A Feed Forward Neural Network for Classifying Information from Landsat Multi-Spectral Scanner (MSS) Imagery

*Narongdech Keeratipranon

Neural networks, more precisely called Artificial Neural Networks (ANN), are computational models that consist of a number of simple processing units that communicate by sending signals to each other over a large number of weighted connections. They were originally developed from the inspiration of human brains. In a human brain, a biological neuron collects signals from other neurons through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches. At the end of each branch, a structure called *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives an excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

Like a human brain, neural networks also consist of processing units (artificial neurons) and connections (weights) between them. The processing units transport incoming information on their outgoing connections to other units. The "electrical" information is simulated with specific values stored in those weights that make these networks have the capacity to learn, memorize, and create relationships amongst data.

A very important feature of these networks is their adaptive nature where "learning by example" replaces "programming" in solving problems. This feature renders these computational models very appealing in application domains where one has little or incomplete understanding of the problems to be solved, but where training data are available.

There are many different types of neural networks, and they are being used in many fields. New uses for neural networks are devised daily by researchers. Some of the most traditional applications include classification, noise reduction, pattern recognition, and prediction.

Landsat multi-spectral scanner imagery is acquired from taking a photo of a scene with four spectral bands, approximate to green and red regions of the visible spectrums and two are in the infra-red.

Before the information from the imagery can be used, it must be processed to extract each component in the picture. One effective method to retrieve the data is using neural networks.

Fundamental of Neural Networks

1 Artificial Neural

An artificial neural (Figure 1) performs a relatively simple job; it receives inputs from its neighbours or external sources and uses them to compute an output signal that is propagated to other units. The *combination function* is used to calculate the *net input*, a_j by multiplying each input, x_i with its corresponding weight, w_{ji} and add a threshold or bias term, θ_j. The net input can be written as

*Lecturer in the Department of Computer Engineering, Faculty of Engineering, Dhurakijpundit University : B.Eng. (Computer) Chulalongkorn University.

$$a_j = \sum_{i=1}^{n} w_{ji} x_i + \theta_j \tag{1}$$

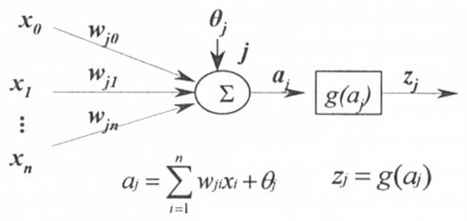

Figure 1: Artificial neural

After that the net input is passed to an *activation function*, yielding a value called unit's activation, z_j. Some of the most commonly used activation functions are

1) identity function

$$g(x) = x \qquad \text{for all } x \tag{2}$$

2) binary step function

$$g(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{3}$$

3) binary sigmoid (Figure 2)

$$g(x) = \frac{1}{1+e^{-x}} \tag{4}$$

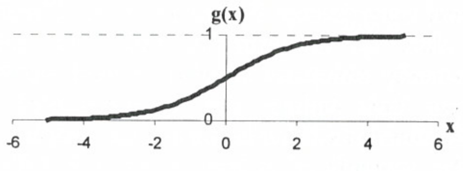

Figure 2: Sigmoid function

4) bipolar sigmoid function (Figure 3)

$$g(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{5}$$

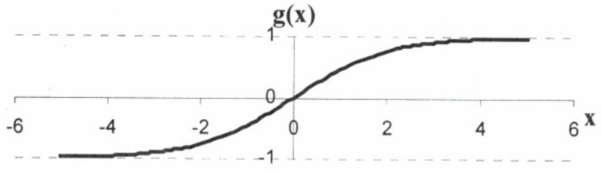

Figure 3: Bipolar sigmoid function

2. Network Topologies

The topology of a network is defined by the number of layers, the number of units per layer, and the interconnection patterns between layers. They are generally divided into three categories based on the pattern of connections :

1) *Feed-forward network* (Figure 4), where the data flow from input units to output units is strictly feed-forward. The data processing can extend over multiple layers of units, but no feedback connections are present. Feed-forward networks are the main focus of this paper. Details will be described in section 2.4.

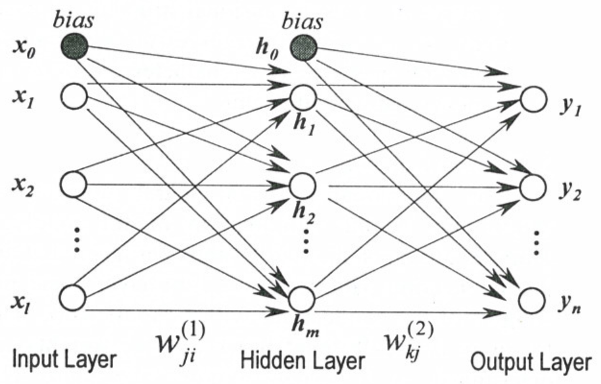

Figure 4: Feed-forward neural network

2) *Recurrent networks* (Figure 5) contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the network will evolve to a stable state in which activation does not change further. In other applications in which the dynamical behaviour constitutes the output of the network, the changes of the activation values of the output units are significant.

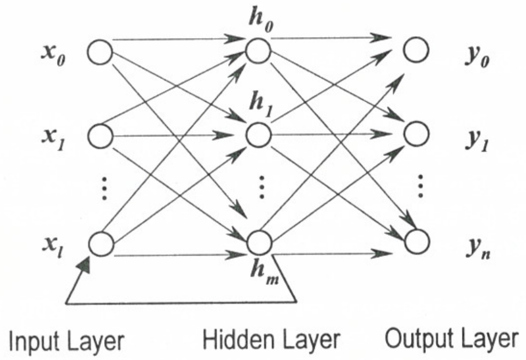

Figure 5: Recurrent neural network

3) *Self-Organising network* is a model for unsupervised learning. In general the model consists of a two-dimensional structure of linear units, where each unit receives the same input. Each unit in the array is characterized by an n-dimensional weight vector.

3. Network Learning

The functionality of a neural network is determined by the combination of the topology (number of layers, number of units per layer, and the interconnection pattern between the layers) and the weights of the connections within the network. The topology is usually held fixed, and the weights are determined by a certain training algorithm. The process of adjusting the weights to make the network learn the relationship between the inputs and targets is called *learning*, or *training*. Many learning algorithms have been invented to help find an optimum set of weights that result in the solution of the problems. They can roughly be divided into three main groups :

1) Supervised Learning - The network is trained by providing it with inputs and desired outputs (target values). These input-output pairs are provided by an external teacher, or by the system containing the network. The difference between the real outputs and the desired outputs is used by the algorithm to adapt the weights in the network.
2) Reinforce Learning – The basic idea of reinforcement learning has its origins in psychology in connection with experimental studies of animal learning. We supply input and response, better or worse, for each action from the system. If the system receives a better signal, then the tendency to produce that action is strengthened. Otherwise, the tendency of the system to produce that action is weakened.
3) Unsupervised Learning - With unsupervised learning, there is no feedback from the environment to indicate if the outputs of the network are correct. The network must discover features, regulations, correlations, or categories in the input data automatically.

4. Feed Forward Neural Networks

A layered feed-forward network consists of a certain number of layers, and each layer contains a certain number of units. There is an input layer, an output layer, and one or more hidden layers between the input and the output layer. Each unit receives its inputs directly from the previous layer (except for input units) and sends its output directly to units in the next layer (except for output units). Obviously, this class of networks is easier to analyse theoretically compared to other general topologies because their outputs can be represented with explicit functions of the inputs and the weights.

An example of a layered network with one hidden layer is shown in Figure 4. In this network there are l inputs, m hidden units, and n output units. The output of the jth hidden unit is obtained by first forming a weighted linear combination of the l input values, then adding a bias,

$$a_j = \sum_{i=0}^{l} w_{ji}^{(1)} x_i \qquad (6)$$

The activation of hidden unit j then can be obtained by transforming the linear sum using an activation function $g(x)$:

$$h_j = g(a_j) \qquad (7)$$

The outputs of the network can be obtained by transforming the activation of the hidden units using a second layer of processing units. For each output unit k, first we get the linear combination of the output of the hidden units,

$$a_k = \sum_{j=0}^{m} w_{kj}^{(2)} h_j \qquad (8)$$

Then applying the activation function $g2(x)$ to (8) we can get the kth output

$$y_k = g2(a_k) \qquad (9)$$

Combining (6), (7), (8), and (9) we get the complete representation of the network as

$$y_k = g2(\sum_{j=0}^{m} w_{kj}^{(2)} g(\sum_{i=0}^{l} w_{ji}^{(1)} x_i)) \qquad (10)$$

The network of Figure 4 is a network with one hidden layer. We can extend it to have two or more hidden layers easily as long as we make the above transformation further.

The feed-forward networks provide a general framework for representing non-linear functional mapping between a set of input variables and a set of output variables. The representation capability of a network can be defined as the range of mappings it can implement when the weights are varied.

1) Single-layer networks are capable of representing only linearly separable functions or linearly separable decision domains.
2) One hidden layered network can arbitrarily approximate well any functional continuous mapping from one finite-dimensional space to another.
3) Two hidden layered networks can represent an arbitrary decision boundary or approximate any smooth mapping in any accuracy.

Though theoretically there exists a network that can simulate a problem to any accuracy, there is no easy way to find it. To define an *exact* network architecture such as how many hidden layers should be used, how many units should there be within a hidden layer for a certain problem is always a painful job. There are two main considerations for the structure: the number of hidden layers and the number of hidden units.

Because networks with two hidden layers can represent functions with any kind of shapes, there is no theoretical reason to use networks with more than two hidden layers. It has also been determined that for the vast majority of practical problems, there is no reason to use more than one hidden layer. Problems that require two hidden layers are only rarely encountered in practice. Even for problems requiring more than one hidden layer theoretically, most of the time, using one hidden layer performs much better than using two hidden layers in practice. When adding more hidden layers, training often slows dramatically and the number of local minima also increases.

Another important issue in designing a network is how many units to place in each layer. Using too few units can fail to detect the signals fully in a complicated data set, leading to *underfitting*. Using too many units will increase the training time and might cause *overfitting,* in which case the network memorises the training examples, but it has not learned to generalise to new situations. The best number of hidden units depends on many factors – the number of input and output units, the number of training cases, the amount of noise in the targets, the complexity of the error function, the network architecture, and the training algorithm. In most situations, there is no easy way to determine the optimal number of hidden units without training using different numbers of hidden units and estimating the generalization error of each. The best approach to find the optimal number of hidden units is through trial and error.

5. Back-Propagation Training

Back-propagation is the most commonly used method for training multi-layer feed-forward networks. It can be applied to any feed-forward network with differentiable activation functions.

For most networks, the learning process is based on a suitable error function which will then minimise the derivative of the error with respect to the weights and bias. There are many methods to minimize the error, *gradient descent, conjugate gradient descent, or newton's method.* The algorithm for evaluating the derivative of the error function is known as *back-propagation*, because it propagates the errors backward through the network.

Methodology and Results

In this paper, the sample of Landsat multi-spectral scanner (MSS) imagery are taken from Department of Statistics and Modeling Science, University of Strathclyde, Scotland U.K., the original Landsat data was generated from data purchased from NASA by the Australian Centre for Remote Sensing, and used for research at The Centre for Remote Sensing University of New South Wales, Australia.

One set of imagery from Landsat MSS consists of four different spectral band images from the same scene. The scene which is used in this paper has a total of 6,435 points and can be divided into seven classes which overlap with each other, shown in Figure 6.

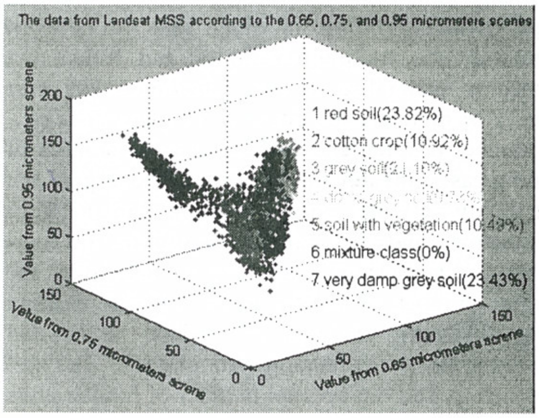

Figure 6: Characteristics of Data

Using feed forward neural networks to classify each pixel in the imagery with each class, the main network architecture can be divided into 3 models;

1) One network with one encoding output. The output must be converted to map each class. For example, if the output falls in the interval -1 and -0.75 it means that point is class 1, red soil. If it is in the interval -0.75 and -0.45 it means that point is class 2, cotton crop.
2) One network with one output for each class. Therefore this network will has seven outputs.
3) Each network for each class. The system will have seven networks with one output.

There are many variables that can affect the performance of the system; number of hidden layers, number of nodes, number of training set, and training method. In the next section, the result of varying each variable will be shown.

1. One network with one encoding output.

When using this model, we first map the outputs to the range between -1 and 1 according to the encoded value in Table 1. The Scaled Conjugate Gradient is then used to train the network because this method is suitable for classifying problems with a large number of nodes and faster than other conjugate gradient because no line search is performed. The networks will use bipolar sigmoid function as activation function for all layers, 500 examples are used as the training set. Figure 7 summarizes the results of training the networks using five different network structures which shown on X-axis (values in parenthesis are time used to train the networks). Each entry in the table represents 10 different trials, where different random initial weights are used in each trial. The simulation is using MATLAB 6.0 Release 12 on Toshiba Satellite 1200, Pentium III 1 GHz, 512 MB RAM.

Table 1: The encoded value for each class

Class	Encoded	Decoded Range
1	-1.00	-1.00 to -0.76
2	-0.60	-0.75 to -0.46
3	-0.30	-0.45 to -0.16
4	0.00	-0.15 to 0.15
5	0.30	0.16 to 0.45
6	0.60	0.46 to 0.75
7	1.00	0.76 to 1.00

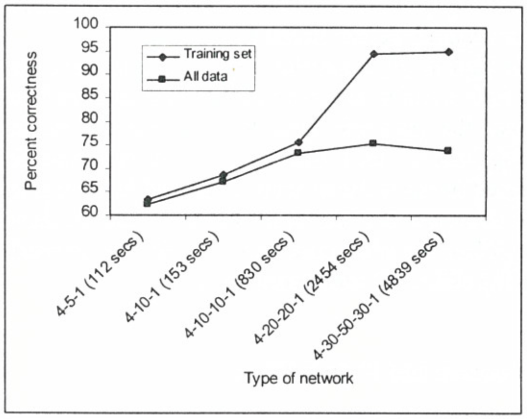

Figure 7:The results of simulation for section 3.1

2. One network with one output for each class.

First, we set up a new matrix with $7 \times n$ dimension which relates to the target output. The values of the matrix determine if the output value at sample i is x, then the value of the matrix at (x,i) is set to 0.5. The other elements in the same column will be set to -0.5. After training the network, the class of data in column i is assigned to the row of the max value in that column from the output of the network. The results from the simulation with the same specification in section 3.1, are shown in Figure 8.

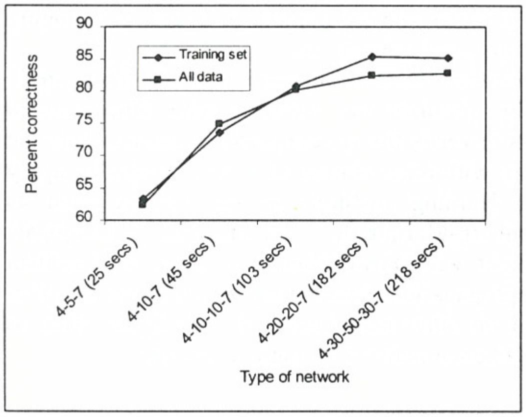

Figure 8:The results of simulation for section 3.2

3. Each network for each class.

By using this method, we will create seven networks with the same structure but different in weights and biases. To determine the class for each pixel, the outputs from each network are combined to build a big matrix with dimension $7 \times n$, same size as the output from the network in section 3.2, and use the same algorithm in section 3.2 to find the exact class for each point. The results are shown in Figure 9.

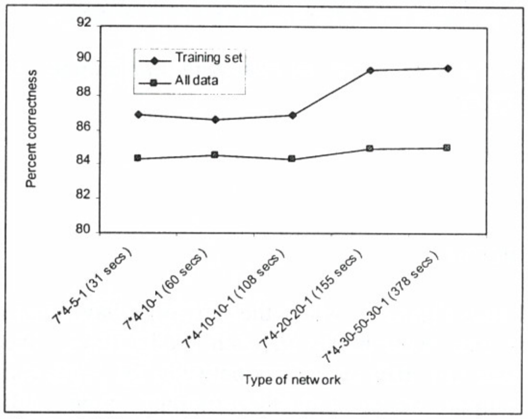

Figure 9:The results of simulation for section 3.3

4. Improve the performance by varying the number of training set.

As we have seen, one of the variables that affect the performance of the system is the number of training sets. In this section, we will study the impact of changing the size of training set. The model in section 3.3, each network for each class, was selected for modification in this study. The simulation result, when the network cannot reduce the mean squared error anymore, is shown in Figure 10.

Figure 10 shows that when the number of training set is 1,000, the percent of correctness is stable and cannot be increased anymore by further enlargement of the number of training set.

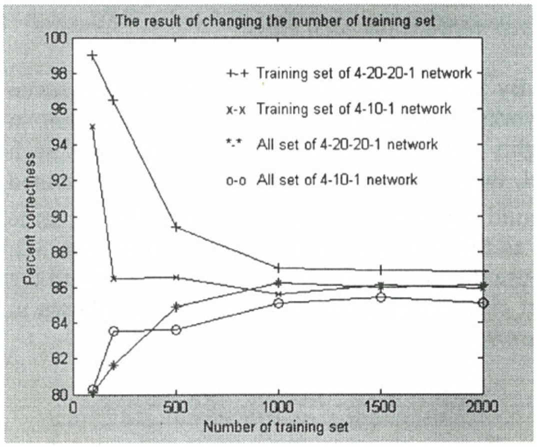

Figure 10: The result of changing the number of training set.

Discussions

From Figure 7, when the networks have more layers and more nodes in each hidden layer, the networks can map the values between the input and the output better but the training time will dramatically increase. When the network has a large number of nodes, in network four and five, the sign for overfitting occurs. The evidence being the percent correction of the data from simulation is obviously decreasing when compared with the percent correction from the training set. Overfitting occurs because the network only remembers the training set but not learning how the output is produced.

In Figure 8, the percent of correction between 4-20-20-7 and 4-30-50-30-7 network is similar except in the number of nodes. Therefore, adding more nodes or hidden layers to this model, the system will not improve its classification performance.

The result from simulating the network in model 3, shown in Figure 9, is obviously better than the other networks. The networks in model 3 are better because they have fewer complexities. Each network just learns to map each class at one time. In contrast the networks in model 1 and 2, one network tries to map all classes at one time.

In Figure 10, when the training set is increased from 100 to 1,000, the performance is seen to reduce because the network tries to map more data in training mode. But the overall performance is increased because the network is trained in a varying range of inputs that cover all kinds of unseen data. Then, when we add more training sets, the network cannot produce a better performance.

Conclusions and Future Work

The feed forward neural network classifier can be used to recognize each point in the Landsat MSS imagery with the high performance. The result is quite good, considering the similar kind of class to recognise: red soil, grey soil, damp grey soil, and very damp grey soil.

The system which composes of many networks with one network for each output seems to be the best model for classifying this kind of imagery, achieving both accuracy and short training time. The network has two hidden layers. Each layer has 20 nodes, training with the Scaled Conjugate Gradient, and the training set is 1,000 is the most suitable in this paper. It gives a correct classification rate of 86.24% for all data sets.

To improve the efficiency of the system, we can modify the network by changing the number of hidden layers and the number of nodes, or changing the number of training set.

Further work is directed towards three areas. Firstly, simulate the network with other unseen inputs to verify the system performance and train the system with other classes of object. Secondly, apply other learning methods to the system such as Levenberg-Marquardt Algorithms, Conjugate Gradient with Powell/Beale Restarts, or Resilient Backpropagation. Finally, explore different solutions based on other neural network paradigms, such as the Kohonen network or logical neurons.

บรรณานุกรมอยู่หน้า 112