an ensemble of

boosted neural networks: a time series approach, IEEE Transactions on Neural Networks 14 (2003) 459-463.

[13] A. A. Ghorbani and K. Owrangh, Stacked generalization in neural networks: generalization on statistically neutral problems, International Joint Conference on Neural Networks (2001) 1715-1720.

[14] S. –U. Guan and S. Li, Parallel growing and training of neural networks using output parallelism, IEEE Transactions on Neural Networks 13 (2002) 542–550.

[15] D. -S. Huang and Z. Chi, Neural networks with problem decomposition for finding real roots of polynomials**,** Proceedings of the International Joint Conference on Neural Networks (2001), A25-A30.

[16] Y. -C. Huang, H. -T. Yang, and K.-Y. Huang, Abductive network model-based diagnosis system for power transformer incipient fault detection, IEE Proceedings on Generation, Transmission and Distribution 149 (2002) 326–330.

[17] A. A. Ilumoka, Efficient and accurate crosstalk prediction via neural net-based topological decomposition of 3-D interconnect**,** IEEE Transactions on Advanced Packaging 24 (2001) 268–276.

[18] R. A. Jacobs and M. I. Jordan, Adaptive mixtures of local experts, Neural Computation 3 (1991) 79-87.

[19] R. E. Jenkins and B. P. Yuhas, A simplified neural network solution through problem decomposition: The case of the truck backer-upper, IEEE Transactions on Neural Networks 4 (1993) 718-720.

[20] D. Jimenez, Dynamically weighted ensemble neural networks for classification, IEEE World Congress on Computational Intelligence (1998) 753-756.

[21] B. Y. Lee and Y. S. Tarng, An abductive network for predicting tool life in drilling, IEEE Transactions on Industry Applications 35 (1999) 190-195.

[22] P. Liang, Problem decomposition and subgoaling in artificial neural networks, IEEE International Conference on Systems, Man and Cybernetics (1990) 178 –181.

[23] W. Mendenhall and R. J. Beaver, Introduction to Probability and Statistics, (Duxbury Press, Belmont, CA, 1994).

[24] G. J. Montgomery and K. C. Drake, Abductive Networks, SPIE Applications of Artificial Neural Networks Conference (1990) 56-64.

[25] I. T. Podolak, S.-L. Lee, A. Bielecki, E. Majkut, A hybrid neural system for phonematic transformation, 15th International Conference on Pattern Recognition (2000) 957-960.

[26] P. S. Prampero and A.C.P.L. De Carvalho, Recognition of vehicles silhouette using combination of classifiers**,** IEEE International Joint Conference on Neural Networks (1998) 1723-1726.

[27] P. S. Prampero and A.C.P.L. De Carvalho, Classifier combination for vehicle silhouettes recognition, Seventh International Conference on Image Processing And Its Applications (1999) 67-71.

[28] L. Y. Pratt and C. A. Kamm, Improving a phoneme classification neural network through problem decomposition, IJCNN-91-Seattle International Joint Conference on Neural Networks (1991) 821-826.

[29] V. Radevski and Y. Bennani, Reliability control in committee classifier environment, IEEE-INNS-ENNS International Joint Conference on Neural Networks (2000) 561-565.

[30] V. R. Reddy and Y. A. Pachepsky, Predicting crop yields under climate change conditions from monthly GCM weather projections, Journal of Environmental Modeling and Software 15 (2000) 79-86.

[31] M. Su, M. Basu, Gating improves neural network performance**,** IEEE International Joint Conference on Neural Networks (2001) 2159–2164.

[32] S. Tadjudin and D. A. Landgrebe, A decision tree classifier design for high-dimensional data with limited training samples, IGARSS '96 International Geoscience and Remote Sensing Symposium (1996) 790-792.

[33] H. Wein and O. Ozdamar, Auditory brainstem response classification using modular neural networks, International Conference of the IEEE Engineering in Medicine and Biology Society (1991) 1879-1880.

[34] X. Yao, M. Fischer, and G. Brown, Neural network ensembles and their application to traffic flow prediction in telecommunications networks, International Joint Conference on Neural Networks (2001) 693-698.

Table 1. Pair-wise values for two separability measures for the four waveform classes and the corresponding upper bound on the Bayes error. 500 records for each class.

Class Pair	Bhattacharyya Distance, B	J_4 Separability Measure	Upper Bound on Bayes error $\varepsilon_u = 0.5\ e^{-B}$
1-2	7.57	0.269	2.57×10^{-4}
1-3	13.59	0.504	6.28×10^{-7}
1-4	3.78	0.119	1.14×10^{-2}
2-3	23.88	0.837	2.12×10^{-11}
2-4	12.67	0.462	1.57×10^{-6}
3-4	9.00	0.342	6.20×10^{-5}

Table 2. Networks for monolithic classifiers synthesized with two sizes of the training set at two levels of model complexity. Shown also are the number of different waveform samples selected as model inputs and the training time relative to the network at the top row.

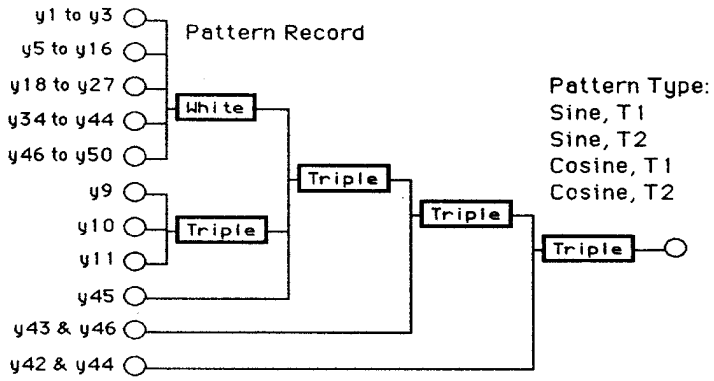
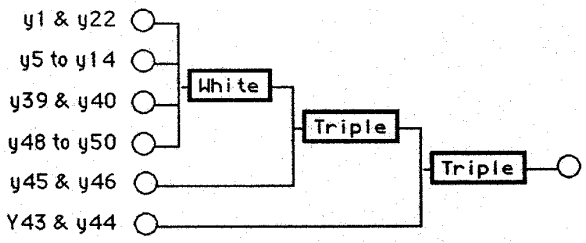
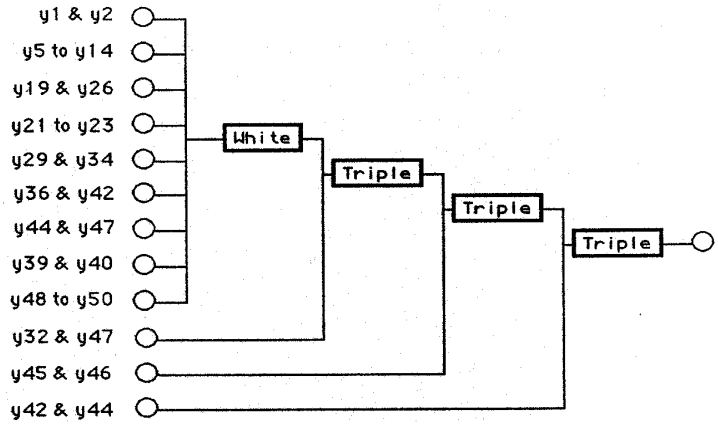
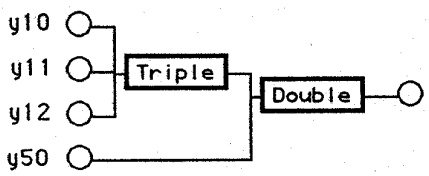
Training Set Size, NT	CPM	Monolithic Classifier Network	Number of Inputs	Relative Training Time
2000	1		42	1.0
	5		21	0.78
400	1		31	0.29
	5		4	0.23

Table 3. Performance of the monolithic classifier for two sizes of the training set and two levels of model complexity.

NT	CPM	Classification Accuracy, %			Output Absolute Error	
		Sine/Cosine	Period	Overall	Mean, μ	Standard Deviation, σ
2000	1	95.9	90.9	87.2	0.249	0.273
	5	92.6	86.4	80	0.328	0.311
400	1	91.2	90.6	82.7	0.295	0.320
	5	89.6	65.7	58.9	0.619	2.451

Table 4. Networks for problem decomposition classifiers synthesized with two sizes of the training set at two levels of model complexity. a = the number of different waveform samples selected as model inputs and b = the training time relative to the monolithic network at the top row of Table 2.

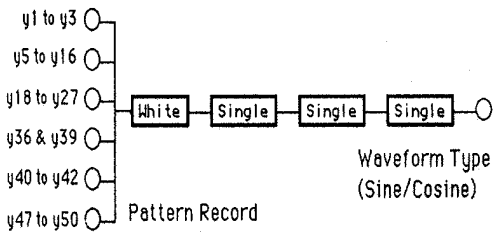
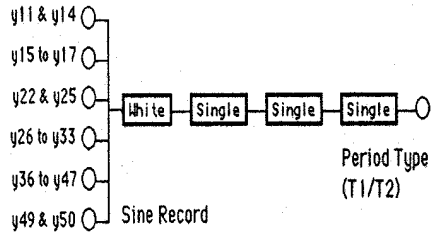
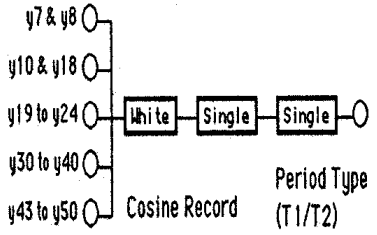
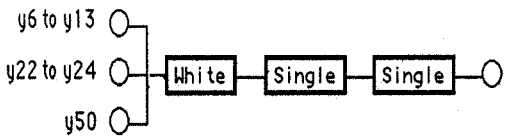
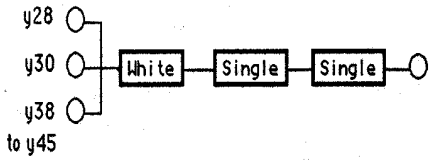
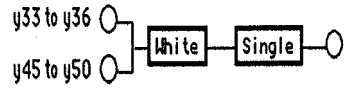
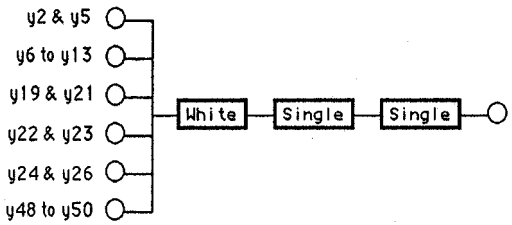
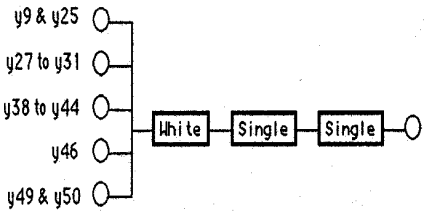
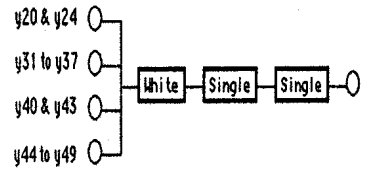
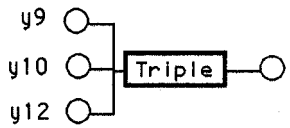
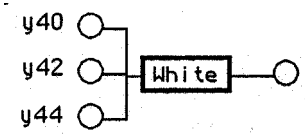
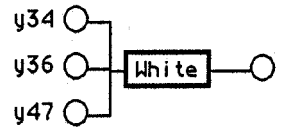
Training Set Size, NT	CPM	Decomposition Classifier Networks								
		Sine/Cosine Classifier (Net1)	a	b	Period Classifier for Sines (Net2)	a	b	Period Classifier for Cosines (Net3)	a	b
2000	1		34	0.65		29	0.31		29	0.30
	5		12	0.57		10	0.31		10	0.24
400	1		19	0.20		17	0.11		17	0.11
	5		3	0.09		3	0.04		3	0.05

Table 5. Performance of the modular (problem decomposition) classifier for two sizes of the training set and two values of model complexity.

NT	CPM	Classification Accuracy, %			Output Absolute Error	
		Sine/Cosine	Period	Overall	Mean, μ	Standard Deviation, σ
2000	1	99	100	99	0.011	0.057
	5	98.6	99.7	98.3	0.054	0.115
400	1	98.7	99.6	98.3	0.027	0.098
	5	93.1	96	89.4	0.233	0.211

Table 6. Performance comparison between abductive network problem decomposition, abductive network committee, and neural network problem decomposition for the same data.

Technique	Approach	Classification Accuracy, %			Output Absolute Error	
		Sine/Cosine	Period	Overall	Mean, μ	Standard Deviation, σ
Abductive (NT=2000, CPM=1)	Monolithic	95.9	90.9	87.2	0.249	0.273
	Problem Decomposition	99	100	99	0.011	0.057
Abductive (NT=5 x 400, CPM=1)	5-Member Committee	95.1	93	88.4	0.250	0.271
Neural Network (50-6-1)	Monolithic	99.3	94.6	93.9	0.149	0.245
	Problem Decomposition	99.7	100	99.7	0.008	0.057

Table 7. Summary of comparison results between monolithic and problem decomposition classifiers for two sizes of the training set at two levels of model complexity.

Modeling Approach	Training Set Size, NT	Model Complexity			Relative Total Training Time[3]	Overall Classification Accuracy, %
		CPM	Number of inputs[1]	Total Classifier Depth[2]		
Monolithic	2000	1	42	4	1	87.2
		5	21	3	0.78	80
	400	1	31	4	0.29	82.7
		5	4	2	0.23	58.9
Problem Decomposition	2000	1	34	8	0.65	99
		5	12	6	0.57	98.3
	400	1	19	6	0.20	98.3
		5	3	2	0.09	89.4

[1] For a problem decomposition solution, this is the largest number of inputs among the three classifier modules.

[2] Maximum number of layers for executing a classification.

[3] For a problem decomposition solution, we assume that training of the three classifier modules is performed in parallel.

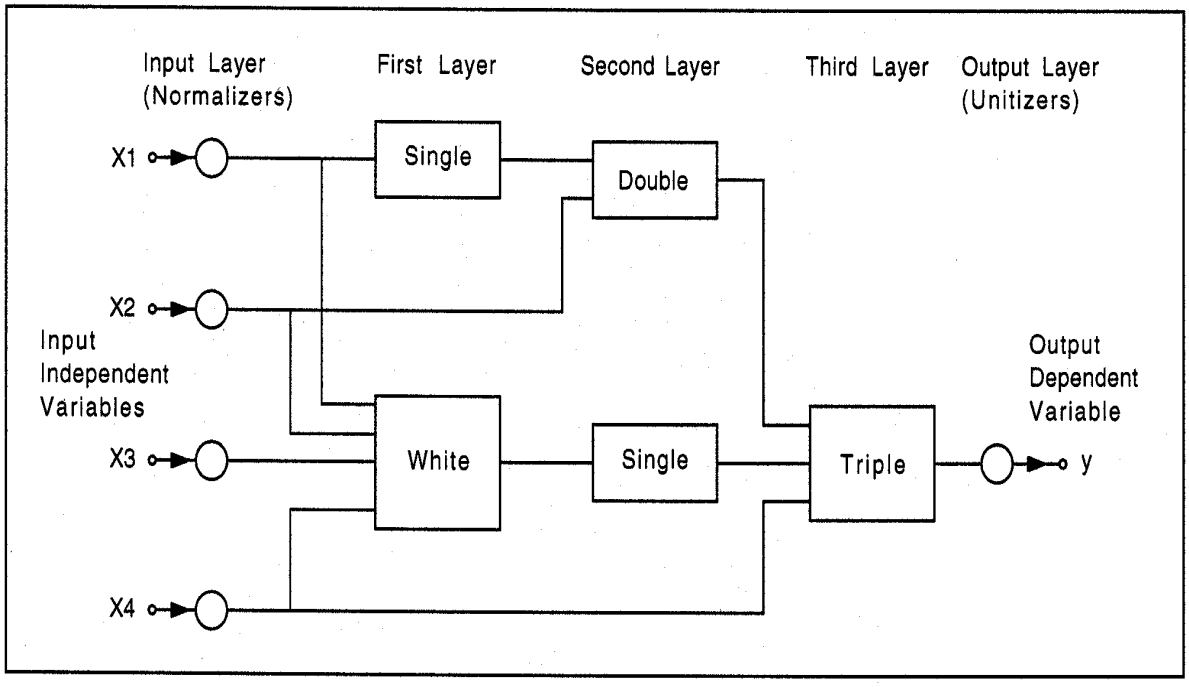


Fig. 1. AIM Abductive network example.

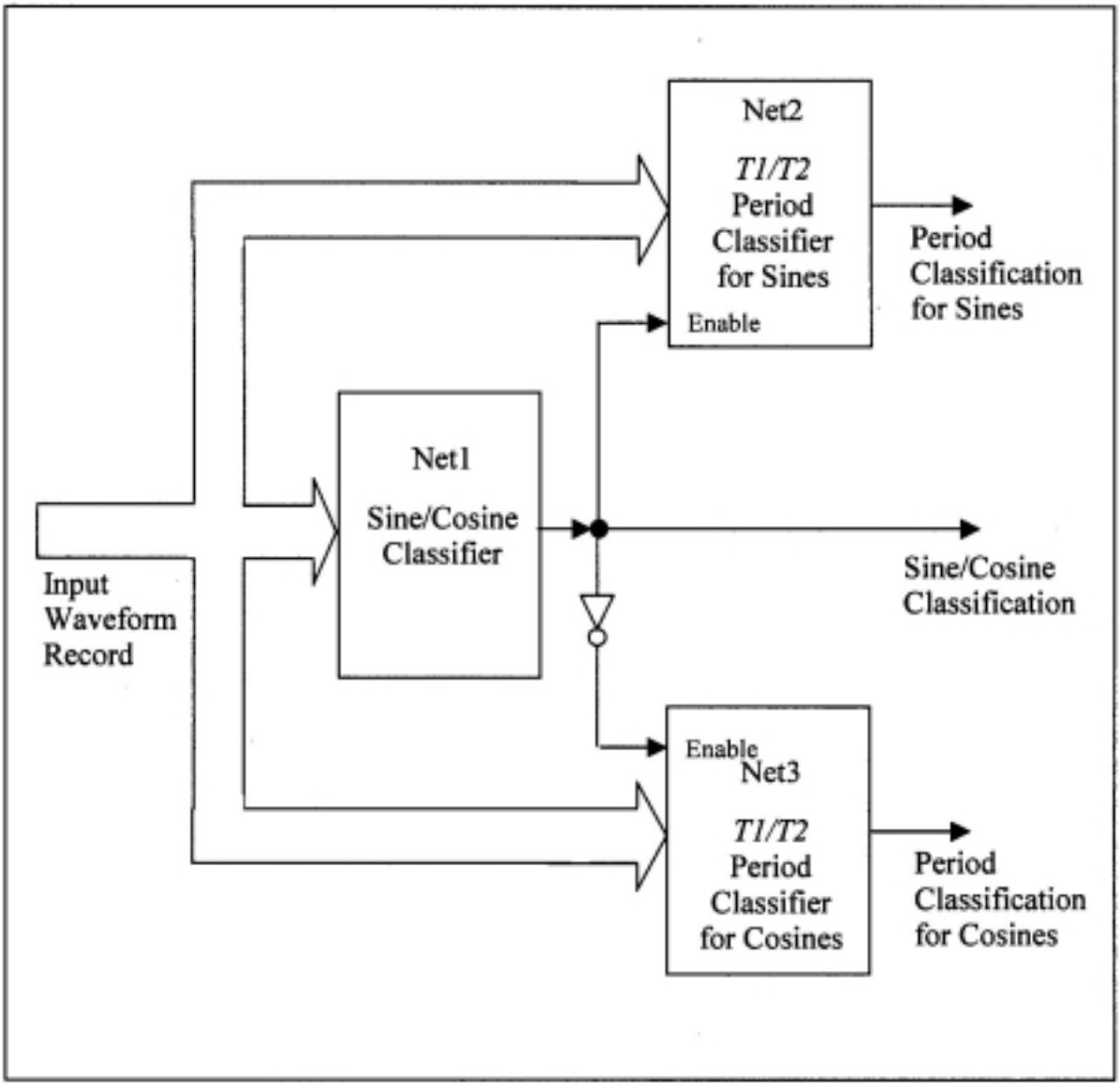


Fig. 2. Block diagram of the modular (problem decomposition) classifier.

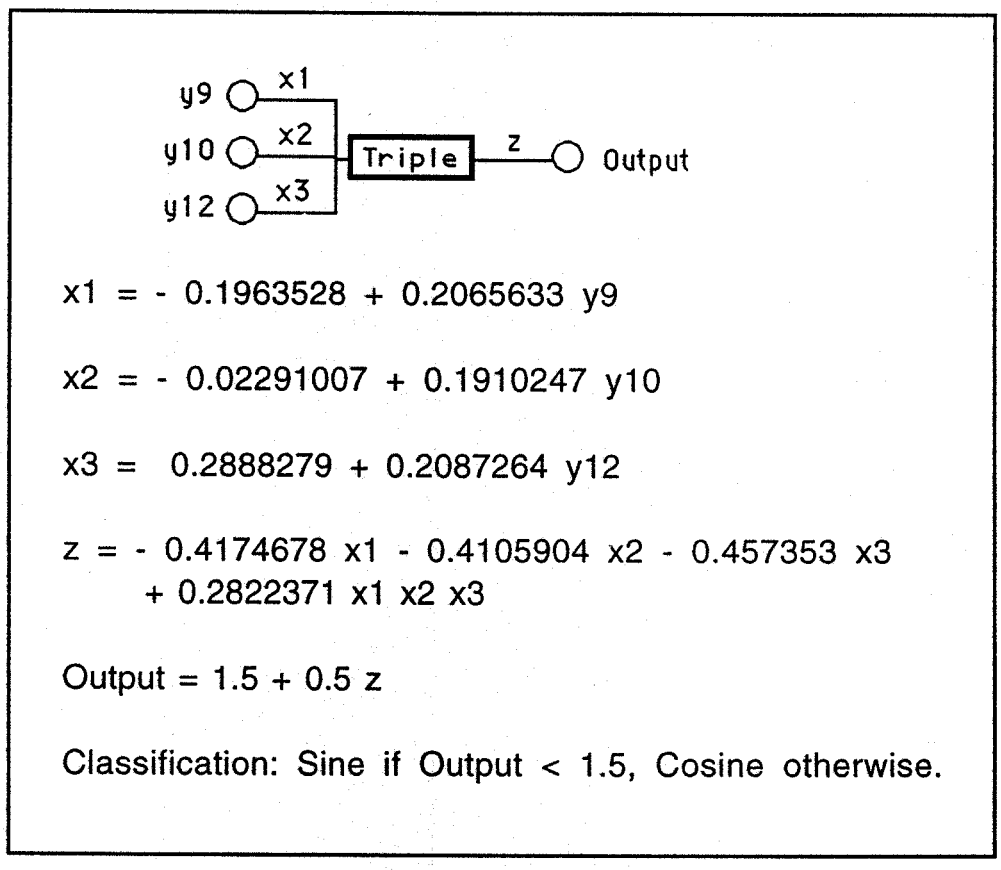


Fig. 3. Details of the Sine/Cosine classifier in the third row of Table 4 (NT = 400, CPM = 5).