Optimized Combination, Regularization, and Pruning in Parallel Consensual Neural Networks

J. A. Benediktsson[a], J. Larsen[b],

J.R. Sveinsson[a], and L.K. Hansen[b]

[a]Department of Electrical and Computer Engineering, University of Iceland,

Hjardarhaga 2-6, 107 Reykjavik, Iceland

[b]Section for Digital Signal Processing, Department of Mathematical Modelling,

Technical University of Denmark, Building 321, DK-2800 Lyngby, Denmark

ABSTRACT

Optimized combination, regularization, and pruning is proposed for the Parallel Consensual Neural Networks (PC-NNs) which is a neural network architecture based on the consensus of a collection of stage neural networks trained on the same input data with different representations. Here, a regularization scheme is presented for the PCNN and in training a regularized cost function is minimized. The use of this regularization scheme in conjunction with Optimal Brain Damage pruning is suggested both to optimize the architecture of the individual stage networks and to avoid overfitting. Experiments are conducted on a multisource remote sensing and geographic data set consisting of six data source. The results obtained by the proposed version of PCNN are compared to other classification approaches such as the original PCNN, single stage neural networks and statistical classifiers. In comparison to the originally proposed PCNNs, the use of pruning and regularization not only produces simpler PCNNs but also gives higher classification accuracies. In particular, using the proposed approach, a neural network based non-linear combination scheme, for the individual stages in the PCNN, produces excellent overall classification accuracies for both training and test data.

Keywords: Consensus Theory, Neural Networks, Data Fusion, Regularization, Remote Sensing

1. INTRODUCTION

In multisource classification, different types of information from several data sources are used for classification in order to improve the classification accuracy as compared to the accuracy achieved by single-source classification. Conventional statistical pattern recognition methods are not appropriate in classification of multisource data since such data cannot, in most cases, cannot be modeled by a common convenient multivariate statistical model. However, methods based on consensus theory have shown potential in classification of multisource data.[1]

The parallel consensual neural network (PCNN)[2] is somewhat based on statistical consensus theory.[1] In the PCNN the input data are transformed several times and input data with different (transformed) representations are fed into different neural networks (called stage neural networks). The output of the PCNN is based on the consensus among neural networks trained on the same original data with different representations.[2]

In this paper, optimized combination, regularization, and pruning is proposed for the PCNN. The need for regularization is two-fold for the PCNN. First, it remedies numerical problems in the training process by smoothing the cost function and by introducing curvature in low (possibly zero) curvature regions of the cost function. Secondly, regularization is a tool for reducing variance by introducing an extra bias.[3] In the paper, we use a regularization scheme for the PCNN in conjunction with Optimal Brain Damage (OBD) pruning to optimize the architecture of the individual stage networks and to avoid overfitting. The OBD pruning eliminates an inherent redundancy in the widely used SoftMax classification network.[4] Also, in the PCNN, the outputs from the individual stage neural networks need to be combined in some optimized way. Here we use neural networks for such combination.

Other author information: (Send correspondence to Jon Atli Benediktsson)
Jon Atli Benediktsson: E-mail: benedikt@hi.is

The paper begins with a short overview of consensus theory which is followed by a discussion on neural networks with parallel stages and the PCNN. Then the individual stage neural networks based on regularization and pruning are discussed. Finally, experimental results and conclusions are given.

2. CONSENSUS THEORY

Consensus theory[3–8] involves procedures with the goal of combining single probability distributions to summarize estimates from multiple experts (data sources) with the assumption that the experts make decisions based on Bayesian decision theory. Consensus theory is closely related to the method of stacked generalization[9] where outputs of experts are combined in a weighted sum with weights which are based on the individual performance of the experts. In most consensus theoretic methods each data source is at first considered separately. For a given source an appropriate training procedure can be used to model the data by a number of source-specific densities that will characterize that source[1]. The data types are assumed to be very general. The source-specific classes or clusters are therefore referred to as data classes, since they are defined from relationships in a particular data space. In general there may not be a simple one-to-one relation between the user-desired information classes and the set of data classes available since the information classes are not necessarily a property of the data. In consensus theory, the information from the data sources is aggregated by a global membership function, and the data are classified according to the usual maximum selection rule into the information classes. The combination formula obtained is called a consensus rule.

Several consensus rules have been proposed. Probably the most commonly used consensus rule is the linear opinion pool (LOP) which has the following (group probability) form for the information class ω_j if n data sources are used:

$$C_j(X) \; = \; \sum_{i=1}^{n} \lambda_i p(\omega_j|x_i) \tag{1}$$

where $X = [x_1, \ldots, x_n]$ is an input data vector, $p(\omega_j|x_i)$ is a source-specific posterior probability and λ_i's ($i = 1, \ldots, n$) are source-specific weights which control the relative influence of the data sources. The weights are associated with the sources in the global membership function to express quantitatively the goodness of each source.[5]

The LOP has a number of appealing properties. For example, it is simple, yields a probability distribution, and the weight λ_i reflects in some way the relative expertise of the i-th expert. Also, if the data sources have absolutely continuous probability distributions, the LOP gives an absolutely continuous distribution. In using the LOP, it is assumed that all of the experts observe the input vector X. Therefore, (1) is simply a weighted average of the probability distributions from all the experts and the result is a combined probability distribution.

The LOP, though simple, has several weaknesses;[6] e.g., it shows dictatorship when Bayes' theorem is applied, i.e., only one data source will dominate in making a decision. It is also not externally Bayesian (does not obey Bayes' rule). The reason it is not externally Bayesian is that the linear opinion pool is not derived from the joint probabilities using Bayes' rule. Another consensus rule, the logarithmic opinion pool (LOGP), has been proposed to overcome some of the problems with the LOP. The LOGP can be described by

$$L_j(X) \; = \; \prod_{i=1}^{n} p(\omega_j|x_i)^{\lambda_i} \tag{2}$$

where $\lambda_1, \ldots, \lambda_n$ are weights which should reflect the goodness of the data sources. Often it is assumed that $\sum_{i=1}^{n} \lambda_i = 1$.

The LOGP treats the data sources independently. Zeros in the LOGP are vetos; i.e., if any expert assigns $p_i(\omega_j|x_i) = 0$, then $L_j(X) = 0$. This dramatic behavior is a drawback if the density functions are not carefully estimated. The LOGP is externally Bayesian, but it is computationally more complicated than the LOP.

In the Parallel Consensual Neural Network (PCNN), consensus theoretic approaches and neural networks are combined.[2] In the PCNN consensus theory is implemented with neural networks based on the fact that feedforward neural networks which minimize mean-squared error at the output have been shown to approximate posterior probabilities,[10] $p(\omega_j|x_i)$ when one output neuron is assigned to each class, ω_j.

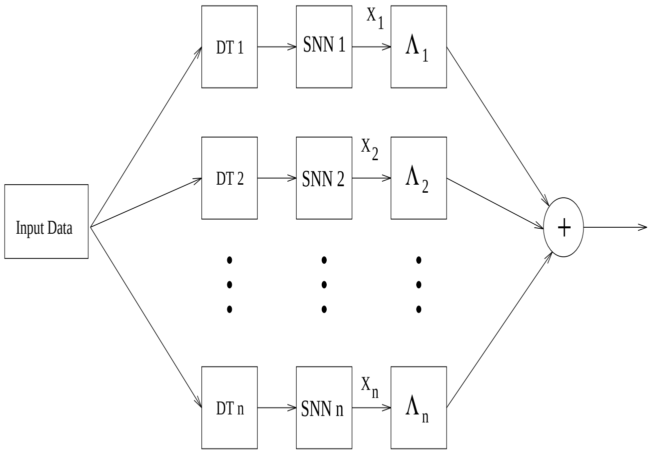

Figure 1. The PCNN with weighted individual stages.[1]

3. NEURAL NETWORKS WITH PARALLEL STAGES

Several methods have been proposed to combine multiple neural networks. Averaging separate networks has been shown to improve generalization performance for the mean-squared error.[13,14] Tumer and Gosh[15] have also shown that substantial improvements can be achieved in difficult pattern recognition problems by combining or integrating the outputs of multiple classifiers. However, the earliest attempt at combining multiple networks can be credited to Nilsson[16] who proposed his committee machines based on a collection of single layer networks as an attempt to design a multilayer neural network which could classify complicated data. Hansen and Salamon[17] discuss the application of an ensemble of multilayer neural networks. Their ensemble consists of several SNNs where each SNN receives the same input data, similar to Nilsson's committee machines. Each SNN is based on the backpropagation network, and the weights in different SNNs are initialized differently in order to avoid the same local minima for all the networks. The ensemble network makes the final decision (classification) based on the majority vote from all the networks. Alpaydin[18] proposed a similar architecture to the one proposed by Hansen and Salamon[17], offering the possibility of using different types of stage networks. Battiti and Colla[19] have looked at several different ways of combining outputs of a set of neural network classifiers, and Rogova[20] has combined the outputs of neural networks by Dempster-Shafer methods. Cho and Kim[21] have combined the outputs of multiple networks based on fuzzy logic.

3.1. The Parallel Consensual Neural Network

Implementing consensus theory in neural networks involves using a collection of neural networks. In the PCNN[2] this is achieved by using neural networks with several parallel stages as depicted in Figure 1. Each stage can be a particular neural network, here referred to as a stage neural network (SNN). Unlike a multilayer network, each SNN is essentially independent of the other SNNs in the sense that each SNN does not receive its input directly from the previous SNN. In the PCNN, the input data to the SNNs are obtained by applying a data transform (DT) to the original input vectors. Therefore, the stages are trained on different representations of the same input data. Each stage neural network (SNN) has the same number of output neurons (equal to the number of data classes) and is trained for a fixed number of iterations or until the training procedure converges. When the training of all the stages has finished, the consensus for the SNNs is computed. The consensus is obtained by taking class-specific weighted averages of the output responses of the SNNs. Thus, the PCNN attempts to improve its classification accuracy by weighted averaging of the SNN responses from several different input representations. By doing this, the PCNN attempts to give highest weighting to the SNN trained on the "best" representation of input data.

Using the proposed PCNN architecture, it can be guaranteed that the PCNN should do no worse than single stage networks, at least in terms of training accuracy in the mean square sense. This is based on an argument that a group decision is better in terms of mean square error than a decision based on the opinion of a single expert.[2] To be able to guarantee such performance in classification of test data, cross-validation methods can be used. Also, it has been shown[17] that if each of the networks in a collection of neural networks arrive at the correct classification with probability $1-p$ and the networks make independent errors, the probability of a sum of network outputs being in error is monotonically decreasing in n if $p < 1/2$. This implies that using a collection of networks reduces the expected classification error if the networks make independent errors. However, the independence assumption is hard to justify in most cases.

It has also been shown[22] that the standard deviation of the classification of a collection of neural networks (such as the PCNN) decreases as the number of stage networks increase.

3.2. Weight Selection Schemes for Consensus Theory

The weight selection schemes in consensus theory should reflect the goodness of the separate input data sources, i.e., relatively high weights should be given to data sources that contribute to high accuracy. There are at least two potential weight selection schemes.[1,2] The first scheme is to select the weights such that they weight the individual data sources but not the classes within the sources. For this purpose, reliability measures which rank the data sources according to their goodness can be used as a bases for *heuristic weight selection*. These reliability measures might be, for example, source-specific overall classification accuracy of training data, overall separability or equivocation.[5]

The second scheme is to choose the weights such that they not only weight the individual stages but also the classes within the stages. This scheme consists of defining a function

$$Y = f(Z, \Lambda)$$

where Z contains source-specific posteriori discriminative information and Λ corresponds to the source-specific weights in (1) and (2).

In the case when f is non-linear, a neural network can be used to obtain a mean square estimate of the function and the consensus theoretic classifiers with equal weights can be considered to preprocess the data for the neural networks. Then, a neural network learns the mapping from the source-specific posteriori probabilities to the information classes. Thus, the neural network is used to optimize the classification capability of the consensus theoretic classifiers.[1,23]

3.3. Data Transforms

The major source of classification error in single stage neural networks is the nonseparability of the classes. To reduce or eliminate classification errors it is desirable to find a transformation which maps the input vectors into another set of vectors that can be classified more accurately. A variety of schemes can be used in the PCNN to transform the data.[2]

A general approach proposed for the transformation of analog input data is based on the wavelet packet transform (WPT). The wavelet packet transform[2,24] provides a transformation of a signal from the time domain to the frequency domain and is a generalized version of the wavelet transform.[25] The WPT is computed on several *levels* with different time/frequency resolutions.

The full WPT for a time domain signal can be calculated by successive application of lowpass and highpass decimation operations. Let $h(k)$ and $g(k)$, $k = 1, 2, \ldots, L$, be the finite lowpass and highpass impulse responses for the WPT.[24] Let $x(m)$, $m = 1, 2, \ldots, N$, denote the original time domain signal of finite length N, where $N = 2^{n-1}$. Define G and H as the operators which perform the convolution of $x(m)$ with $h(k)$ and $g(k)$, respectively, followed by a decimation by two. Then we have

$$\{Hx\}(m) = \sum_{k=1}^{L} x(k)h(2m-k)$$

$$\{Gx\}(m) = \sum_{k=1}^{L} x(k)g(2m-k),$$

for $m = 1, 2, \ldots, N$. Due to decimation, Hx and Gx each contain half as many samples as x. The operators H and G form a pair of quadrature mirror filters (QMFs) and satisfy the following orthogonality conditions:

$$HH^* = GG^*, \qquad HG^* = GH^* \qquad \text{and} \qquad H^*H + G^*G = I,$$

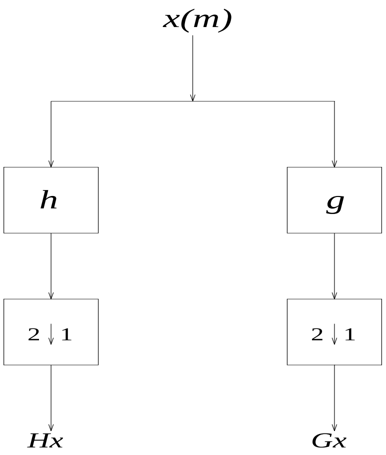

Figure 2. Lowpass-Decimation and Highpass-Decimation of a Time Domain Signal.

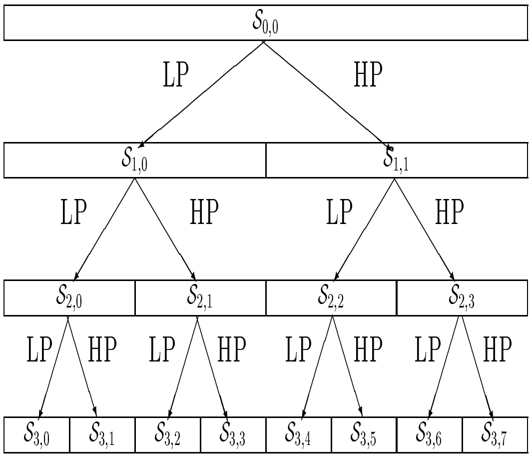

Figure 3. Levels and Bins for the Full WPT.

where I is the identity operator, H^* and G^* are adjoint operations of H and G, respectively. Various design criteria such as regularity, symmetry, etc., on the lowpass filter coefficients $h(k)$ are suggested by Daubechies.[25] Once the L-tap lowpass FIR-filter $h(k)$ is fixed, the L-tap highpass filter can be found by $g(k) = (-1)^k h(L-1-k)$.

The WPT may be calculated using a recursion of the above mentioned filter decimation operations. The top level, called level 0, of the full WPT contains the original time domain signal and thus has one bin. Level 1 of the WPT has two bins where the first bin contains Hx and the second bin contains Gx. The level 1 representation has two degrees of frequency resolution, i.e., the low and the high frequency portions of the original signal have been separated into two bins, but due to decimation, each bin has only half the time resolution that exists at level 0. Level 2 of the WPT contains four bins, where each bin contains sequences generated by the operations H^2x, GHx, HGx and G^2x. Hence, level 2 has four degrees of frequency resolutions, but each bin has only half the time resolution that existed at level 1. This process can be repeated $n-1$ times where $N = 2^{n-1}$. By proceeding down through the levels of the WPT, the tradeoff between time resolution and frequency resolution is observed. The sequence, $\mathcal{S}_{l,b}x$, at bin b and level l of the WPT can be written as

$$\mathcal{S}_{l,b}x = H^*\mathcal{S}_{l+1,2b}x + G^*\mathcal{S}_{l+1,2b+1}x \qquad \text{for} \quad l = 0, 1, \ldots, n-1, \quad b = 0, 1, \ldots, 2^{n-1}-1,$$

and the lowpass and highpass filter decimations of the sequence $\mathcal{S}_{l,b}x$ are

$$\mathcal{S}_{l+1,2b}x = H\mathcal{S}_{l,b}x$$

and

$$\mathcal{S}_{l+1,2b+1}x = G\mathcal{S}_{l,b}x.$$

The WPT provides a systematic way for transforming the input data for the PCNN. Each level of the full WPT consists of data for the different stage networks. Therefore, the stages will have the same original input data with different time-frequency resolutions. Thus, the PCNN attempts to find the consensus for these different representations of the input data, and the optimal weighting method will consequently give the best representation the highest weighting.[2]

An advantage of the WPT is that it is not computationally intensive, i.e., the computations are $O(N \log_2 N)$. This property is very important for the PCNN, especially if the number of stages (n) is large.

4. REGULARIZATION AND PRUNING IN NEURAL NETWORK CLASSIFICATION

Suppose that the input data vector is denoted by x with $\dim(x) = n_I$. The aim is to model the posterior probabilities $p(\omega_i|x)$, $i = 1, 2, \cdots, c$. Then the Bayes optimal* classifier assigns class label ω_i to x if $i = \arg\max_j p(\omega_j|x)$.

*That is, minimal probability of misclassification.

The outputs,[10,26] $\widehat{y}_i$, of the neural network represent *estimates* of the posterior probabilities, i.e., $\widehat{y}_i = \widehat{p}(\omega_i|x)$; hence, $\sum_{i=1}^{c} p(\omega_i|x) = 1$. Therefore, it is only needed to estimate $c-1$ posterior probabilities, e.g., $p(\omega_i|x)$, $i = 1, 2, \cdots, c-1$, since the remaining posterior is calculated as $p(\omega_c|x) = 1 - \sum_{i=1}^{c-1} p(\omega_i|x)$.

Define a two-layer feed-forward network with n_I inputs, n_H hidden neurons and $c-1$ outputs by:

$$b_j(x) \;=\; tanh(\sum_{\ell=1}^{n_I} w_{j\ell}^I x_\ell + w_{j0}^I) \tag{3}$$

$$\phi_i(x) \;=\; \sum_{j=1}^{n_H} w_{ij}^O b_j(x) + w_{i0}^O \tag{4}$$

where $w_{j\ell}^I$, w_{ij}^O are the input-to-hidden and hidden-to-output weights, respectively. All weights are assembled in the weight vector $w = \{w_{j\ell}^I, w_{ij}^O\}$.

In order to interpret the network outputs as probabilities a *modified* normalized exponential transformation similar to SoftMax is used,

$$\widehat{y}_i = \frac{\exp(\phi_i)}{\sum_{j=1}^{c-1} \exp(\phi_j) + 1}, \quad i = 1, 2, \cdots, c-1, \qquad \widehat{y}_c = 1 - \sum_{i=1}^{c-1} \widehat{y}_i. \tag{5}$$

4.1. Training and Regularization

Assume that a training set $\mathcal{T}$ is available which consists of N_{train} related input-output pairs $\mathcal{T} = \{(x(k), y(k))\}_{k=1}^{N_{\text{train}}}$ where

$$y_i(k) = \begin{cases} 1 & \text{if } x(k) \in \omega_i \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The likelihood of the network parameters is given by,[26]

$$p(\mathcal{T}|w) = \prod_{k=1}^{N_{\text{train}}} p(y(k)|x(k), w) = \prod_{k=1}^{N_{\text{train}}} \prod_{i=1}^{c} (\widehat{y}_i(k))^{y_i(k)} \tag{7}$$

where $\hat{y}(k) = \hat{y}(x(k), w)$ is a function of the input and weight vectors. The training error is the normalized negative log-likelihood

$$S_{\mathcal{T}}(w) = -\frac{1}{N_{\text{train}}} \log p(\mathcal{T}|w) \equiv \frac{1}{N_{\text{train}}} \sum_{k=1}^{N_{\text{train}}} \ell\left(y(k), \hat{y}(k); w\right) \tag{8}$$

with $\ell(\cdot)$ denoting the loss given by

$$\ell\left(y(k), \hat{y}(k); w\right) = \log(1 + \sum_{j=1}^{c-1} \exp(\phi_j(x(k)))) - \sum_{i=1}^{c-1} y_i(k)\phi_i(x(k)). \tag{9}$$

The objective of training is to minimize the regularized cost function† which is defined as the negative log-likelihood augmented by separate weight decay regularization for input-to-hidden and hidden-to output layers, as shown by,

$$C(w) = S_{\mathcal{T}}(w) + \kappa_I \cdot |w^I|^2 + \kappa_O \cdot |w^O|^2 \tag{10}$$

where $w = [w^I, w^O]$ with w^I, w^O denoting the input-to-hidden weights and hidden-to-output weights, respectively,and $\kappa_I \equiv \alpha_I/(2N_{\text{train}})$ and $\kappa_O \equiv \alpha_O/(2N_{\text{train}})$ are the weight decay parameters.

Training provides the estimated weight vector $\widehat{w} = \arg\min_w C(w)$ and is done using gradient descent or a pseudo Gauss-Newton scheme. The gradient scheme is

$$w^{\text{new}} = w^{\text{old}} - \eta \cdot \nabla(w^{\text{old}}) \tag{11}$$

†This might be viewed as a maximum a posteriori (MAP) method.

where η is the step-size (line search parameter). The step-szie, η, is determined using simple bisection line search, i.e., η is successively divided by a factor of 2 until a decrease in the cost is observed. For that purpose it is require to compute the gradient, $\nabla(w) = \partial C/\partial w$, of the cost function given by,

$$\frac{\partial C}{\partial w}(w) = -\frac{1}{N_{\text{train}}} \sum_{k=1}^{N_{\text{train}}} \sum_{i=1}^{c-1} [y_i(k) - \widehat{y}_i(k)] \frac{\partial \phi_i(x(k))}{\partial w} + \frac{\alpha_I}{N_{\text{train}}} \cdot w^I + \frac{\alpha_O}{N_{\text{train}}} \cdot w^O \tag{12}$$

The pseudo Gauss-Newton scheme is obtained by

$$w^{\text{new}} = w^{\text{old}} - \eta \cdot \tilde{J}^{-1}(w^{\text{old}}) \nabla(w^{\text{old}}) \tag{13}$$

$\tilde{J}(w)$ is the pseudo Hessian of the cost function defined as the diagonal of the full Hessian, $\tilde{J}(w) = \{J_{mn}\} = \partial^2 C/\partial w \partial w^\top$ where

$$J_{mm}(w) = \frac{1}{N_{\text{train}}} \sum_{k=1}^{N_{\text{train}}} \sum_{i=1}^{c-1} \sum_{j=1}^{c-1} \widehat{y}_i(k)[\delta_{ij} - \widehat{y}_j(k)] \frac{\partial \phi_i(x(k))}{\partial w_m} \frac{\partial \phi_j(x(k))}{\partial w_m} + \frac{\alpha_m}{N_{\text{train}}} \tag{14}$$

where α_m equals α_I or α_O depending on whether the m-th weight is an input-to-hidden or an hidden-to-output weight. Above δ_{ij} denotes the Kronecker delta and the Gauss-Newton approximation to the Hessian is used.

4.2. Pruning

In order to reduce and optimize a neural network architecture, the pruning scheme Optimal Brain Damage (OBD)[3,4] can be used. The aim of OBD is to estimate the importance of the weights for the training error and rank the weights according to their importance which is denoted the *saliency* in OBD terminology. The saliency is estimated using a second order expansion of the training error around its minimum. For weight w_i, the saliency is given by

$$s_i = \left(\frac{\alpha_i}{N_{\text{train}}} + \frac{1}{2} H_{ii} \right) \widehat{w}_i^2 \tag{15}$$

where the H_{ii} is the i-th diagonal element of the Hessian of the un-regularized cost function $S_{\mathcal{T}}(w)$. α_i is the weight decay associated with weight w_i.

The following assumptions enter the derivation of the OBD:

- Terms of third and higher orders for the deleted weights can be neglected.
- The off-diagonal terms in the Hessian can be neglected (if more than one weight is pruned).

By repeatedly removing weights with the smallest saliencies and retraining the resulting network, a nested family of networks is obtained. Here it is possible to use an estimate of the test error to select the "optimal" network.

5. EXPERIMENTAL RESULTS

The data used in experiments, the Anderson River data set, are a multisource remote sensing and geographic data set made available by the Canada Centre for Remote Sensing (CCRS).[27] The imagery involves a 2.8 km by 2.8 km forestry site in the Anderson River area of British Columbia, Canada. The area is characterized by rugged topography, with terrain elevations ranging from 330 m 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.[27] Six data sources were used:

1. Airborne Multispectral Scanner (ABMSS) with 11 spectral 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).
3. Shallow Mode SAR with 4 data channels (X-HH, X-HV, L-HH, L-HV).

Table 1. Training and Test Samples for Information Classes in the Experiment on the Anderson River Data.

Class #	Information Class	Training Size	Test Size
1	Douglas Fir (31-40m)	971	1250
2	Douglas Fir (21-30m)	551	817
3	Douglas Fir + Other Species (31-40m)	548	701
4	Douglas Fir + Lodgepole Pine (21-30m)	542	705
5	Hemlock + Cedar (31-40m)	317	405
6	Forest Clearings	1260	1625
Total		4189	5503

4. Elevation data (1 data channel, where elevation in meters = 61.996 + 7.2266 * pixel value).

5. Slope data (1 data channel, where slope in degrees = pixel value).

6. Aspect data (1 data channel, where aspect in degrees = 2 * pixel value).

The ABMSS and SAR data were detected 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 spatially co-registered with pixel resolution of 12.5 m.

There are 19 information classes in the ground reference map provided by CCRS. In the experiments, only the six largest ones were used, as listed in Table 1. Here, training samples were selected uniformly, giving 10% of the total sample size. Test samples were then selected randomly from the rest of the labeled data.

For the PCNN experiments, the WPT was applied to obtain input vectors for the different stages. Here, the Daubechies 4-point (D4) filters[24] were used for the WPT. Zero-filling was used in the WPT to achieve vectors of length 32 (nearest power of 2). Here, the PCNN with the WPT was trained with the two-layer regularization/pruning neural networks described in Section 4 for the individual SNNs. Each SNN was implemented in the following fashion:

- Network: 33 inputs (one extra input for computing bias in the neurons), 20 hidden neurons and 5 output neurons (one less than the number of information classes).
- The SoftMax threshold was used for each neuron.
- The input data were normalized to zero mean and unit variance in order to facilitate training and weight initialization.
- Weights were initialized uniformly over [-0.3,0.3].
- Weight decay parameters were selected as: $\alpha_i = \alpha_o = 0.001$. It is important for the weight decay parameters to be so small that they do not reduce the approximation capabilities of the network significantly. They should also be large enough in order to prevent the algorithm from being trapped in a local optimum and eliminate numerical instabilities.
- Training is initially done using a gradient descent algorithm for 5 iterations followed by a Gauss-Newton algorithm for 15 iterations. Then pruning of the weights based on OBD is performed for the first time.
- At each pruning operation, two weights are removed.
- Between pruning operations, the network is trained for 50 iterations with a Gauss-Newton algorithm.
- The minimum number of weights in the network was specified as 550.

Table 2. Overall Training and Test Accuracies for the Individual SNNs in the PCNN

SNN #	Training Accuracy	Test Accuracy
1	78.68%	72.36%
2	81.89%	71.94%
3	81.36%	72.42%
4	77.63%	70.07%
5	74.67%	69.20%
6	72.74%	70.94%

Table 3. Overall Training and Test Accuracies for the Classification Methods Applied to the Anderson River Data Set.

Method	Training Accuracy	Test Accuracy
MED	50.51%	50.83%
ML	68.23%	65.12%
CGBP (20 hidden neurons)	73.56%	70.94%
LOP (optimized by a neural network)	71.75%	69.15%
LOGP (optimized by a neural network)	81.62%	74.11%
PCNN (optimized by a neural network)	86.99%	75.40%

Six SNNs were used in each PCNN (corresponding to the number of levels in the full WPT). Training and test classification accuracies for the individual SNNs in the PCNN are shown in Table 2. In the table, the SNNs are numbered from 1 (time domain) to 6 (frequency domain).

For the PCNN, a neural network was used to combine the outputs from the individual SNNs. Using this combination scheme, overall training accuracies of 86.99% and test accuracies of 75.40% were obtained. These accuracies were significantly improved from the accuracies obtained by the individual SNNs and were also significantly higher than the results obtained by PCNNs trained on the same data with a conjugate-gradient backpropagation (CGBP) algorithm[28] for the individual SNNs (overall training accuracy: 76.71%; overall test accuracy: 73.57%).[2,23] The PCNN based on the proposed training scheme for the individual neural networks is not only more accurate than the PCNN based on the CGBP training scheme but it also uses fewer parameters. However, the training time for the proposed approach can be considerably longer than for the CGBP and that was observed in experiments.

The results of the PCNNs, were compared to results for four statistical methods[1]: the minimum Euclidean Distance (MED) classifier, the Gaussian maximum likelihood method (ML), the linear opinion pool (LOP), and the Logarithmic Opinion Pool (LOGP). For the LOP and LOGP, six data classes were defined for each data source and the remote sensing data sources were modeled to be Gaussian but the topographic data sources were modeled by the maximum penalized likelihood method.[29] The outputs from individual source-specific classifiers in the LOP and LOGP were combined using a CGBP neural network.[1] The CGBP algorithm with two layers was also trained on the same data with a different number of hidden neurons. Using 20 hidden neurons gave the most accurate results.

The overall classification accuracies for the different methods are summarized in Table 3. There it can be seen that the PCNN optimized by a neural network outperformed both the single stage CGBPs, and all the statistical methods in terms of overall classification accuracy of test data. In comparison, the maximum overall accuracies for the CGBP method were 73.56% for training data and 70.94% for test data. The maximum overall accuracies for the statistical methods were achieved by the LOGP which gave 81.62% overall accuracy for training data and 74.11% overall accuracy for test data.

6. CONCLUSIONS

In this paper, optimized combination, regularization, and pruning was proposed for individual stage neural networks (SNNs) in the Parallel Consensual Neural Network (PCNN). Experiments were conducted on a multisource remote sensing and geographic data set consisting of 6 data sources. The results obtained by the proposed optimized PCNN were compared to other approaches, such as other versions of the PCNN, individual single stage neural networks, and statistical approaches. The proposed version of the PCNN outperformed all other methods in terms of overall classification accuracy of both training and test data. In the experiments, it was observed that the use of pruning and regularization resulted not only in in lower generalization error when compared to other versions of the PCNN, trained on the same data, but also that the individual SNNs were simpler because of the pruning. However, the training time for the proposed approach was seen to be significantly longer than for other approaches.

Acknowledgements

This work was done in part while J.A. Benediktsson visited the Department of Mathematical Modelling, Denmark Technical University (DTU), Lyngby, Denmark. The research is in part supported by the Icelandic Research Council and the Research Fund of the University of Iceland. 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.

REFERENCES

1. J.A. Benediktsson, J.R. Sveinsson and P.H. Swain, Hybrid Consensus Theoretic Classification, *IEEE Transactions on Geoscience and Remote Sensing* vol. 37, pp. 833-843, 1997.
2. J.A. Benediktsson, J.R. Sveinsson, O.K. Ersoy and P.H. Swain, Parallel Consensual Neural Networks, *IEEE Transactions on Neural Networks* vol. 8, pp. 54-64, 1997.
3. J. Larsen, C. Svarer, L. Nonboe Andersen, L.K. Hansen, "Adaptive Regularization in Neural Network Modeling," to appear in "Book of Tricks" G.B. Orr, K. Muller, R. Caruana, (eds), Springer-Verlag.
4. Nonboe Andersen, J. Larsen, L.K. Hansen, and M. Hintz-Madsen L. Nonboe Andersen, J. Larsen, L.K. Hansen, and M. Hintz-Madsen "Adaptive Regularization of Neural Classifiers," *Procceedings of the IEEE Workshop on Neural Networks for Signal Processing VII*, IEEE Press, 1997.
5. J.A. Benediktsson and Philip H. Swain, "Consensus Theoretic Classification Methods," *IEEE Transactions on Systems Man and Cybernetics*, vol. 22, no. 4, pp. 688-704, July/August 1992.
6. C. Berenstein, L.N. Kanal and D. Lavine, "Consensus Rules," in *Uncertainty in Artificial Intelligence*, L.N. Kanal and J.F. Lemmer (eds.), North Holland, New York, 1986.
7. R.F. Bordley, *Studies in Mathematical Group Decision Theory*, Ph.D. thesis, University of California, Berkeley, 1979.
8. C. Genest and J.V. Zidek, "Combining Probability Distributions: A Critique and Annotated Bibliography," *Statistical Science*, vol. 1. no. 1, pp. 114-118, 1986.
9. R.L. Winkler, "Combining Probability Distributions from Dependent Information Sources," *Management Sciences*, vol. 27, pp. 479-488, 1981.
10. R.A. Jacobs, "Methods for Combining Experts' Probability Assessments," *Neural Computation*, vol. 3, pp. 867-888, 1995.
11. D.H. Wolpert, "Stacked Generalization," *Neural Networks*, vol. 5, pp. 241-259, 1992.
12. D. W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, and B.W. Suter, "The Multilayer Perceptron as an Approximation to a Bayes Optimal discrimination Function," *IEEE Transactions on Neural Networks*, vol. 1, no. 4, pp. 296-298, 1990.
13. M.P. Perrone and L.N. Cooper, "When Networks Disagree: Ensemble Methods for Hybrid Neural Networks," *Neural Networks for Speech and Image Processing*, R.J. Mammone, Ed., Champman-Hill, London, 1993.
14. M.P. Perrone, *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*, Ph.D. Thesis, Department of Physics, Brown University, 1993.
15. K. Tumer and J. Ghosh, "Theoretical Foundations of Linear and Order Statistics Combiners for Neural Pattern Classifiers," to appear *IEEE Transactions on Neural Networks.*

16. N. Nilsson, Learning Machines, McGraw-Hill, New York, 1965.
17. L.K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993-1001, 1990.
18. E. Alpaydin, "Multiple Networks for Function Learning", *Proceedings of the 1993 IEEE International Conference on Neural Networks*, vol. 1, pp. 9-14, San Francisco, 1993.
19. R. Battiti and A.M. Colla, "Democracy in Neural Nets: Voting Schemes for Classification", *Neural Networks*, vol. 7, No. 4, pp. 691-707, 1994.
20. G. Rogova, "Combining the Results of Several Neural Network Classifiers", *Neural Networks*, vol. 7, No. 5, pp. 777-781, 1994.
21. S-B Cho and J.H. Kim, "Multiple Network Fusion Using Fuzzy Logic," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 497-501, 1995.
22. G. Mani, "Lowering Variance of Decisions by Using Artificial Neural Network Portfolios," *Neural Computation*, vol. 3, pp. 484-486, 1991.
23. J.A. Benediktsson, J.R. Sveinsson, J.I. Ingimundarson, H.S. Sigurdsson, and O.K. Ersoy, "Multistage Classifiers Optimized by Neural Networks and Genetic Algorithms," *Nonlinear Analysis, Theory, Methods and Applications*, vol. 30, no. 3, pp. 1323-1334, 1997.
24. R.R. Coifman and M.V. Wickerhauser, "Entropy-Based Algorithms for Best Basis Selection," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 713-718, 1992.
25. I. Daubechies, *Ten Lectures on Wavelets*, CBMS-NFS Regional Conference Series in Applied Mathematics, vol. 61, SIAM, Philadelphia, 1992.
26. M. Hintz-Madsen, L.K. Hansen, J. Larsen, M.W. Pedersen and M. Larsen, "Design and Evaluations of Neural Network Classifiers," Preprint, submitted to *Neural Networks*, 1997. Available by ftp://eivind.imm.dtu.dk/dist/1997/hintz.nn.ps.Z.
27. D.G. Goodenough, M. Goldberg, G. Plunkett and J. Zelek, "The CCRS SAR/MSS Anderson River Data Set," *IEEE Transactions on Geoscience and Remote Sensing*, vol. GE-25, no. 3, pp. 360-367, May 1987.
28. E. Barnard, "Optimization for Training Neural Nets,"*IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 232-240, March 1992.
29. B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Monographs on Statistics and Applied Probability, Chapman and Hall, New York, 1986.