4-1-1992

CONJUGATE-GRADI3NT NEURAL NETWORKS IN CLASSIFICATION OF MULTISOURCE AND VERY-HIGH-DIMENSIONAL REMOTE SENSING DATA

Jon A, Benediktsson
University of Iceland, Department of Electrical Engineering and Laboratory for Information Technology and Signal Processing

Philip H. Swain
Purdue University School of Electrical Engineering

Okan K. Ersoy
Purdue University School of Electrical Engineering

CONJUGATE-GRADIENT NEURAL NETWORKS IN CLASSIFICATION OF MULTISOURCE AND VERY-HIGH-DIMENSIONAL REMOTE SENSING DATA

Jon A, Benediktsson
Department of Electrical Engineering
and
Laboratory for Information Technology and Signal Processing
University of Iceland
Hjardarhaga 2-6, 107 Reykjavik, ICELAND

Philip H. Swain and Okan K. Ersoy
School of Electrical Engineering
and
Laboratory for Applications of Remote Sensing
Purdue University
West Lafayette, IN 47907, U.S.A.

ABSTRACT

Application of neural networks to classification of remote sensing data is discussed. Conventional two-layer backpropagation is found to give good results in classification of remote sensing data but is not efficient in training. A more efficient variant, based on conjugate-gradient optimization, is used for classification of multisource remote sensing and gedgraphic data and very-high-dimensional data. The conjugate-gradient neural networks give excellent performance in classification of multisource data but do not compare as well with statistical methods in classification of very-high-dimensional data.

CONJUGATEGRADIENT NEURAL NETWORKS IN CLASSIFICATION OF MULTISOURCE AND VERY-HIGH-DIMENSIONAL REMOTE SENSING DATA

1. INTRODUCTION

Great interest has been shown recently in classification of remotely sensed data using neural networks. Several researchers have applied neural network classifiers to such data: Benediktsson et al. (1990b) used. two-layer backpropagation networks to classify multisource remote sensing and geographic data and compared the results to the performance of several statistical methods. McClelland et al. (1989) used a two-layer backpropagation algorithm to classify Landsat TM (Thematic Mapper) data. Decatur (1989a, 1989b) used two-layer backpropagation, learning vector quantization (LVQ) and adaptive resonance theory (ART) networks to classify Synthetic Aperture Radar (SAR) data and compared the results to the results of Bayesian classification. Ersoy et al. (1990) developed a hierarchical neural network (PSHNN) which they applied to classification of aircraft multispectral scanner data and multisource data. Heermann et al. (1990) used two-layer backpropagation to classify multitemporal data. Maslanik et al. (1990) used two-layer neural networks to classify Scanning Multichannel Microwave Radiometer (SMMR) passive microwave data. All of these researchers have reported promising performance by neural networks, but the neural networks have been found to be slow in training as compared to statistical methods.

This research was supported in part by the National Aeronautics and Space Admininistration (NASA) through Grant No. NAGW-925.

Faster training methods are thus attractive for classification of remotely sensed data.

In this paper, "fast" neural networks are investigated. The neural network methods are applied to classification of multisource remote sensing/geographic data and very-high-dimensional remote sensing data. In this research, the principal reason for using neural network methods for classification of multisource remote sensing/geographic data is that these methods are distribution-free. Since multisource data are in general of multiple types, the data from the various sources can have diierent statistical distributions. The neural network approach does not require explicit modeling of the data from each source. In addition, neural network methods have been shown to approximate class-cohditional probabilities in the mean-squared sense (Wan 1990). Consequently, there is no need to treat the data sources independently as in many statistical methods (Benediktsson et al. 1990b). The neural network approach also avoids the problem in statistical multisource analysis of specifying how much influence each data source should have in the classification (Benediktsson et al. 1990b).

A problem with conventional multivariate Gaussian statistical classification of very-high-dimensional data is that this method relies on having nonsingular (invertible) class-specific covariance matrices. When n features are uaed, the training samples for each class must include at least n+1 different samples so that the covariance matrices are nonsingular; in high-dimensional cases involving limited training samples the matrices may be singular. In this paper, we explore the feasibility of using neural networks for classification of very-high-dimensional data in order to avoid this problem.

The paper begins with a general discussion of neural networks used for pattern recognition, followed by a discussion of well-known neural network models. Next, optimization techniques for the neural network models are addressed with the goal of making the training procedures for the networks more efficient. Finally, classification results are given for multisource remote sensing data and very-high-dimensional data.

2.. NEURAL NETWORK METHODS FOR PATTERN RECOGNITION

A neural network is an interconnection of neurons, where a neuron can be described in the following way: A neuron receives input signals x_j, $j = 1,2,...,N$, which represent the activity at the input or the momentary frequency of neural impulses delivered by another neuron to this input (Kohonen 1988). In the simplest formal model of a neuron, the output value or the frequency of the neuron, o, is often represented by a function

$$o = K\,\phi(\sum_{j=1}^{N} w_j x_j - \theta) \tag{1}$$

where K is a constant and 4 is a nonlinear function, e.g., the threshold function which takes the value 1 for positive arguments and 0 (or -1) for negative arguments. The w_j are called *synaptic efficacies* or weights, and θ is a threshold.

In the neural network approach to pattern recognition the neural network operates as a black box which receives a set of input vectors x (observed signals) and produces responses o_i from its output neurons i, $i = 1,...,L$ where L depends on the number of information classes. A general idea followed in neural network theory is that $o_i = 1$ if neuron i is active for the current input vector **x**, or $o_i = 0$ (or -1) if it is inactive. The weights are learned through an adaptive (iterative) training procedure in which a set of training samples is presented at the input (Figure 1). The network gives an output response for each sample. The actual output response is compared to the desired response for the sample and the error between the desired output and the actual output is used to modify the weights in the neural network. The training procedure ends when the error is reduced to a prespecified threshold or cannot be minimized any further. Then all of the data are fed into the network to perform the classification, and the network provides at the output the class representation for each pixel.

Data representation is very important in application of neural network models. It is possible in some problems to use continuous-valued inputs[1] to the neural network but our experience in classification of remotely sensed image data has shown it necessary to increase the network size, e.g., by binarizing the input data when the data dimensionality is low (e.g., less than 10 dimensions). The reason for this binarization is mainly that remote sensing data are very complex and adding extra dimensions to the input data can help in discriminating the data.

A straightforward coding approach used by many researcher:; is to code the input and output by a simple binary coding scheme (0 = 00, 1 = 01, 2 = 10, etc.). However, it is more appropriate to use the Gray-code representation (Lathi 1983) of the input data. The Gray-code representation can be derived from the binary code representation in the following manner: If $b_1\ b_2\ \ldots\ b_n$

1. Using continuous-valued inputs means that the whole value is accepted by a single input neuron; binarieation means each input neuron accepts just one bit of the value.

is a code word in an n - digit binary code, the corresponding Gray-code word $g_1\ g_2 \dots g_n$ is obtained by the rule:

$$g_1 = b_1$$

$$g_k = b_k \oplus b_{k-1} \qquad k \geq 2$$

where $\oplus$ is modulo-two addition. The reason that the Gray-code representation is more appropriate than the binary code for this application is that neighboring integers differ in the Gray-code by only one bit. Adjacent data values in the code space tend to belong to the same information class. When they belong to the same class, the use of the Gray-code representation leads to a smaller number of weight changes, since for values from a given class, most of the input bits are identical.

Using Gray-coded input data has given good experimental results for data of relatively low dimensionality. However, Gray-coding of the data makes the decision regions both more localized and more complex as compared to continuous-valued inputs Figures 2 and 3 illustrate different decision regions for two features of remote sensing data with 4 information classes. The decision regions for continuous-valued input data are more uniform and the use of continuous-valued data can be more successful in generalization especially for very-high-dimensional data with a limited number of training samples. In our research, both Gray-coded and continuous-valued input data were used to see how each input mechanism affected the classification results.

Representation of the output of the neural network is also important. If binary coding is used at the output, the number of output neurons can be reduced to $\lceil \log_2 M \rceil$ where M is the number of information classes.. However, using more output neurons than the minimum $\lceil \log_2 M \rceil$ can make the neural network more accurate in classification. Even though adding more output neurons makes the network larger and therefore computationally more complex, it can also lead to fewer learning cycles, since the Hamming distance (Lathi 1983) of the output representations of different classes can be larger. One output coding mechanism is "temperature coding," in which the representation for n has 1 for its n most significant digits and 0 for the rest (e.g., 4 = 1111000).

However, the most commonly used output representation is the following. The number of output neurons is selected to equal the number of classes, and only one output neuron is active (has the value 1) for each class. For example, in a four class problem, class #1 would be represented by 1000 and class #3 by 0010. This particular representation has the advantage that only one neuron should be active and all of the others should be inactive. Therefore, the "winner take all" principle can be used. Thus, during testing an input sample can be classified to the class which has the largest output response (output responses during testing will be real numbers in the interval from 0 to 1 for each output neuron). If other coding schemes were used for output representation, some samples might need to be rejected in testing since their output would not be close to any of the desired output representations. No such problem is evident with this representation. Therefore, this "winner take all" representation will be used in the experiments reported here.

3. NEURAL, NETWORK MODELS

Several neural network models have been proposed since Rosenblatt (1958) introduced the perceptron in 1952. The perceptron is a one-layer neural network which has the ability to learn and recognize simple patterns. Rosenblatt proved that if the input data are linearly separable, the training procedure of the perceptron will converge and the perceptron can separate the data. However, when the input data are not linearly separable, the decision boundaries may oscillate indefinitely when the perceptron algorithm is applied (Lippman 1987). An adaptation of the perceptron algorithm is the one-layer delta rule.

The delta rule, developed by Widrow and Hoff (1960) in the early 1960's, is a supervised training approach in which error correction is done with a least-mean-squares algorithm (LMS) (Anderson et al. 1988). The delta rule is so named because it changes weights in proportion to the difference ("delta") between actual and desired output responses. The delta rule neural network has one layer and can be used to discriminate linearly separable data (one-layer neural networks can form decision regions which are convex). It has been extended to include two or more layers, an extension called backpropagation. By applying neural networks with two or more layers, arbitrarily shaped decision regions can be formed.

In contrast to the delta rule, the backpropagation algorithm. (Rumelhart et al. 1986) is a multilayer neural network algorithm that can. be used to discriminate data that are not linearly separable. But a problem with the backpropagation is that its training process is computationally very complex. Neural network methods, in general, need a lot of training samples to be successful in classification. A lot of training samples together with a computationally complex algorithm can result in a very long learning time.

Rumelhart et al. (1986) added a momentum term to the backpropagation algorithm in order to speed up the training. This has the advantage that it filters out high frequency variations in the weight space. On the other hand, the momentum term causes an upper bound on how large an adjustment can made to a weight. The sign of the momentum term may also cause a weight to be adjusted up the gradient of the error surface instead of down the gradient as desired. Jacobs (1988) introduced a delta-bar-delta learning rule as an attempt to overcome these limitations. The training of the backpropagation method can also be speeded up by using optimization methods other than the gradient descent. Such methods are discussed in the next section.

3.2 "Fast" Neural Networks

Neural network classifiers have been demonstrated to be attractive alternatives to conventional classifiers (Benediktsson et al. 1990b, Gorman et al. 1988). The two major reasons why these classifiers have not gained wider acceptance are (Barnard et al. 1989):

1. They have a reputation for being highly wasteful of computational resources during training.
2. Their training has conventionally been associated with the heuristic choice of a number of parameters; if these parameters are chosen incorrectly. poor performance results, yet no theoretical basis exists for choosing them appropriately for a given problem.

Most neural network methods are based on the minimization of a cost function. The most commonly used optimization approach applied for the minimization is gradient descent (Luenberger 1984). Both the delta rule and the backpropagation algorithm are derived by minimizing the criterion function:

$$E = \sum_{p=1}^{N} \epsilon_p = \frac{1}{2} \sum_{p=1}^{N} \sum_{j=1}^{m} (t_{pj} - o_{pj})^2 \tag{2}$$

where p is a pattern number, N is the sample size, t_{pj} is the desired output of the jth output neuron, o_{pj} is the actual output of the neuron and m is the number of output neurons. Both the delta rule and the backpropagation algorithm are derived from (2) using gradient descent. Both have the two problems listed above, but can be modified to reduce the problems by using different optimization methods.

Watrous (1988) has studied the effectiveness of learning in neural networks and has shown that quasi-Newton methods are far superior to gradient descent for training of neural networks. However, quasi-Newton methods need the approximation of an inverse Hessian matrix which can be computationally intensive in itself. Conjugate-gradient optimization (Barnard et al. 1989, Luenberger 1984) is a method which is only slightly more complicated than gradient descent but does not need any parameter selections like the gain factor of gradient descent. Conjugate-gradient methods have proved to be extremely effective in dealing with general objective functions and are considered among the best general-purpose methods available. Also, in our experience they converge about an order of magnitude faster than gradient descent.

Conjugate-gradient optimization methods differ from gradient descent methods in that search directions in the conjugate-gradient method are not specified beforehand but are determined at each step of the iteration. At each step the current negative gradient vector is computed and added to a linear combination of previous direction vectors to obtain a new conjugate direction vector along which to move (Luenberger 1984). The gradients can be computed using the conventional methods in neural networks (Rumelhart et al. 1986).

The conjugate-gradient method is an "epoch" learning algorithm, i.e., weights are updated in the network only after all patterns have been presented to the network in each cycle. The direction vectors are reinitialized (restarted) every k-th iteration (where k is a fixed humber) since the conjugacy usually deteriorates after several iterations. Line search (Luenberger 1984) is performed to find the minimum of the error curve.

In this paper, conjugate-gradient versions of the delta rule and the backpropagation are applied. The conjugate-gradient neural networks are derived from (2) using conjugate-gradient optimization (Barnard et al. 1989). These methods are called: CGNN-1 (1 layer: output layer) and CGNN-2 (2 layers: hidden and output layers). Both methods are implemented with a sigmoid activation function at the neurons (Rumelhart et al. 1986)

4. EXPERIMENTAL RESULTS

The methods discussed above were applied to classification of multisource and very-high-dimensional data sets and compared to results of statistical methods. Three data sets were used in experiments. Two data sets consisted of multisource remote sensing and geographic data. The third data set was very-high-dimensional simulated High Resolution Imaging Spectrometer (HIRIS) data.

The results of the neural network algorithms were compared to two statistical classifiers: 1) the minimum Euclidean distance classifier (MD) and 2) the maximum likelihood method for Gaussian data (ML).

4.1 Experiments with Colorado Data

The first data set consisted of 4 data sources:

1) Landsat MSS data (4 data channels)
2) Elevation data (in 10 m contour intervals, 1 data channel)
3) Slope data (0-90 degrees in 1 degree increments, 1 data channel)
4) Aspect data (1-180 degrees in 1 degree increments, 1 data channel)

Each channel comprised an image of 135 rows and 131 columns; all channels were co-registered.

The area used for classification was a mountainous area in Colorado, part of a larger region previously analyzed by Hoffer et al. (1975, 1979). The area has 10 ground-cover classes which are listed in Table 1. One class is water; the others are forest *types.* It was very difficult to distinguish among the forest types using the Landsat MSS data alone since the forest classes showed very similar spectral responses. With the help of elevation, slope and aspect data, they could be better distinguished.

Reference data were compiled for the area by comparing a cartographic map to a color composite of the Landsat data and also to a line priinter output of each Landsat channel. By this method 2019 reference points (11.4% of the area) were selected comprising two or more homogeneous fields in the imagery for each class. Two experiments were conducted with this data set. In the initial experiment, the largest field for each class was selected as a training field and the other fields were used for testing the classifiers. Overall 1188 pixels were used for training and 831 pixels for testing the classifiers. This was the same data used in (Benediktsson et al. 1990b) for conventional backpropagation.

4.1.1 Results of the First Experiment on Colorado Data

The results of the classifications are shown in Tables 2.a (training) and 2.b (test), where OA represents overall accuracy (weighted by the number of pixels in each class) and AVE means average (over the classes) accuracy. (The ML method was not applicable, because the data were not truly Gaussian and a few of the covariance matrices were singular.) The results for the MD method are clearly unacceptable since the method gave only 43.27% overall accuracy for training data and 22.26% overall accuracy for test data.

The two neural network approaches, the one-layer CGNN-1 and the two-layer CGNN-2, were trained with Gray-coded input vectors rather than binary input vectors, as discussed in Section 3. Since the data are of relatively low dimensionality, it was necessary to expand the dimensionality and use Gray-coded inputs rather than continuous-valued inputs. Experimental results verified this (results using continuous-valued inputs were about 10% lower in overall accuracy than the results using Gray-coded inputs). Since five of the seven data channels take values in the range from 0 to 255, each data channel was represented by 8 bits and therefore 8 input neurons. The total :number of inputs was 7*8 + 1 = 57 (one extra input is always active and is used to compute the biases (Rumelhart et al. 1986) of the neurons in the succeeding layers). Since the number of information classes was 10, the number of output neurons was selected as 10. The training of the neural networks was considered to have converged if the norm of the gradient of the error at the outputs was less than 0.0001.

The training procedure for the CGNN-1 network did not converge but found a minimum at 319 iterations. The highest overall accuracy (94.87%) and the highest average accuracy (92.49%) for training data were achieved then. However, the best overall accuracy for test data was reached at 100 iterations (55.11%). A major problem with the CGNN-1 and other neural networks is deciding when to stop the training procedure. If a neural network is overtrained it will not necessarily give the best accuracies for test data. The reason is that the network gets too specific to the training data and does not generalize as well. The 319 iterations required to train the CGNN-1 took 547 CPU sec.; the classification of the data took only 10 sec.

The CGNN-2 was implemented in experiments with two or more layers (output and hidden layers). Having more than one hidden layer did not improve the classification performance of this neural network, so only the results with two layers are discussed here. Two-layer networks with 8, 16, 32, 48 and 64 hidden neurons were tried but the performance of the CGNN-2 in terms of classification accuracy was not improved by using more than 32 hidden neurons. Therefore, 32 hidden neurons were used in the experiments reported here.

The CGNN-2 showed the best performance of all the methods in terms of overall and average classification accuracies of both training and. test data. As with the CGNN-1, the training procedure of the CGNN-2 did not converge. At 676 iterations the error function could not be decreased and the training procedure stopped. For test data, the CGNN-2 gave very similar accuracies to the CGNN-1. At 200 iterations the highest overall and average accuracies of test data were reached, 56.32% and 52.59% respectively. In these experiments the CGNN-2 had an overtraining problem similar to the CGNN-1; it gave somewhat less than optimal results for test data classified by the network giving the most accurate results for training data.

The CGNN-2 was much slower in training than the CGNN-1 because of the 32 hidden neurons. Training the CGNN-2 for 676 iterations took 4709 sec. However, the classification of the data took 21 sec which is about twice the time consumed by the CGNN-1.

The results in this experiment illustrate how important it is to select representative training samples when training a neural network. The CGNN-2 network gave more than 97% overall accuracy of training data but only just more than 55% for test data. The training data used here might not be representative since only one training field was selected for each information class. This iimited each information class to a single subclass. The classification results for the training fields indicate that if representative training samples are available, the neural networks can do very well in classification of multisource data. Significantly, arriving at a truly representative set of training samples can be very difficult in practical remote sensing appiications. In order to demonstrate how well the classification methods could do with a more representative sample, a second experiment on the Colorado data was conducted, as discussed below.

4.1.2 Results of the Second Experiment on Colorado Data

To achieve a more representative training sample, uniformly spaced samples were selected from all fields available for each class. The remaining samples were used for testing. By this approach, 1008 samples were obtained for training and 1011 samples for testing (Table 3). By considering the JM distances (Swain 1978) between the different training fields in the MSS data, it was determined that the Landsat MSS source should be trained on 13 data classes. The selection of the data classes was done in the following way. If a field from a specific class was more distant than 0.85 in the sense of JM distance from another field within the same class, the fields were considered to be from two different data classes (using a definition of JM distance with a maximum of 1.00). Using this criterion, class 3 (mountane/subalpine meadow) was split into two data classes, and class 7 (Engelmann spruce) was divided into 3 data classes. All of the other information classes had only one data class. In the methods applied below, the classifiers were trained on the resulting 13 data classes.

The results of this experiment are shown in Tables 4.a (training) and 4.b (test). Since the training data are more representative than in Section 4.1.1, the test results are significantly better (compare to Table 2.b). However, the results in both Tables 2 and 4 show that the MD is not an acceptable choice for classification of this data set.

The neural network methods were trained as in Section 4.1.1. There were 57 inputs; 13 output neurons accounted for the 13 data classes. The input data were Gray-coded and the convergence criterion for the training procedures was the same as in Section 4.1.1.

The training procedure for the CGNN-1 stopped after 344 iteration when the error function did not decrease further. The highest overall accuracy of training data was reached at 344 iterations (82.24%). The highest overall accuracy of test data was reached at 200 iterations (79.62%). The highest average accuracy of test data was also achieved at 200 iterations. At 343 iterations the overall accuracy of test data was 79.43% and the average accuracy was 68.91%.

The two layer CGNN-2 was trained with 8, 16 and 32 hidden neurons. Using more than two layers did not improve the accuracy of the network. The classification results with 8 hidden neurons were the best and are shown in Tables 4.a and 4.b. The training procedure stopped after 933 iterations for which the highest overall accuracy was reached (87.80%) together with the highest average accuracy (79.62%). Using the 8 hidden neurons improved the overall accuracy of training data by over 5% and the average accuracy by over 6% as compared to the CGNN-1. However, the CGNN-2 training procedure was more time-consuming than the CGNN-1, as seen in Table 4.a. Although the training results were better for the CGNN-2 with 8 hidden neurons as compared to the CGNN-1, the test results were worse, both in terms of overall accuracies and average accuracies. The best accuracy for test results with the CGNN-2 were achieved at 150 iterations (overall: 79.23%, average: 65.62%). The results at 933 iterations were lower (overall: 77.65%, average: 65.05%).

The results of this experiment show that the neural network, methods can do much better in classification when representative training samples are used. The highest overall accutacy for test data with the neural network methods was reached with the CGNN-1 (79.62%). Adding hidden neurons did not improve the performance of the networks in terms of classification accuracy for test data, even though it did improve the accuracy for training data. Using hidden neurons also slowed the training procedure. As mentioned before, one of the major problems with the neural network methods is determining how to prevent them from "overtraining." The highest accuracy for test data may be achieved with fewer iterations than the training procedures require.

Up to this point the neural networks have been tested on relatively low-dimensional data with a limited number of samples. It is interesting to see how the networks perform on a data set with more features and more samples. For that purpose an experiment on another multisource data set, the Anderson River data, was conducted.

4.2 Experiments with Anderson River Data

The Anderson River data set is a multisource data set made available by the Canada Centre for Remote Sensing (CCRS) (Goodenough et al. 1987). The imagery involves a 2.8 km by 2.8 km forestry site in the Anderson River area of British Columbia, Canada, characterited by rugged topography, with terrain elevations ranging from 330 to 1100 m above sea level. The forest cover is primarily coniferous, with Douglas fir predominating up to approximately 1050 m elevation, and cedar, hemlock and spruce types predominating at higher elevations. The Anderson River data set consists of six data sources:

1) Airborne Multispectral Scanner (ABMSS) with 11 data channels (10 channels from 380 to 1100 nm and 1 channel from 8 to 14 μm).
2) Steep Mode Synthetic Aperture Radar (SAR) with 4 data channels (X-HH, X-HV, L-HH, L-HV)[2].
3) Shallow Mode SAR with 4 data channels (X-HH, X-HV, L-HH, L-HV).
4) Elevation data, 1 data channel, with elevation in meters = 61.996 + (7.2266 x pixel value).
5) Slope data, 1 data channel, with slope in degrees = pixel value.
6) Aspect data, 1 data channel, with aspect in degrees = 2 x pixel value.

The ABMSS and SAR data were recorded during the week of July 25 to 31, 1978. Each channel comprises an image of 256 lines and 256 columns. All of the images are co-registered with pixel resolution of 12.5m.

There are 19 information classes in the ground reference map provided by CCRS. In the experiments reported here, only the 6 predominant classes were used, as listed in Table 5. Training samples were selected on a uniform grid as 10% of the total sample size of a class. The information classes in the data have been shown to be very hard to separate (Benediktsson et al. 1990c).

2. X- and L-band synthetic aperture radar imagery (horizontal polarization transmit (HH) and horizontal/vertical polarization receive (HV)).

4.2.1 Results of Experiments on Anderson River Data

The results for each of the classification methods are shown in Tables 6.a (training) and 6.b (test). Although the MD method did much better in classification of training and test data than for the Colorado data, it did significantly worse than the multivariate Gaussian ML method. It is questionable for two reasons whether it is appropriate, from a theoretical standpoint, to use a multivariate Gaussian distribution for all of the sources: first, because the topographic sources were not Gaussian; and second, because no information was available for modeling the dependencies between all the data sources. In view of this, the ML method showed surprisingly good performance in terms of training and test accuracy. Three of the data sources [ABMSS, SAR sh, SAR st) can be modeled as Gaussian. Those three sources consist of 19 of the 22 data channels used in the classification. The number of the Gaussian channels is one of the reasons for the relatively good performance of the MI, method.

The CGNN-1 and CGNN-2 were originally trained with Gray-coded input data. Each of the 22 data channels was coded with eight bits and therefore 177 (or 8*22 + 1) input neurons were used for each networks. The data were trained on the six information classes in Table 5. Therefore, six output neurons were selected. The convergence criterion for the training procedures was the same as in the Colorado experiments (gradient of the error function has to be less than 0.0001 for the training procedure to "converge").

After 295 iterations, the training procedure of the CGNN-1 had reached minimum error. The highest overall accuracy of training data was achieved then (OA: 73.50%, Ave: 72.45%). These results were significantly better than the results reached by the statistical methods. The best test result using the CGNN-1 was also achieved at 295 iterations: the CGNN-1 gave overall accuracy of 67.88% and average accuracy of 66.48%.

The CGNN-2 was tested extensively with two layers of neurons since adding more layers did not improve the classification accuracy. In contrast to the CGNN-1, the CGNN-2 gave better results with continuous-valued inputs than Gray-coded inputs (Benediktsson et al. 1990c). The results of the classifications with continuous-valued inputs are reported here. The reason for this good performance with the continuous-valued inputs is the relatively high dimensionality of the data (22 input features). The CGNN-2 was implemented with 23 input neurons and 20 hidden neurons. Adding more 'hidden neurons did not increase the classification accuracy. When the training procedure stopped (the error function did not decrease further) after 1333 iterations, the overall accuracy of training data had reached 75.13% and the average accuracy 74.93%. The CGNN-2 outperformed all the other methods in classification of training data. Also, the CGNN-2 was by far the best method in classification of test data. The highest accuracies of test data were reached after 1300 iterations (OA: 72.77%, Ave: 73.32%). These accuracies are excellent for classification of these data (Benediktsson et al. 1990c). However, after 1300 iterations, the test performance of the CGNN-2 fell off significantly. The test accuracies decreased until the training procedure was stopped. At 1333 iterations, the overall accuracy of test data was only 66.54% and the average accuracy only 65.03%. Obviously the training procedure of the CGNN-2 had the problem of overtraining. However, the CGNN-1 was reasonably fast in training the data. Because of binarization of the inputs, the

CGNN-1 was almost as time consuming as CGNN-2.

4.3 Experiments with Simulated HIRIS Data

This experiment investigated how well the statistical methods and the neural network models perform as classifiers of very-high-dimensional data (data that have many features, possibly hundreds of them). In these experiments, the very-high-dimensional data were simulated High Resolution Imaging Spectrometer (HIRIS) data. The HIRIS instrument is planned to be a part of a cluster of scientific instruments forming the Earth Observing System (EOS). A simulation program called RSSIM (Kerekes et al. 1989) was used to simulate the data.

The simulated data used in the experiments were Gaussian distributed, which is one of the reasons why multivariate statistical approaches were used for the classification. However, a problem with using conventional multivariate statistical approaches for classification of high-dimensional data is that these methods rely on having nonsingular (invertible) class-specific covariance matrices. As mentioned earlier, when n features are used, the training samples for each class need to include at least n+1 *different* samples so that the matrices are nonsingular. Therefore, the covariance matrices may be singular in high-dimensional cases involving limited training samples.

The RSSIM simulation program generated 201 spectral bands of HIRIS data based on statistics from Earth surface reflectance measurements taken at a site in Finney County, Kansas, on May 3, 1977. A total of 1551 observations were combined from three information classes: winter wheat, summer fallow, and an "unknown" class. Each class consisted of 675 samples.

The information classes were assumed to be Gaussian distributed:.

For these experiments, three feature sets (20-, 40- and 60-dimensional) were extracted from the 201 data channels. Each feature set consisted of data channels uniformly spaced over the HIRIS spectral range (0.4 μm to 2.4 μm) excluding the water absorption bands. Also, the 20-dimensional data set was selected as a subset of the 40-dimensional data set and the 40 dimensional data set was selected as a subset of the 60-dimensional data set.

Experiments were conducted using both statistical classification algorithms and the neural network methods (CGNN-1 and CGNN-2). To see how sample size affected the performance of all the algorithms, the experiments were conducted for 100, 200, 300, 400, 500 and 600 training samples per class. In each case, the overall sample size was the same for all of the classes; therefore, the overall accuracy and the average accuracy were equal.

4.3.1 Experimental Results with Simulated HIRIS data.

The data were relatively separable according to the average JM-distance of all feature sets (Benediktsson et al. 1990c). However, classes 2 (summer fallow) and 3 (unknown) were not as distinguishable from each other as both of them were from class 1 (winter wheat).

The results of the experiments with the simulated HIRIS data are shown in Figures 4 (training), 5 (test) and 6 (time of classification plus training). In every case, the statistical ML method was superior to the neural network methods. The ML method, when applicable, was overall the most accurate and fastest, in classification of the 20- and 40-dimensional data sets. The performance of the ML method improved with more features and more training samples. However, it could not be applied to the 60-dimensional data because of a singular covariance matrix. As noted earlier, the singularity problem is a shortcoming of the ML method.

The MD classifier performed poorly. It is very fast but cannot discriminate the classes adequately. Since it does not use any second order statistics, it is likely to perform poorly in classification of high-dimensional data (Lee 1989). Also, it shows saturation, i.e., above a certain number of dimensions its classification accuracy does not increase. In these experiments, the MD classification accuracy did not improve for data sets more complex than the 20-dimensional data.

The CGNN-1 and CGNN-2 were implemented in the experiments with continuous-valued inputs because the results using continuous-valued inputs were found to be about 10% better than with Gray-coded inputs (Benediktsson et al. 1990a, Benediktsson et al. 1990c). Again, the high dimensionality of the data is the reason for the good performance of the continuous-valued input representation.

Of the neural network methods applied, CGNN-2 showed in most cases better performance than the CGNN-1 in terms of overall classification accuracy. The neural network methods performed, in general, slightly better as the number of training samples was increased. Their performance also improved in terms of overall accuracy when more features were used. Although the ML-method was superior to the neural networks in most cases, the results of this experiment show that the neural networks can do almost as well as the ML method when the training sample size is small. For instance, when 200 training samples were used for 40-dimensional data, both neural networks outperformed the Gaussian ML in terms of overall accuracy of test data. The reason for this is that the ML method is undertrained (e.g., 400 training samples per class would be more appropriate for 40 features). However, these results demonstrate the capabilities of the neural networks when a small representative sample size is used. Also, the neural networks clearly outperformed the statistical methods when 60 features were used.

The CGNN-1 uses no hidden neurons, and in the experiments with high-dimensional data it did not do much worse than the CGNN-2. The relatively good performance of the CGNN-1 is consistent with good separability of the data. The CGNN-1 is computationally less intensive than the CGNN-2, so it could be considered a reasonable alternative for classification of very-high-dimensional data.

In defense of the neural network methods, it should be noted that the Gaussian maximum likelihood method had an unfair advantage since the simulated data were generated to be Gaussian. Furthermore, neural networks are relatively easy to implement and do not need any prior information about the data whereas a suitable statistical model has to be available for the ML method. Also, neural network methods were shown earlier to have potential for classifying difficult multitype data sets. However, the neural networks tend not to have as much ability to generalize as the statistical methods, which was evident in the test data results. These methods will not compare favorably with the statistical methods in terms of speed unless implemented on parallel machines. Currently their computation time required for training increases substantially with an increased number of training samples; the statistical methods require very little additional time as the training sample size increases (Figure 6).

5. CONCLUSIONS

The two conjugate-gradient neural network models, CGNN-1 and CGN'N-2, performed well as pattern recognition methods for multisource remotely sensed data. Both neural networks performed well in classification of test data and the two layer CGNN-2 was, as expected, in most cases the better of the two. However, the neural network models have an overtraining problem. If their training procedure goes through too many learning cycles, the neural networks will get too specific in classifying the training data and give less than optimal results for test data. This overtraining problem is a shortcoming that has to be considered in the application of neural networks for classification.

The neural network models have the advantage that they are distribution-free and therefore no prior knowledge is needed about the underlying statistical distributions of the data. This is an obvious advantage over most statistical methods requiring modeling of the data; such modeling is difficult when there is no prior knowledge of the distribution functions or the data are non-Gaussian.

However, the neural networks, especially the CGNN-2, are computationally complex. When the sample size was large in the experiments, the training time could be relatively long. The training of the CGNN-2 is more efficient than conventional backpropagation and requires fewer parameter selections. However, as in the conventional backpropagation, the number of hidden neurons must be selected empirically. Use of too many hidden neurons increases the computational complexity and can degrade the network performance.

The experiments also demonstrated the importance of the representation of the data when a neural network is used. In the experiments, Gray-coded inputs gave better accuracy when the data were relatively low-dimensional but continuous-valued input representation was superior when the data were very-high-dimensional. Input representation is a subject of ongoing research.

Any trainable classifier needs to be trained using representative training samples, but the neural networks are more sensitive to this than are the statistical methods. If the neural networks are trained with representative training samples, the results showed that a one-layer or a two-layer net can do even better than statistical methods in multisource classification of test samples. Although the neural network methods were inferior to the statistical methods in the classification of the very-high-dimensional simulated HIRIS data, the HIRIS data were simulated to be Gaussian and, therefore, the neural network methods did not have much chance of doing better than the statistical methods. The neural network models are more appropriate when the data are of multiple types and cannot be modeled by a convenient multivariate statistical model. However, the results of the experiments with neural network methods showed that when the number of training samples is limited and the Gaussian ML classifier is undertrained, the neural networks can outperform the ML in classification of Gaussian data.

ACKNOWLEDGEMENTS

The Colorado data set was originally acquired, preprocessed, and loaned to us by Dr. Roger Hoffer, who is now at Colorado State University. The Anderson River SAR/MSS data set was acquired, preprocessed, and loaned by the Canada Centre for Remote Sensing, Department of Energy, Mines and Resources, of the Government of Canada. Access to both data sets is gratefully acknowledged.

LIST OF REFERENCES

Anderson, J.A., and Rosenfeld, E. (eds.), 1988, *Neurocomputing*, (Cambridge: MIT Press).

Barnard, E., and Cole, R.A., 1989, A Neural-Net Training Program Based on Conjugate-Gradient Optimization, Technical Report No. CSE 89-014, Oregon Graduate Center, Beaverton, OR 97006.

Benediktsson, J.A., Swain, P.H., Ersoy, O.K., and Hong, D., 1990a, Classification of Very High Dimensional Data Using Neural Networks, Proceedings IGARSS '90, vol. 2, pp. 1269-1272.

Benediktsson, J.A., Swain, P.H., and Ersoy O.K., 1990b, Neural Network .Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data, IEEE Transactions on Gcoscience and Remote Sensing, vol.

GE-28, no. 4, pp. 540-552.

Benediktsson, J.A., and Swain, P.H., 1990c, Statistical Methods and Neural Network Approaches for Classification of Data from Multiple Sources, Technical Report TR-EE 90-64, Laboratory for Applications of Remote Sensing and School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

Decatur, S.E., 1989a, Application of Neural Networks to Terrain Classification, Proceedings of IJCNN '89, vol 1., pp. 283-288.

Decatur, S.E., 1989b, Application of Neural Networks to Terrain Classification, M.S. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Ersoy, O.K., and Hong, D., 1990, Parallel, Self-Organizing, Hierarchical Neural Networks, IEEE Transactions on Neural Networks, vol. 1, no. 2, pp. 167-178.

Goodenough, D.G., Goldberg, M., Plunkett, G., and Zelek, J., 1987, The CCRS SAR/MSS Anderson River Data Set, IEEE Transactions on Geoscience and Remote Sensing, vol. GE-25, no. 3, pp. 360-367.

Gorman, R.P., and Sejnowski, T.J., 1988, Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Signals, Neural Networks, vol. 1, no. 1, pp. 75-90.

Heermann, P.H., and Khazenie, N., 1990, Application of Neural Networks for Classification of Multi-Source Multi-Spectral Remote Sensing Data, Proceedings of IGARSS '90, vol. 2, pp. 1273-1276.

Hoffer, R.M., and staff, 1975, Computer-Aided Analysis of Skylab Multispectral Scanner Data in Mountainous Terrain for Land Use, Forestry, Water Resources and Geological Applications, LARS Information Note 121275, Laboratory for Applications of Remote Sensing, Purdue University, West Lafayette, IN 47907.

Hoffer, R.M., Fleming, M.D., Bartolucci, L.A., Davis, S.M., and Nelson, R.F., 1979, Digital Processing of Landsat MSS and Topographic Data to Improve Capabilities for Computerized Mapping of Forest Cover 'Types, LARS Technical Report 011579, Laboratory for Applications of Remote Sensing in cooperation with Department of Forestry and Natural Resources, Purdue University, West Lafayette, IN 47907.

Jacobs, R.A., 1988, Increased Rates of Convergence Through Learning Rate Adaptation, Neural Networks, vol. 1, no. 4, pp. 295-307.

Kerekes,.J.P., and Landgrebe, D.A., 1989, RSSIM: A Simulation Program for Optical Remote Sensing Systems, Technical Report TR-EE 89-48, School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

Kohonen, T., 1988, An Introduction to Neural Computing, Neural Networks, vol. 1, no. 1, pp. 3-16.

Lathi, B.P., 1983, Modern Digital and Analog Cornmunieation Systems, (New York: Holt, Rinehart and Winston).

Lippman, R.A., 1987, An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, April 1987, pp. 4-27.

Lee, C., 1989, Classification Algorithms for High Dimensional Data, Ph.D. thesis proposal, School of Electrical Engineering, Purdue University, 1989.

Luenberger, D.G., 1984, Linear and Nonlinear *Programming*, 2nd ed., (Reading: Addison-Wesley).

Maslanik, J., Key, J., and Schweiger, A., 1990, Neural Network Identification of Sea-Ice Seasons in Passive Microwave Data, Proceedings of' IGARSS '90, vol. 2, pp. 1281-1284.

McClellan, G.E., DeWitt, R.N., Hemmer, T.H., Matheson, L.H., and Moe, G.O., 1989, Multispectral Image Processing with a Three-Layer Backpropagation Network, Proceedings of *IJCNN* '89, vol. 1, pp. 151-153.

Rosenblatt, F., 1958, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Psychological *Review*, vol. 65, pp. 386-408.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J., 1986, Learning Internal Representation by Error Propagation. In Parallel Distributed Processing: *Explorations* in the Microstructures of Cognition, vol. 1, edited by D.E. Rumelhart and J.L. McClelland, (Cambridge: MIT Press), pp. 318-362.

Swain, P.H., 1978, Fundamentals of Pattern Recognition in Remote Sensing, in Remote Sensing - The Quantitative Approach, edited by P.H. Swain and S.M. Davis, (New York: McGraw-Hill Book Company), pp. 136-1187.

Watrous, R.L., 1988, Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization, Technical Report MS-CIS-88-62, LINC LAB 124, University of Pennsylvania, Philadelphia, PA 19104.

Widrow, B., and Hoff, M.E., 1960, Adaptive Switching Circuits, 1960 IRE WESCON Convention Record, IRE, pp. 96-104.

Wan, E.E., 1990, Neural Network Classification: A Bayesian Interpretation, IEEE Transactions on Neural Networks, vol. 1, no. 4, pp. 303-305.

CAPTIONS TO ILLUSTRATIONS

Figure 1. Schematic Diagram of Neural Network Training Procedure.

Figure 2. Decision Regions for Neural Network with Gray-Coded Inputs.

Figure 3. Decision Regions for Neural Network with Continuous-Valued Inputs.

Figure 4. Classification of Simulated HIRIS Data: Training Samples.

Figure 5. Classification of Simulated HIRIS Data: Test Samples.

Figure 6. Classification of Simulated HIRIS Data: Time of Training and Classification.

Table 1

Training and Test Samples for Information Classes in the First Experiment on the Colorado Data Set

Class #	Information Class	Training Size	Testing Size
1	water	408	195
2	Colorado blue spruce	88	24
3	mountane/subalpine meadow	45	42
4	aspen	75	65
5	Ponderosa pine	105	139
6	Ponderosa pine/Douglas fir	126	188
7	Engelmann spruce	224	70
8	Douglas fir/white fir	32	44
9	Douglas fir/Ponderosa pine/aspen	25	25
10	Douglas fir/white fir/aspen	60	39
Total		1188	831

Table 2

Classification Results for (a) Training Samples and (b) Test Samples in the First Experiment on Colorado Data,.

Table 2.a

Method	Number of Iterations	CPU Time	Percent Agreement with Reference for Class											
			1	2	3	4	5	6	7	8	9	10	OA	AVE
MD		2	47.3	100.0	31.1	28.0	0.0	0.0	67.4	59.4	44.0	28.3	43.27	40.55
CGNN-1	100	186	100.0	98.9	82.2	98.7	69.5	84.9	99.6	90.6	84.0	98.3	94.11	90.67
CGNN-1	319	557	100.0	98.9	82.2	98.7	70.5	85.7	100.0	96.9	92.0	100.0	94.78	92.49
CGNN-2	200	1427	100.0	100.0	93.3	100.0	85.7	92.1	100.0	100.0	100.0	100.0	97.64	97.11
CGNN-2	676	4730	100.0	100.0	95.6	100.0	88.6	96.0	100.0	100.0	100.0	100.0	98.40	98.02
# of pixels			408	88	45	75	105	126	224	32	25	60	1188	1188

Table 2.b

Method	Number of Iterations	Percent Agreement with Reference for Class											
		1	2	3	4	5	6	7	8	9	10	OA	AVE
MD		38.9	100.0	0.0	16.9	0.0	6.9	75.7	4.5	4.0	12.8	22.26	25.99
CGNN-1	100	96.4	83.3	40.5	41.5	11.5	43.6	100.0	2.3	12.0	87.2	55.11	51.83
CGNN-I	319	94.9	83.3	33.3	38.5	11.5	44.1	100.0	2.3	16.0	79.5	54.03	50.34
CGNN-2	200	99.0	83.3	45.2	38.5	17.3	40.4	100.0	2.3	12.0	94.9	56.32	53.29
CGNN-2	676	97.9	79.2	38.1	41.5	15.1	42.0	100.0	2.3	16.0	82.1	55.35	51.42
# of pixels		195	24	42	65	139	188	70	44	25	39	831	831

Table 3

Training and Test Samples for Information Classes in the Second Experiment on the Colorado Data Set

Class #	Information Class	Training Size	Testing Size
1	water	301	302
2	Colorado blue spruce	56	56
3	mountane/subalpine meadow	43	44
4	aspen	70	70
5	Ponderosa pine	157	157
6	Ponderosa pine/Douglas fir	122	122
7	Engelmann spruce	147	147
8	Douglas fir/white fir	38	38
9	Douglas fir/Ponderosa pine/aspen	25	25
10	Douglas fir/white fir/aspen	49	50
Total		1008	1011

Table 4

Classification Results for (a) Training Samples and (b) Test Samples in the Second Experiment on Colorado Data.

Table 4.b

Method	Number of Iterations	CPU Time	Percent Agreement with Reference for Class											
			1	2	3	4	5	6	7	8	9	10	OA	AVE
MD		2	41.5	98.2	25.6	37.1	37.6	0.0	73.5	0.0	40.0	24.5	40.28	37.80
CGNN-1	200	375	100.0	85.7	53.5	84.3	58.6	74.6	100.0	23.7	56.0	91.8	82.24	72.82
CGNN-1	343	644	100.0	85.7	55.8	82.9	61.1	69.6	100.0	26.3	56.0	93.9	82.24	73.13
CGNN-2	150	292	100.0	91.1	46.5	87.1	66.9	77.0	100.0	34.2	8.0	91.8	83.23	70.26
CGNN-2	933	1719	100.0	96.4	48.8	95.7	64.3	91.8	100.0	60.5	40.0	98.0	87.70	79.55
# of pixels			301	56	43	70	157	122	147	38	25	49	1008	1008

Table 4.b

Method	Number of Iterations	Percent Agreement with Reference for Class											
		1	2	3	4	5	6	7	8	9	10	OA	AVE
MD		40.1	100.0	34.1	30.0	32.5	0.8	69.4	0.0	28.0	20.0	37.98	35.49
CGNN-1	200	100.0	85.7	45.5	75.7	54.8	74.6	98.0	18.4	60.0	78.0	79.62	69.07
CGNN-1	343	100.0	85.7	47.7	72.9	55.4	73.0	98.0	18.4	60.0	78.0	79.43	68.91
CCNN-2	150	100.0	96.4	27.3	82.9	61.1	65.6	98.6	26.3	20.0	78.0	79.23	65.62
CGNN-2	933	100.0	87.5	38.6	77.1	56.1	63.1	96.6	39.5	20.0	72.0	77.65	65.05
# of pixels		302	56	44	70	157	122	147	38	25	50	1011	1011

Table 5

Information Classes, Training and Test Samples Selected from the Anderson River Data Set.

Class #	Size	Information Class	Training	Testing
1	9715	Douglas Fir (31-40m)	971	8744
2	5511	Douglas Fir (21-30m)	551	4960
3	5480	Douglas Fir + Other Species (31-40m)	548	
4	5423	Douglas Fir + Lodgepole Pine (21-30m)	542	
5	3173	Hemlock + Cedar (31-40m)	317	
6	12600	Forest Clearings	1260	11340
Total	41902		4189	37713

Table 6

Classification Results for the Anderson River
Data Set: (a) Training Samples, (b) Test Samples.

Table 6.a

Method	Number of iterations	CPU time	Percent Agreement with Reference for Class							
			1	2	3	4	5	6	OA	AVE
MD		68	40.4	8.9	47.6	87.7	42.3	72.4	50.51	46.55
ML		1095	54.6	31.8	87.8	90.9	81.4	73.3	68.23	69.92
CGNN-1	295	5129	69.4	45.2	72.3	74.5	87.7	85.8	73.50	72.45
CGNN-2	1300	10601	78.3	51.9	77.0	71.8	85.8	80.2	74.17	74.93
CGNN-2	1333	10896	78.4	52.3	77.4	72.0	86.4	80.2	75.13	74.43
# of pixels			971	551	548	542	317	1260	4189	4189

Table 6.b

Method	Number of iterations	Percent Agreement with Reference for Class							
		1	2	3	4	5	8	OA	AVE
MD		39.7	8.9	48.4	70.2	46.0	71.7	50.83	47.48
ML		50.8	27.7	84.5	81.9	73.8	72.0	64.30	65.12
CGNN-1	,295	63.5	38.2	68.7	68.1	79.8	80.7	67.88	66.48
CGNN-2	1300	70.9	49.1	76.9	71.4	85.5	80.1	72.77	72.32
CGNN-2	1333,	65.7	28.2	65.8	69.9	81.6	79.1	66.54	65.03
# of pixels		87.44	4960	4932	4881	2856	11340	37713	37713

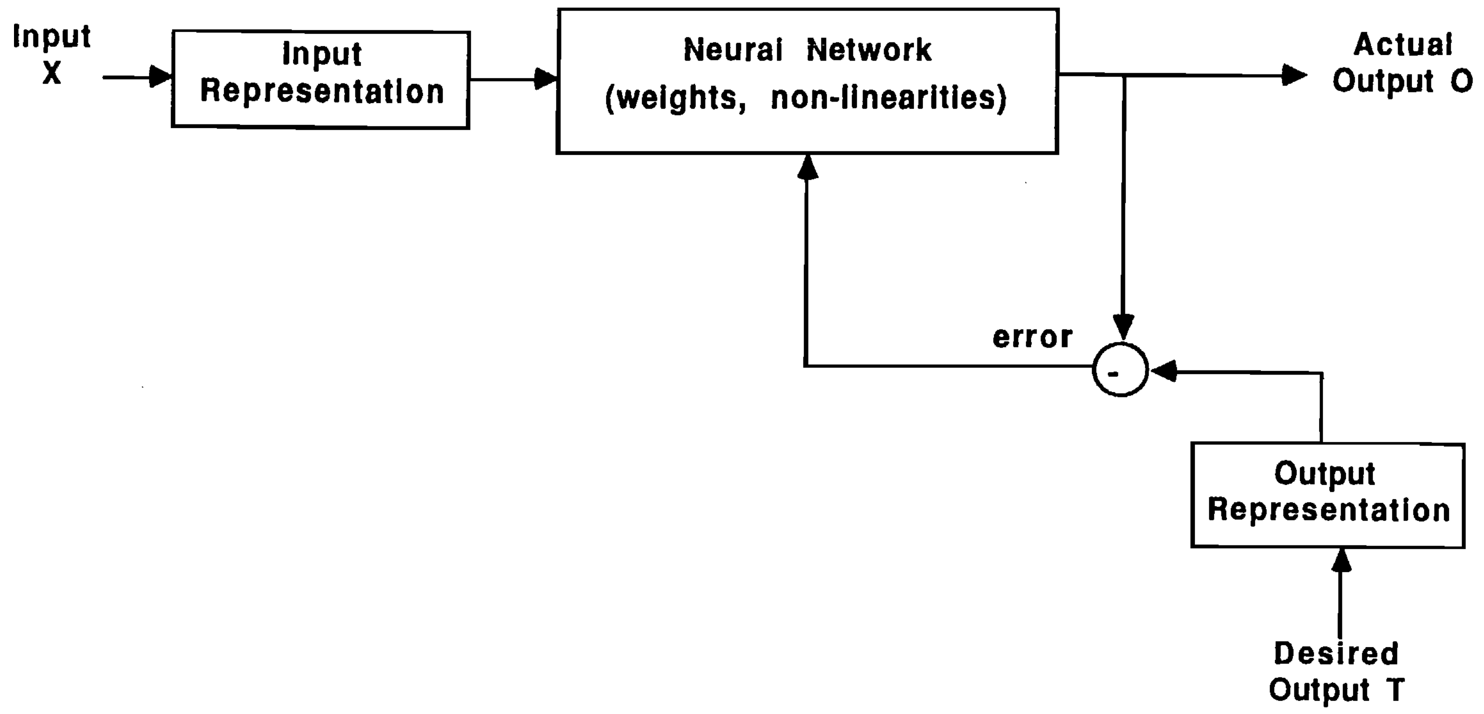

Figure 1. Schematic Diagram of Neural Network Training Procedure

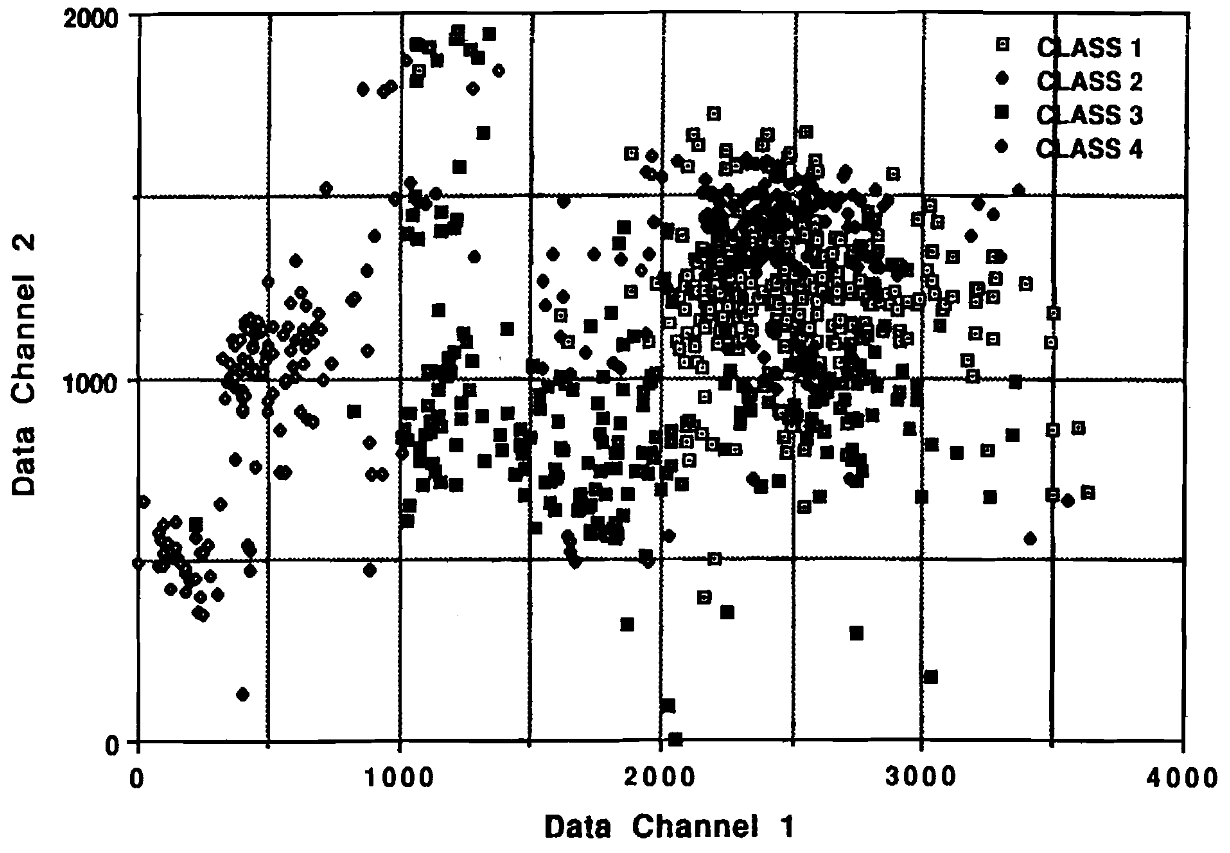

Figure 2. Decisions Regions for Neural Network with Gray-Coded Inputs.

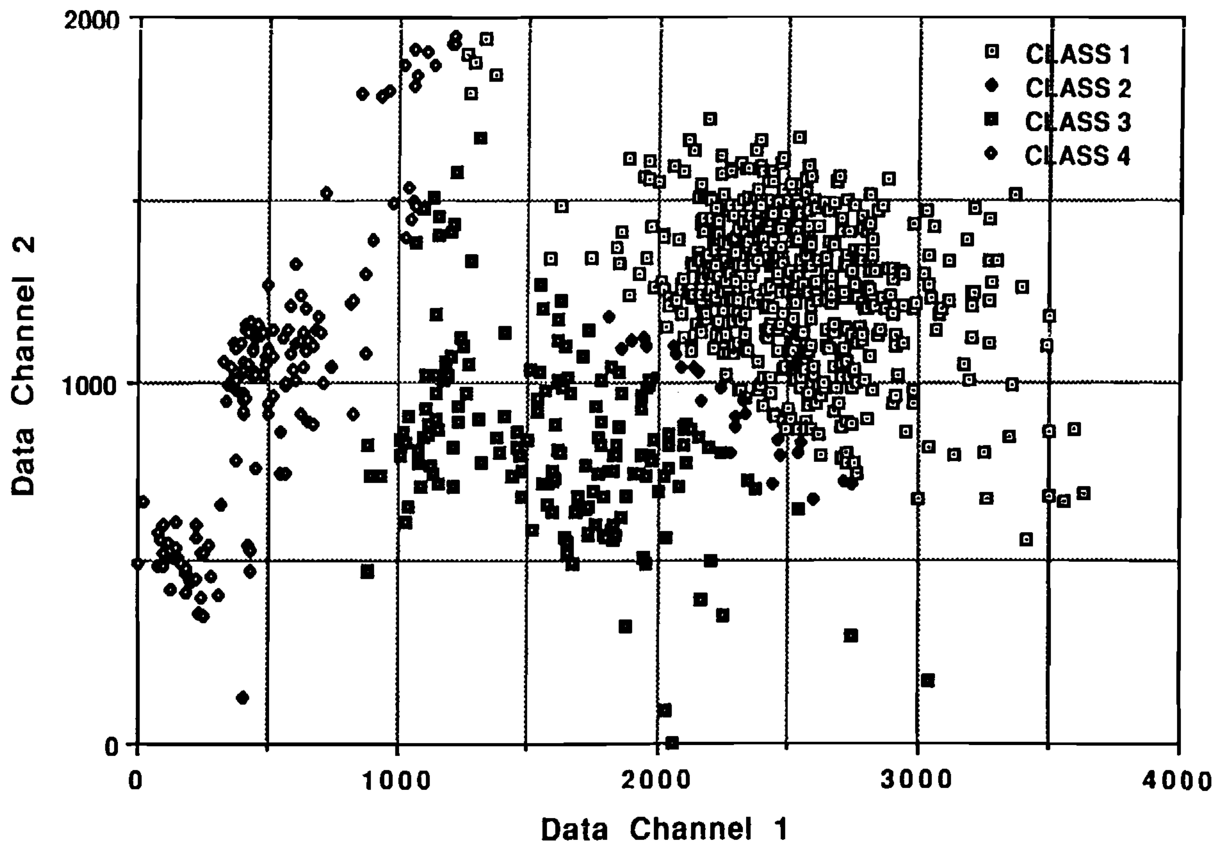

Figure 3. Decisions Regions for Neural N⊟ iwork with Continuous-Valued Inputs.

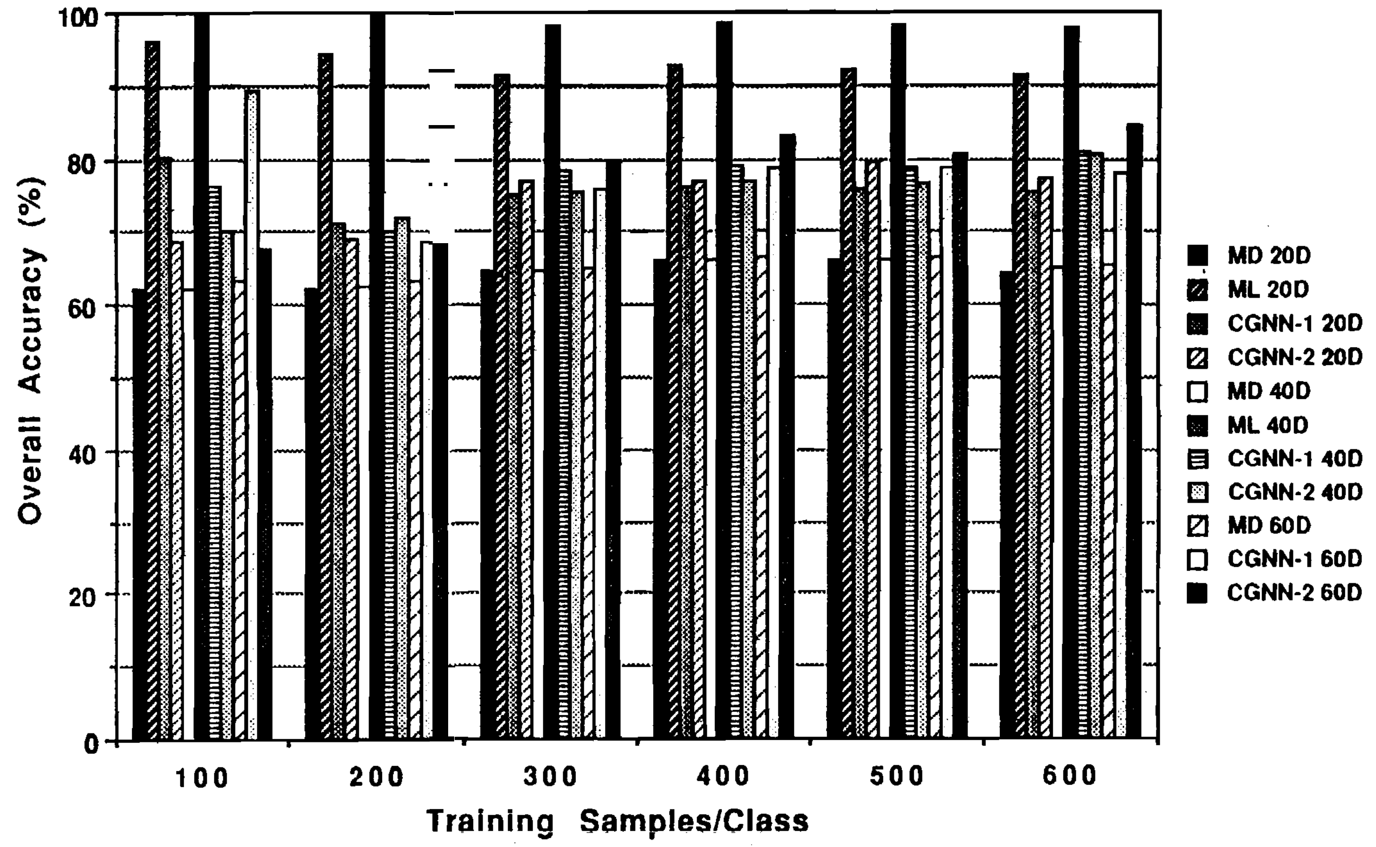

Figure 4. Classification of Simulated HIRIS Data: Training Samples.

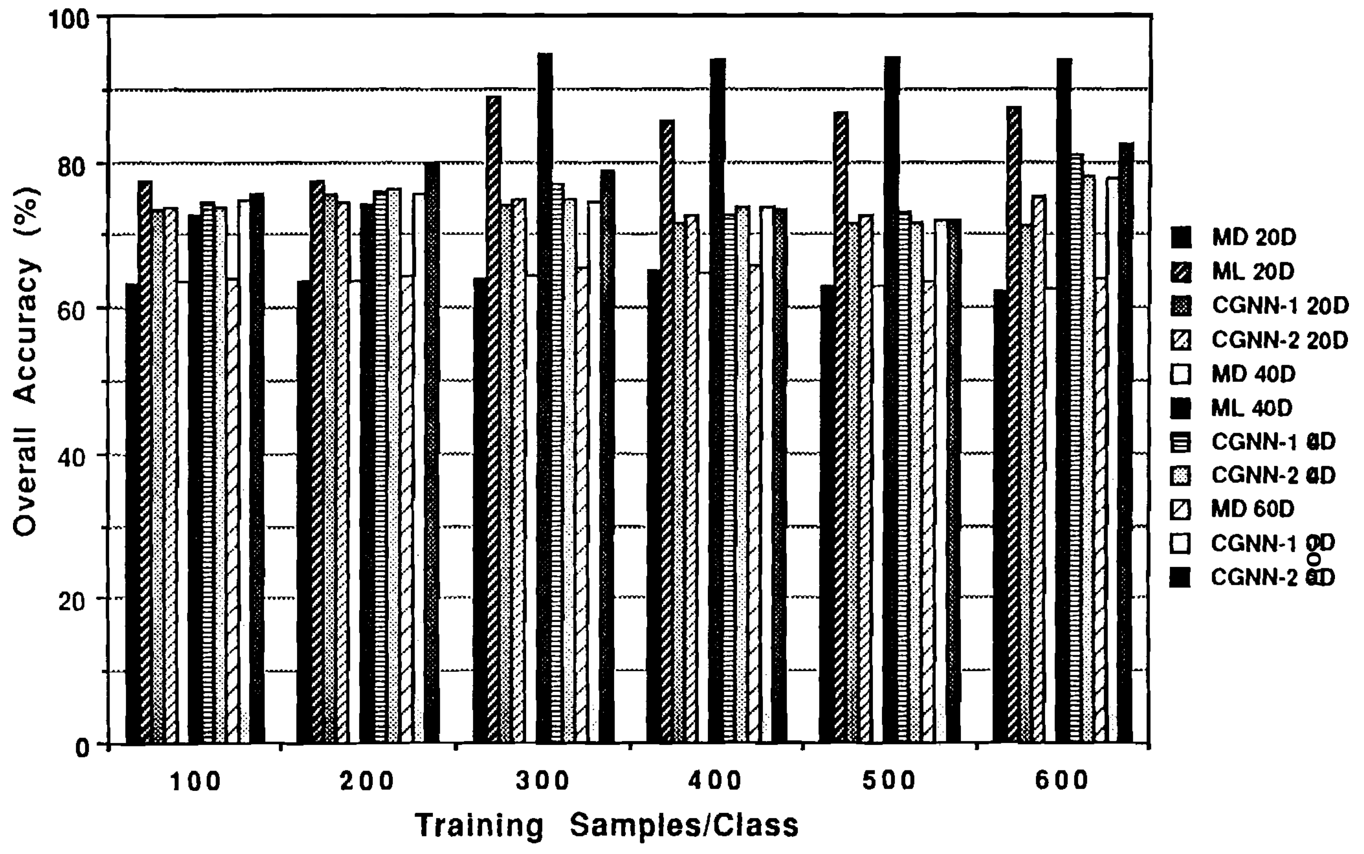

Figure 5. Classification of Simulated HIRIS Data: Test Samples.

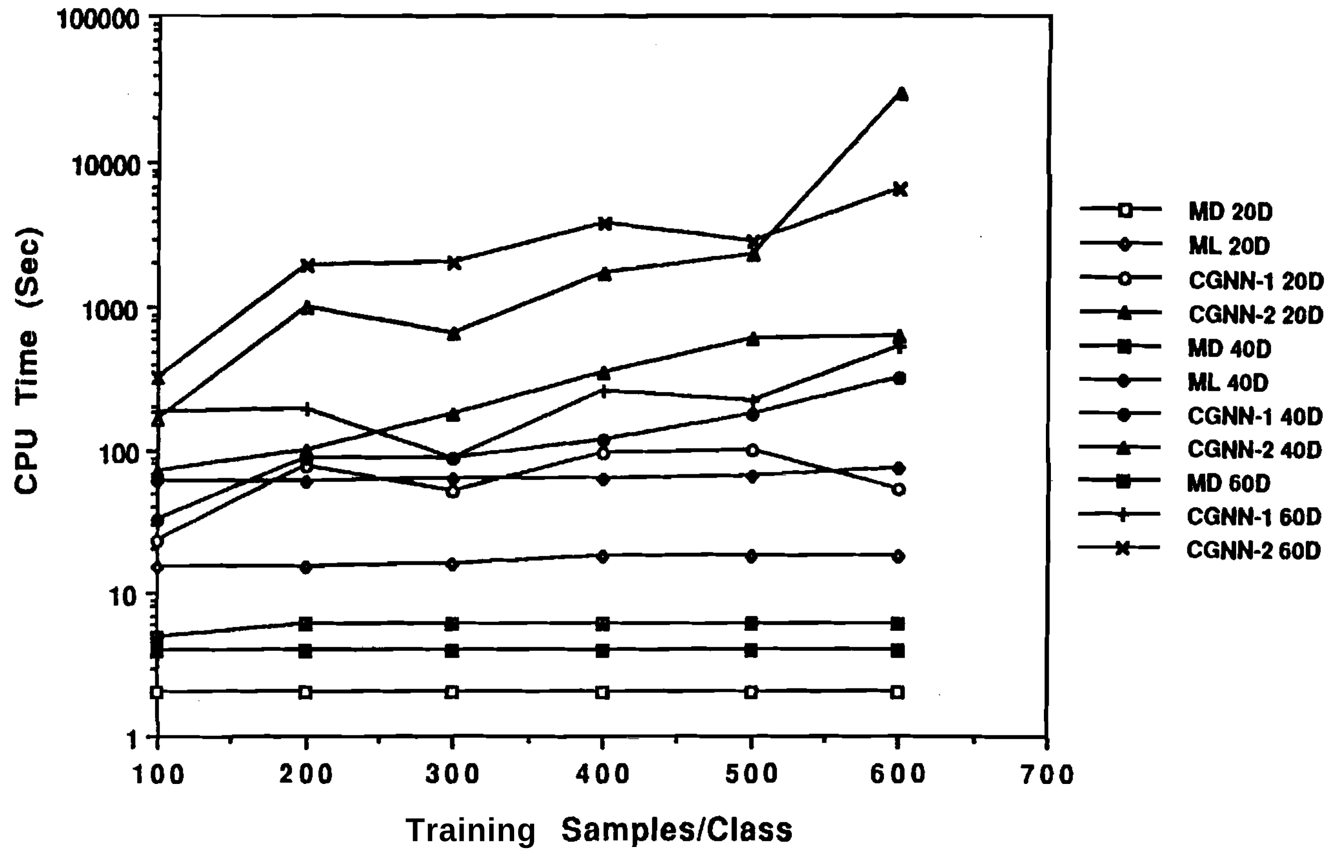

Figure 6. Classification of Simulated HIRIS Data: Time of Training and Classification.