

Interpolation of Sparse Digital Elevation Model Using Back Propagation Neural Networks

Komal Kachru, MPT Chamy and Santanu Chowdhury

Advanced Image Processing Division Space Applications Centre, Ahmedabad - 380 015

Email: {komal, mpt, santanu}@ipdpg.gov.in

Keywords: DEM interpolation, neural networks, back propagation algorithm

Abstract

Neural networks provide an information processing paradigm suitable for a host of problems including two-dimensional interpolation of sparse digital elevation data. Neural networks are applicable where there is no algorithmic solution or for which algorithmic solutions are too complex to be found or parameters that describe the algorithmic processes are not easy to identify or quantify. Sophisticated algorithms for two-dimensional interpolation include Kriging or Gerchberg techniques. Neither the estimation of semi-variogram for Kriging nor the frequency characterisation for the Gerchberg model is easy to quantify. Neural networks are good when there is a need to pick out the structure from the existing data. Neural networks can probe through the sparse DEM data and learn relationship not readily apparent otherwise. Neural networks are robust and can handle noisy and missing data and has the ability to generate from small training set. In this paper we explore the use of multi-layer feed forward neural network trained by the *back propagation* algorithm. These networks can learn arbitrary mappings or classifications. The ability to learn mappings allows us to use it as an interpolator. In this paper, we show how the network was trained successfully and tested on simulated and real terrain in the form of contours and randomly distributed height points. However, we find that the time required for learning is high when the size of the training set is large or when the DEM data has inherently more complexity. In order to reduce learning time, we propose a technique for generating separate Neural networks for overlapped sub-regions and blending DEMs obtained from these overlapped regions.

Overview

1. Introduction
2. Artificial Neural Networks
3. The Back Propagation Algorithm
4. The Back Propagation Neural Network Software
5. Performance Evaluation
 - 5.1 Interpolation of Simulated Contours
 - 5.2 Interpolation of Real Contour Data
 - 5.3 Interpolation of Natural Elevation Surface
 - 5.4 Blending of Interpolated Images
6. Conclusions
7. References

1. Introduction

Digital elevation data is often obtained in the form of contour or a sparse data set. However it is required in a regular grid form for a number of applications. Various techniques have been employed in the past for interpolation of sparse DEM. The distance weighted average methods are the simplest to implement but can result in a large interpolation error. The techniques known to provide better interpolation accuracies are the Gerchberg technique and the Kriging algorithm. However, the spectral characteristics of the elevation data are difficult to be specified accurately apriori as required by the Gerchberg algorithm. There is also no unique way to estimate semivariogram parameters for the Kriging algorithm. In this paper, we have explored the use of Neural networks for interpolation of DEM data. The characteristics of the neural network paradigm are discussed. A software for two-dimensional interpolation has been developed using the back propagation algorithm and its performance evaluated.

2. Artificial Neural Networks

An artificial neural network (ANN), also referred to as connectionist architectures, parallel distributed processing and neuromorphic systems, is an information processing paradigm inspired by the way the densely interconnected, parallel structure of the mammalian brain processes information (Fig. 3). ANN is a collection of mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. Neural network is composed of a large number of processing elements (neurons) working in unison to solve specific problems. The functions of a neural network is determined by a network structure, connection strengths and the processing performed at the computing elements or nodes. A neural network is thus a massively parallel-distributed processor that has a natural propensity for storing experimental knowledge and making it available for use (Vemuri, 1992). Learning in biological systems involves adjustments to the synaptical connections that exist between the neurons. The ANN resembles the brain in two respects:

- 1) Knowledge is acquired by the network through a learning process
- 2) Interneuron connection strengths known as synaptic weights are used to store the knowledge.

Unlike Von Neuman machines which rely on processing, memory abstraction of human information processing - the neural networks - are a form of multiprocessor computer system with simple processing elements, with high degree of connections between processing elements, conveying simple scalar messages and having adaptive interaction between elements.

Neural networks are tools that let us build behavior models starting from a collection of examples. Neural net, ignorant at the start, will through a learning process, become a model of the dependencies between the descriptive variables and the behavior to be explained. This model is automatically and straightforwardly built from the data. No skilled and costly techniques such as an expert, statistician or knowledge engineer are required. The traditional algorithmic methodologies, a programmer or an analyst specifically codes every facet of the problem in order for the computer to understand the situation. Neural networks do not require the explicit coding of the problem. Neural networks on the other hand, self adapt to learn from information, providing powerful models representing knowledge about a specific problem (Haykin, 1994). Since design of Neural networks are not dependent on problem specific algorithms, they are well suitable for all applications where there is a need for identifying patterns or trends in data or a need for prediction or forecasting, including sales forecasting, industrial process control, customer research, data validation, risk management and target marketing. Currently Neural networks are used for recognition of speakers, diagnosis of hepatitis, interpretation of multi meaning Chinese words, under sea detection, texture analysis, three dimensional object recognition, hand-written word recognition and facial recognition.

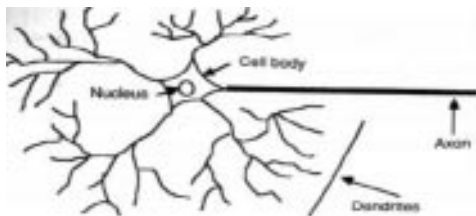


Fig. 1: Components of a Neuron

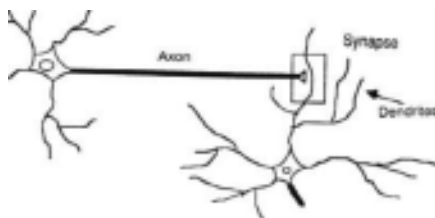


Fig. 2: The Synapse

In the human brain, a typical neuron (Fig. 1) collects signals from others 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 a *synapse* (Fig 2) converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives 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.

Neural networks are typically organised in layers (Fig 4). Layers are made up of a number of interconnected 'nodes' which contain an 'activation function' (Lippman, 1987). Patterns are presented to the network via the input layer, which communicates to one or more 'hidden layers', where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is provided as shown in Fig 4.

Most ANNs contain some form of 'learning rule', which modifies the weights of the connections according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts; a child learns to recognise dogs from examples of dogs.

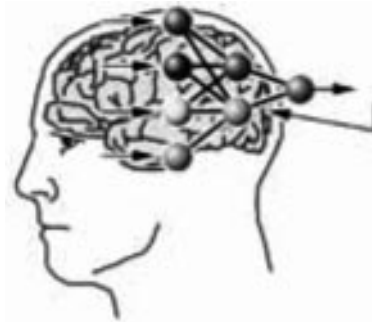


Fig. 3: Neurons in Human Brain

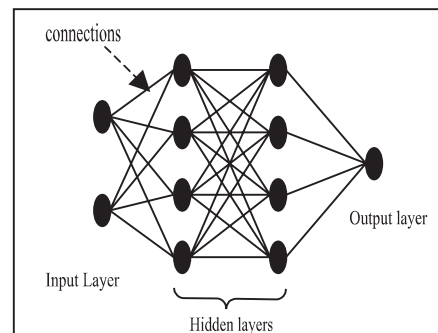


Fig. 4: Basic Architecture of Neural Network

Artificial Neural networks are typically composed of interconnected units, which serve as model neurons. The function of the synapse is modified by a modifiable weight, which is associated with each connection. Each unit converts the pattern of incoming activities that it receives into a single outgoing activity that it broadcasts to other units. It performs this conversion in two stages

(Fig 5). First, it multiplies each incoming activity by the weight on the connection and adds together all these weighted inputs to get a quantity called the total input. Secondly, a unit uses an input-output function (called

'activation function') that transforms the total input into the outgoing activity. Here, we have used Sigmoid function (Fig 6) as the input-output function.

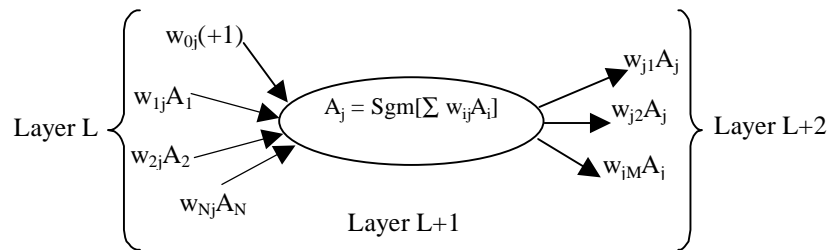


Fig. 5: Activation Output of Neuron j

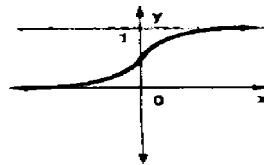


Fig. 6: Sigmoid Function $Sgm(x) = \frac{1}{1 + e^{-x}}$

3. The Back Propagation Algorithm

The capabilities of the multi-layer perceptron can be viewed from three different perspectives. The first has to do with its ability to implement boolean logic functions, the second with its ability to partition the pattern space for classification problems, and the third, with its ability to implement non-linear transformations for functional approximation problems.

We can teach a multi-layer network to perform a particular task by using the following procedure.

1. Present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
2. Determine how closely the actual output of the network matches the desired output.
3. Change the weight of each connection so that the network produces a better approximation of the desired output.

The back propagation procedure is a relatively efficient way to compute how much performance improves with individual weight changes (Hush & Horne, 1993). The procedure is called the *back propagation* because, it computes changes to the weights in the final layer first, reuses much of the same computation to compute changes to the weights in the penultimate layer, and ultimately goes back to the initial layer. The goal of this method is to adjust the synaptic weights so as to minimise the mean squared error between the desired

and actually obtained output taken over all the known elevation points. The learning process is iterative in nature with each iteration consisting of a forward and backward pass (Widrow & Lehr, 1990). In the forward pass the outputs are evaluated and the error at the output units calculated. In the backward pass, the output unit error is used to alter weights on the output units based on the mean squared error criteria. Then the error at the hidden nodes is calculated and the weights on the hidden node altered using these values. Fig 7 shows the forward pass (bold arrows) and the backward pass (dashed arrows) paths.

The overall idea behind back propagation is to make a large change to a particular weight, w , if the change leads to a large reduction in the errors observed at the output nodes (Zurada, 1992). For each sample input combination, consider each output's desired value, d , its actual value, o , and the influence of a particular weight, w , on the error, $d-o$. A big change to w makes sense if that change can reduce a large output error and if the size of that reduction is substantial. On the other hand if the change to w does not reduce any large output error substantially, little should be done to that weight.

The computation needed to compute the change to any particular weight is also needed to compute the changes to weights that are closer to the output nodes. It is seen that, in order to compute a change for a typical weight, $w_{Li \rightarrow (L+1)j}$, between a node in layer L and a node in layer $(L+1)$, the required computations involve computations needed for the weights, $w_{(L+1)j \rightarrow (L+2)k}$, between nodes in layer $(L+1)$ and nodes in layer $(L+2)$.

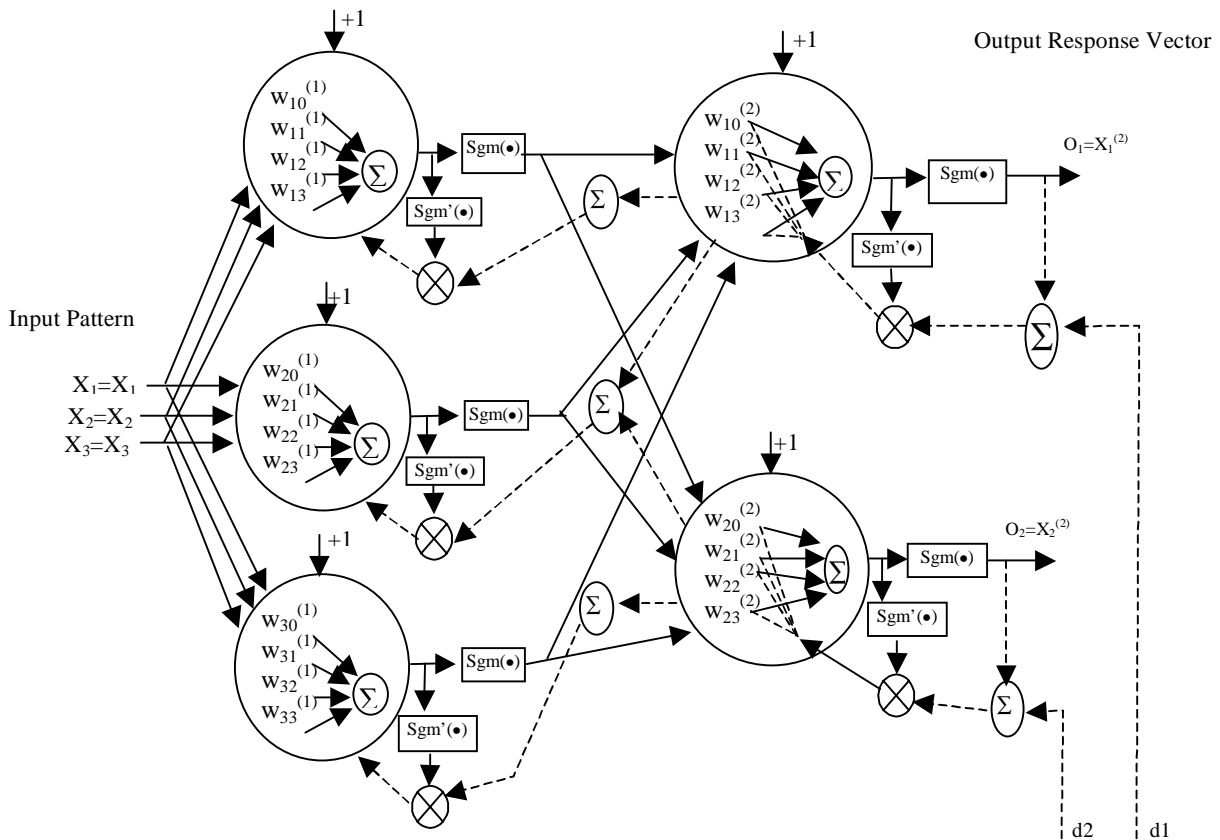


Fig. 7: Weight Updating in a 2-Layer Back Propagation Network

A change in input to node j results in a change in output at node j that depends on the slope of the threshold function. Where the slope is steepest, a change in the input has the maximum effect on the output. Accordingly, the change in $w_{i \rightarrow j}$ is arranged, to depend on the slope of the threshold function at node j on the ground that change should be liberal only where it can do a lot of good. The slope of the squashed S function is given by a particularly simple formulae, $o(I-o)$. Thus, the use of the squashed function as the threshold function leads to the following conclusion about changes to $w_{Li \rightarrow (L+1)j}$:

- Let the change in $w_{Li \rightarrow (L+1)j}$ be proportional to $o_j(I-o_j)$. The change in the input to node j , given a change in the weight, $w_{Li \rightarrow (L+1)j}$, depends on the output of node i . Again that change should be liberal only where it can do a lot of good, and $w_{Li \rightarrow (L+1)j}$ to change substantially only if the output of node i is high.
- The change in $w_{Li \rightarrow (L+1)j}$ be proportional to o_i , the output at node i .

Putting these considerations together, it seems that the change to a weight, $w_{Li \rightarrow (L+1)j}$, should be proportional to o_i and to $o_j(I-o_j)$, and to a factor, β , that captures how beneficial it is to change the output of node j . Thus, the change to $w_{Li \rightarrow (L+1)j}$ should be proportional to $o_i \times o_j(I-o_j) \times \beta_j$.

Suppose that the node j of layer L be connected to just one node k of layer $L+1$. Since, the change should be liberal only where it can do substantial good, the change to o_j should be proportional to $o_k(I-o_k)$, the slope of the threshold function at node k . For the same reason, the change to o_j should be proportional to $w_{Lj \rightarrow (L+1)k}$ the weight on the link connecting node j to node k . Of course, node j is connected to many nodes in the next layer, hence the overall benefit obtained by changing o_j must be the sum of the individual effects. Thus, the benefit that we obtain by changing the output of node j is summarised as follows:

$$\beta_j = \sum_k w_{Lj \rightarrow (L+1)k} o_k(I-o_k) \beta_k \quad \dots (1)$$

At this point, recall that the change to $w_{Lj \rightarrow (L+1)k}$ is proportional to β_j . Evidently, the weight change in any layer of weights depends on a benefit calculation, β_j , that depends, in turn, on benefit calculations, β_k , needed to deal with weights closer to the output nodes.

The value of β depends on how wrong the output node's value happens to be. If the difference between d_z , the desired output at node z , and o_z , the actual output at the same node, is small, then the change in the output at node z should be relatively small. On the other hand, if the difference is large, the change in the output at node z should be large. Accordingly, the appropriate change to o_z should be in proportion to the difference, $d_z - o_z$ implies that

$$\beta_z = d_z - o_z \quad \dots (2)$$

Finally, weight changes should depend on a rate parameter, μ , that should be as large as possible to encourage rapid learning, but not so large as to cause changes to output values that considerably overshoot the desired values. Combining all equations, we have the following Back Propagation formulas:

$$\delta_{w_{Lj \rightarrow (L+1)k}} = \mu o_j (1 - o_j) \beta_j \quad \dots (3)$$

$\beta_j = \sum_k w_{Lj \rightarrow (L+1)k} o_k (1 - o_k) \beta_k$ for nodes in hidden layers,

$\beta_z = d_z - o_z$ for nodes in the output layer.

The network is allowed to run for many epochs, till the network converges, i.e. the mean square error of the training set is within the threshold. Thus in each epoch, the network tries to learn through the training set, i.e. all the patterns in the training set. The total mean square error of an epoch is calculated as

$$E = \frac{1}{2} \sum_{p=1}^P E_p(w) \quad \dots (4)$$

where, E_p is the error calculated for each pattern in the training set and P is the total number of patterns in the training set. E_p is calculated as

$$E_p(w) = \frac{1}{2} \sum_{i=1}^{N_L} (d_i - o_i)^2 \quad \dots (5)$$

where, N_L is the total number of nodes in the output layer.

4. The Back Propagation Neural Network Software

We have developed a user-friendly neural network software, which can be trained using the Back Propagation algorithm (Hush & Horne, 1993). The software can handle arbitrary number of inputs and outputs and can be configured for arbitrary number of layers and number of neurons per layer. This is written in a modular fashion in C language with callable routines for configuring the network, training the network for known patterns and using the network for unknown patterns. This software has been used for both classification and mapping problems. For two-dimensional interpolation, the number of input nodes are two (the spatial coordinates) and the number of output node is one (the elevation value at those spatial coordinates). The training set uses the two spatial coordinates as the input to the network and elevation values at known spatial positions as the desired output. The weights obtained at convergence can be used to interpolate the data at any spatial coordinate. The mean square error value between the elevation values in the training set and that obtained by the neural network provides a measure of learning ability of the neural network. The choice of the number of layers and the number of neuron in each layer play a vital role in obtaining a low mean square error value.

5. Performance Evaluation

The network was trained and tested on three cases, namely simulated contours, real contour data and irregular set of data points. We also show how overlapped regions trained by different Neural networks can be seamlessly blended.

5.1 Interpolation of Simulated Contours: Elevation contours were simulated as shown in Fig 8A. It consists of concentric circles of different radii and concentric ellipses of different semi-major and semi-minor axes. Different circular and elliptical contours represent different elevation values. Spatial coordinates of the contours were fed to the network as input and the elevation values at those spatial coordinates were fed as desired output to the network. The network was allowed to run for different node configurations and epochs. Interpolated results were obtained, once the network converged. 2-D interpolation results of simulated contours obtained with a 3-layer network, having 26 nodes in first layer, 19 nodes in second layer and one output node is shown in Fig. 8B. This network has converged after 5000 epochs and this resulted in a mean square error, which is 0.94% of the maximum elevation value in the dataset.

5.2 Interpolation of real contour data: The real elevation contours, as shown in Fig. 9A, are of a small portion of Australian landscape. Spatial coordinates of the contours were fed to the network as input and the elevation values at those coordinates as desired output. The 2-D interpolation of these natural contours was obtained after training the network for different network configurations.

Two-dimensional interpolation of natural contours obtained with a 3-layer network having 15 nodes in first layer and 10 nodes in second layer is as shown in Fig. 9B. The mean square error obtained was 2.4% of the maximum elevation value. The accuracy obtained was not satisfactory. A neural network of higher complexity was required. Interpolation was thereafter carried out with a 3-layer network having 25 nodes in the first layer and 20 nodes in the second layer and the interpolation results are shown in Fig. 9C. The mean squared error was 1.56% of the maximum elevation value. Thus interpolation error could be reduced by employing a neural network with more number of nodes. Fig. 9D gives the error pattern for the back propagation training network for the network configuration, as shown in Fig. 9C. We observed that the mean squared error parameter has stabilized after 5000 iterations.

5.3 Interpolation of Natural Elevation Surface: The Natural Surface, as shown in Fig. 10A, is of a small portion of the Australian landscape. 20% random points (spatial coordinates) from the image were fed to the network for training. These random points are uniformly distributed (as shown in Fig. 10B) over the image extent. The 2-dimensional interpolation for the natural surface was obtained with different network configurations. Interpolation with a neural network having 15 nodes in the

first layer and 10 nodes in the second layer yielded a mean square error which is 1.81% of the maximum elevation value. Interpolation with a neural network having 25 nodes in the first layer and 20 nodes in the second layer yielded a mean square error which is 1.73% of the

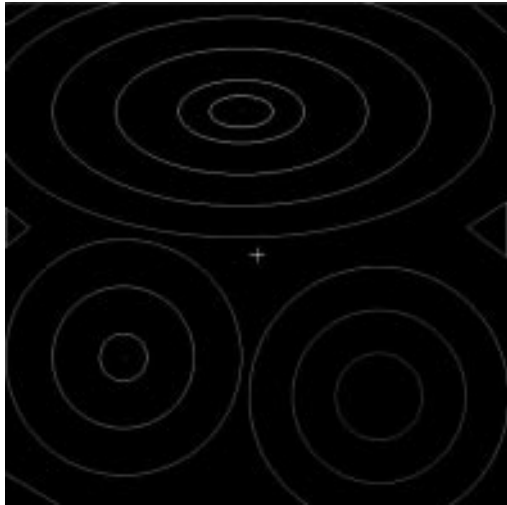


Fig. 8A: Simulated Contours, Input Pattern

maximum elevation value in the dataset. Interpolated results are shown in Fig. 10C & 10D. It was observed that two small bright regions were missing in the interpolated results. It was analyzed and found that it is due to inadequate representation in the training samples.

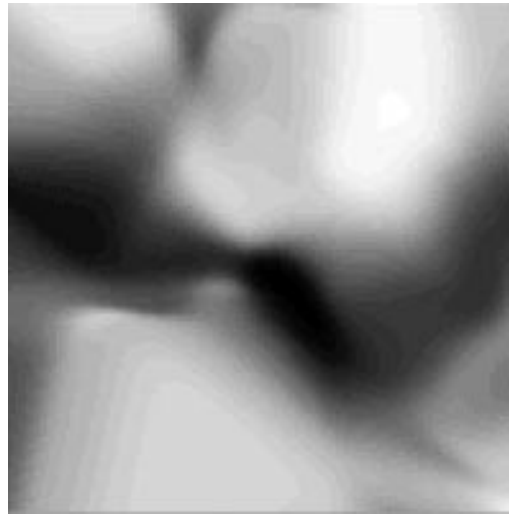


Fig. 8B: 2-D interpolation of simulated Contours

Network configuration :
 Number of layers : 3
 Number of nodes in :
 1st hidden layer : 26
 2nd hidden layer : 19
 Output layer : 1

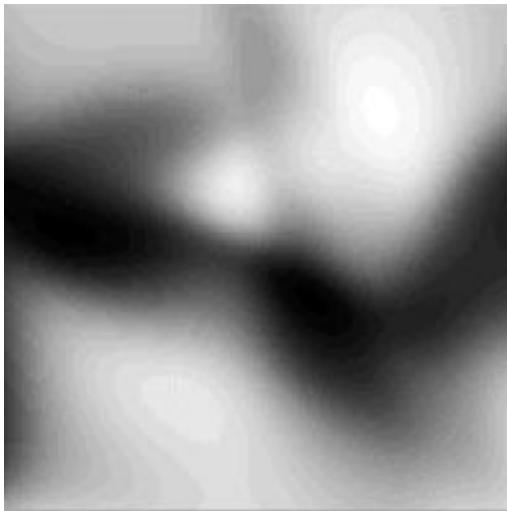


Fig. 9C: 2-D interpolation of Natural Contours

Network configuration :
 Number of layers : 3
 Number of nodes in :
 1st hidden layer : 25
 2nd hidden layer : 15
 Output layer : 1

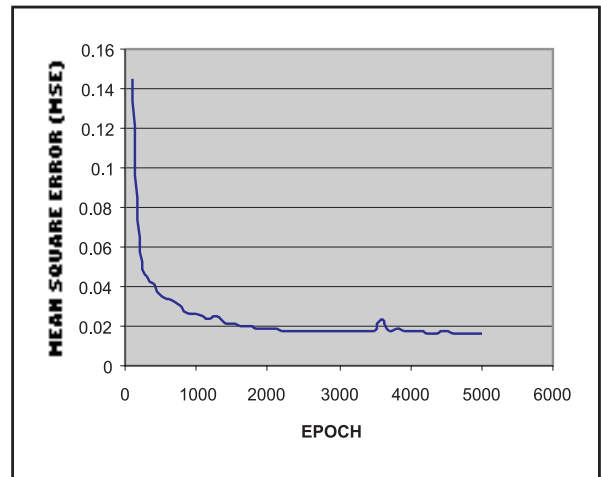


Fig. 9D: MSE expressed as a fraction of maximum elevation value for 2-D interpolation of Natural Contours

Network configuration :
 Number of layers : 3
 Number of nodes in :
 1st hidden layer : 25
 2nd hidden layer : 15
 Output layer : 1

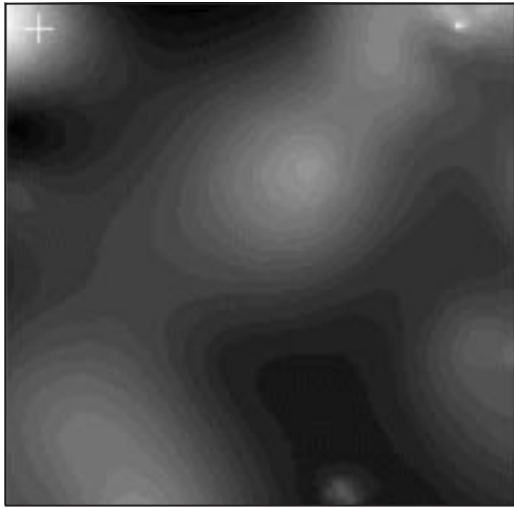


Fig. 10A: Natural Surface, Input Pattern

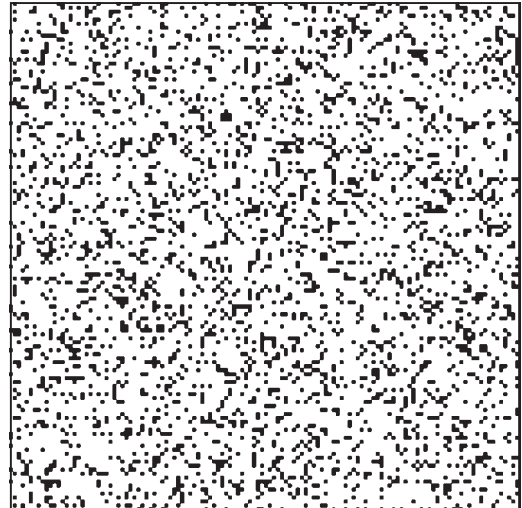


Fig. 10B: 20 % Random Sample Points of the Input Image For Network Training

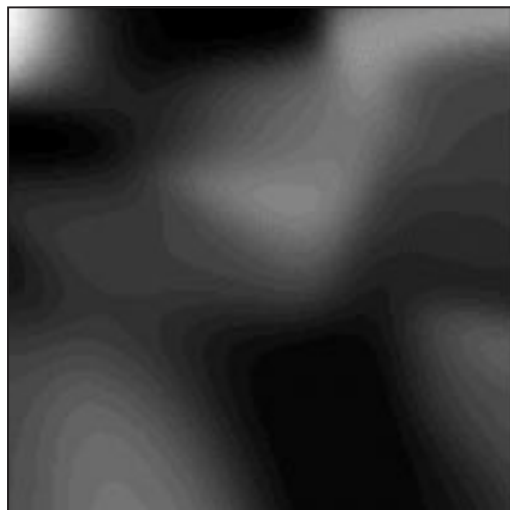


Fig. 10C: 2-D interpolation of Natural Surface from 20% data points

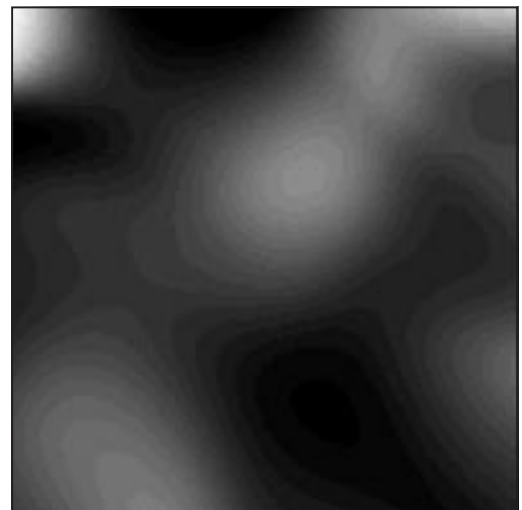


Fig. 10D: 2-D interpolation of Natural Surface from 20% data points

Network configuration :
 Number of layers : 3
 Number of nodes in :
 1st hidden layer : 15
 2nd hidden layer : 10
 Output layer : 1

Network configuration :
 Number of layers : 3
 Number of nodes in :
 1st hidden layer : 25
 2nd hidden layer : 20
 Output layer : 1

5.4 Blending of Interpolated Images: In order to obtain digital elevation model of large areas, we can obtain interpolated sub-regions by employing separate Neural networks and form a mosaic. However, if the regions are non-overlapped, discontinuities are observed at boundaries of the regions. Hence, it is preferable to take overlapped regions. Two regions with 50% overlap was trained using

two separate Neural networks and their interpolated results are shown in Fig 11A and Fig. 11B. The result

obtained by carrying out a distance weighted blending is shown in Fig. 11C. We observe that there are no discontinuities at region boundaries.

6. Conclusion

We have demonstrated the performance of the neural network back propagation software for interpolation of 2-dimensional sparse DEM data. However, as the size of the data set increases or as the complexity of data increases, a neural network with a large number of neurons has to be configured. This requires a large training sample

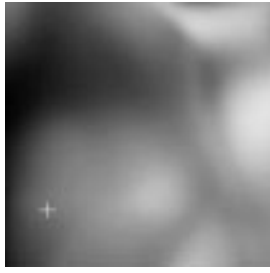


Fig. 11A: Interpolated DEM of the top region

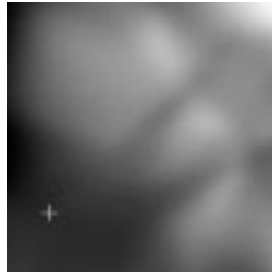


Fig. 11B: Interpolated DEM of the bottom region

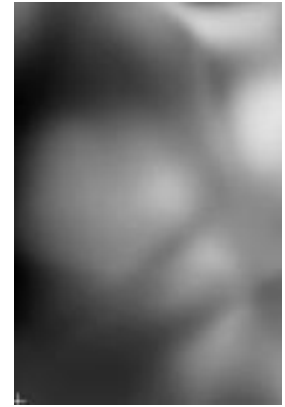


Fig. 11C: Interpolated DEM Obtained after the merging 50% overlapping images of Fig 11A and 11B

which is difficult to provide. Further, the time required for training becomes very high. In order to avoid these problems we have blended interpolated DEMs obtained from Neural networks of overlapped regions. This provides a practical methodology for implementing two-dimensional interpolation using neural networks.

7. References

- [1] Lippman, Richard P., "An introduction to computing with neural nets" IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. 37, April 1987.
- [2] Haykin, Simon, "Neural networks: A Comprehensive Foundation", Prentice Hall, Upper Saddle River, NJ, 1994
- [3] Hush, Don R., and Bill Horne, "Progress in supervised neural networks: What's new since Lippman" IEEE Trans. on Signal Processing, Vol. 41, January 1993.
- [4] Vemuri, V. Rao, "Artificial Neural networks: Concepts and Control Applications", IEEE Computer Society Press, Los Alamitos, CA, 1992.
- [5] Widrow and Lehr, " Perceptron, Madaline, and Back Propagation", Proceedings of IEEE, Vol. 78, 1990.
- [6] Zurada, Jacek M., "Introduction to Artificial Neural Systems", Jaico Publishers, Bombay, 1992