KOHONEN'S SELF ORGANIZING NETWORKS WITH "CONSCIENCE"

By: Dr. M. Turhan (Tury) Taner, Rock Solid Images November 1997

INTRODUCTION:

The effectiveness of the interpretive use "Seismic Attributes" depends on the discrimination ability of the chosen set of attributes. As in many mathematical problems, some attributes may be necessary; a few may be sufficient. This could be determined by experimenting with logical combinations of various mixtures of attributes. In cases where well logs or lithological columns are available, Feed Forward fully connected Neural Networks may be trained in a supervised manner. In cases where no well information is available, we will have to use some unsupervised methodology to cluster the data and check to see if it will have any relation to our experience based interpretation.
In our case the data is the computed set of attributes. The user selects a set of attributes to be used in clustering or in classification. There are several unsupervised clustering methods to do the job. One of the simplest is to cluster the data sequentially. Assume that we are given a set of data traces, where each sample is represented by a vector of attributes. This can be viewed as data represented in an N dimensional space, where each attribute is one of the axes. In the beginning we compute a simple mean of all of the attributes; then we collect all of the data samples within some distance to this mean value. As we are checking all of the samples, we can identify the data samples within the allowed distance as members of the first cluster and take them away from the further computation. After all of the data sets are computed we have formed the first cluster and identification. While this is going on we can compute the average of the rejected samples, which will represent the mean of the second set. We continue clustering in this manner until no significant data samples are left.

The problem with this procedure, although it may be fast, is that all clusters are arbitrary and no reasonable relationship is maintained between the clusters. This feature, as we will discuss later, represents an important aspect of Kohonen's method where clusters are formed in such a way that they relate to one another in an organized manner. For example, clustering for the sand, sandy-shale, shaly-sand and shale cluster may occupy adjacent locations on the organized map. This is analogous to the human brains "associative memory".

Another, more classical approach is the development of statistics of the data set by forming the covariance matrix and computing the principal components. This will give us Eigen values and Eigen vectors from the most to the least important. The Eigen vector corresponding to the most significant Eigen value will represent the attribute combination which is most abundant in the data set. The next Eigen value will represent the next most abundant and so on. Once these are determined, we can classify the data set by computing the distance between the data set and the Eigen vectors and selecting the one with the minimum Euclidean distance. This method will form clusters based on population and may not have any relationship to adjacent Eigen values and vectors. Furthermore, we will have to solve a very large matrix, composed of correlations between all pairs of the data set.

KOHONEN'S ANALOGY:

Kohonen's self-organizing maps (SOM) are simple analogs of the human brain's way of organizing information in a logical manner. Research has shown that the cerebral cortex of the human brain is divided into functional subdivisions. The cortex contains billions of neurons with many billions of connections (synapses) between them. The subdivision of the cortex is an orderly manner. The most important ones are motor cortex, sensory cortex, visual cortex and auditory cortex. For example, the auditory cortex is also subdivided into many cells, each functioning as cognitive parts for some auditory signal. It is theorized that some of these are trained in a supervised manner, and some are developed in an unsupervised self-organizing manner. Topologically adjacent areas perform somewhat related cognitive functions. Kohonen's method emulates the unsupervised learning in an elegant, extremely simple manner. During the self-organizing procedure the topologically close relationship of the organized information is maintained. Initially, a large area is treated in a similar fashion. Later in the iteration this zone shrinks. A number of other computational procedures were introduced after the original Kohonen algorithm. Since "***winner take all***" logic

may result in one neuron dominating the training, some modification is necessary. I will describe the original method with only one simple modification, called the " ***conscience*** ".
Kohonen's method consists of one layer of neurons and uses the method of competitive learning with "***winner take all*** ". The modification looks at the winning rate of each neuron, and if it sees any neuron dominating, it uses the "*conscience*" algorithm and lets other neurons have a chance in the learning procedure.

METHOD:

As in the case of covariance studies, input to the self-organizing maps will have to be normalized. That is each set of attributes has to be scaled so that their RMS value will be equal to unity. This will prevent any attribute arbitrarily dominating the clustering.
Kohonen's self-organizing maps consist of one layer of neurons organized in one, two and multi-dimensional arrays. Each neuron has as many input connections as there are number of attributes to be used in the classification, as shown on figure 1. The training procedure consists of finding the neuron with weights closest to the input data vector and declaring that neuron as the winning neuron. Then the weights of all of the neurons in the vicinity of the winning neuron are adjusted, by an amount inversely proportional to the distance. The radius of the accepted vicinity is reduced as the iteration numbers increase. The training process is terminated if RMS errors of all of the input are reduced to an acceptable level or a prescribed number of iterations are reached.
There are two methods to determine the similarity. In the first method, each input is weighted by a neuron's corresponding weight vector and the results are summed. This represents the net input of the particular neuron. Let k represent the k'th neuron and N attributes are used, then the net input will be (in terms of a vector scalar product);

$$net_k = \sum_{i=1}^{N} [x(i).w(i,k)] \qquad \textbf{(1)}$$

The vector scalar product (1) will give the projection of one vector on to the other. If the unit vectors are defined in the x(i) and w(i) directions, their dot product yields the cosine of the angle between **x** and **w** vectors. This is one of the reasons we normalize each set of attributes to have the zero mean and their RMS amplitude be unity. A net input of 1 would represent two collinear vectors; they point in the same direction and their parameters are similar. A value of 0 will mean that two vectors are perpendicular to each other, thus they are not similar. The second method to measure the similarity of two vectors is to compute the Euclidean distance between two vectors, as;

$$net_k = \sqrt{\sum_{i=1}^{N} [x(i) - w(i,k)]^2} \qquad \textbf{(2)}$$

In this case a net result of zero will mean that the two vectors are identical. A value of close to twice to their normalized amplitudes will mean that they are in opposite directions, thus they are not similar. These are repeated for all of the neurons and the neuron with largest output of dot product or the minimum Euclidean distance is chosen as the winner.
In order to maintain similarity of topologically close neurons, weights of all neurons within a selected radius are adjusted. The winning neuron is adjusted by the most amount, which will bring the weights of the winning neuron closer to the input data values. All other neurons are adjusted by lesser amounts, inversely proportional to their distance from the winning neuron. As the iteration progresses and RMS error reduces, the radius of correction is also gradually reduced. This will eventually become one neuron distance; thus no other neuron is adjusted other than the winning neuron. In the case where one neuron is continually the winning neuron, then its computed distance is also modified by some amount to allow other neurons win. This process is called the "***conscience***".
At the conclusion of training, each neuron's weight represent reference data, which could later be used for classification. If the input is seismic attributes, then the weights represent the reference attributes (i.e. the mean) of each classification (i.e. cluster). The classification process at the conclusion of training is then, the computation of each input data sets distance to each of the neurons and classify the input as belonging to the class represented by the winning neuron.

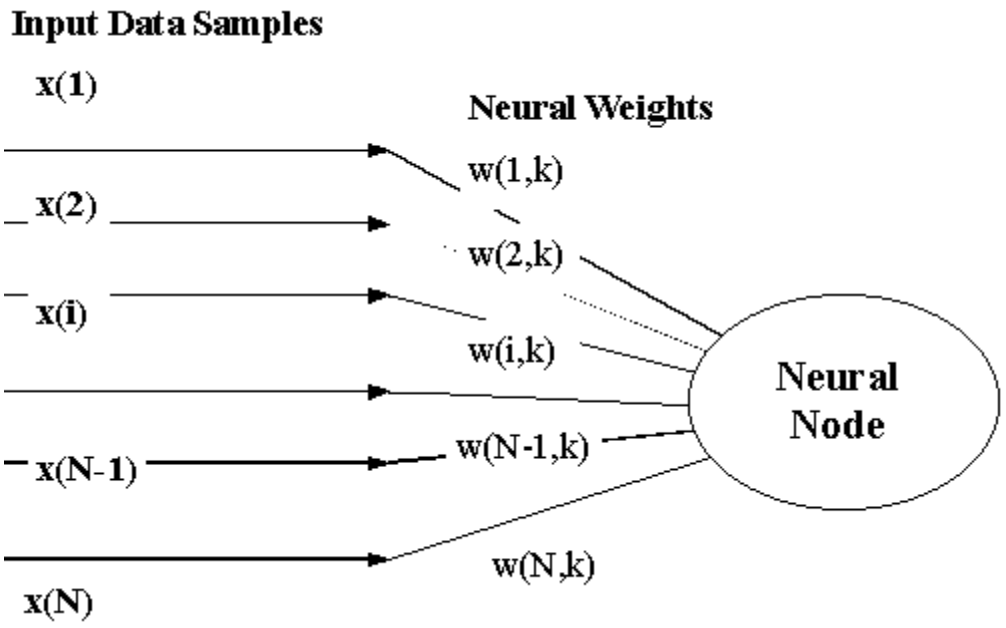

Figure 1. Connection of input data to a Neural Node

TOPOLOGICAL ORGANIZATION:

Neurons can be organized in any topological manner. Figure 2 shows a one-dimensional organization. This will allow us clustering with a one-dimensional topological relationship. That is, each adjacent neuron will have smaller differences than the neurons two spaces away. The difference will gradually increase with increasing distance. Each neuron is connected to the input data with their own weights. In the case of the SOM these weights will be equivalent to the normalized attributes representing the centroid coordinates of each cluster. This feature of the SOM is very different than the feed forward fully connected Artificial Neural Networks, where the weights do not relate directly to the input data.

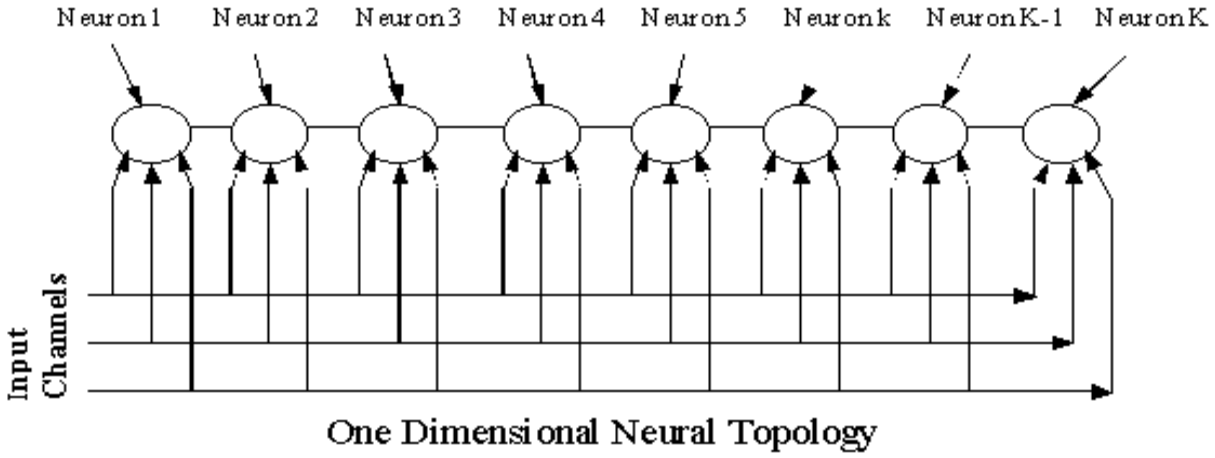

Figure 2 Input to One Dimensional Neural Topology

The same characteristics will prevail in two-dimensional mapping as shown on figure 3. This time we have four adjacent neurons to each neuron, east, west, north and south direction. The differences will be inversely proportional to the topological distance between the neurons. We can think of three dimensional mapping the same way. In practice, one or two-dimensional mapping is usually used. I will discuss the case of two-dimensional mapping in seismic interpretation. The mapping configuration will affect only the definition of the neighborhood function; thus it is very easy to handle the various dimensionality of the maps.

USES OF SELF-ORGANIZING MAPS:

SOM's can be used in seismic interpretation in two principal ways, one as a cluster generator and the other as a classifier. Input data consists of 2-D or 3-D trace segments or their attributes. Each data sample and corresponding attributes representing a point in the N dimensional space is used to form a set of weights for each Neural node according to the network topology. These weights are called the associated memory, because they represent reference attributes of each cluster and they have some organized association to their neighbors. This process takes a large number of iterations, of the order of tens of thousands. Measuring the RMS error reduction after each iteration can monitor convergence. Once the convergence reaches a satisfactory minimum, then the weights are saved for the classification. Since each weight vector represents a reference attribute of a particular cluster, we perform classification by computing the Euclidean distance between each data sample and the weight vector of each node, and select the node with minimum distance. The **SOM** is an unsupervised learning algorithm; it does not give us direct information on the actual physical or lithological classification of the data samples. It merely clusters and classifies the data set based on the set of attributes used. We will need to attach lithological meaning after the classification is completed by the use of nearby well information or based on our experience in the area. If there is a lithological column available in the general vicinity of the seismic data, then we can experiment with different sets of attributes to see which set gives us similar classification boundaries as the lithological column.

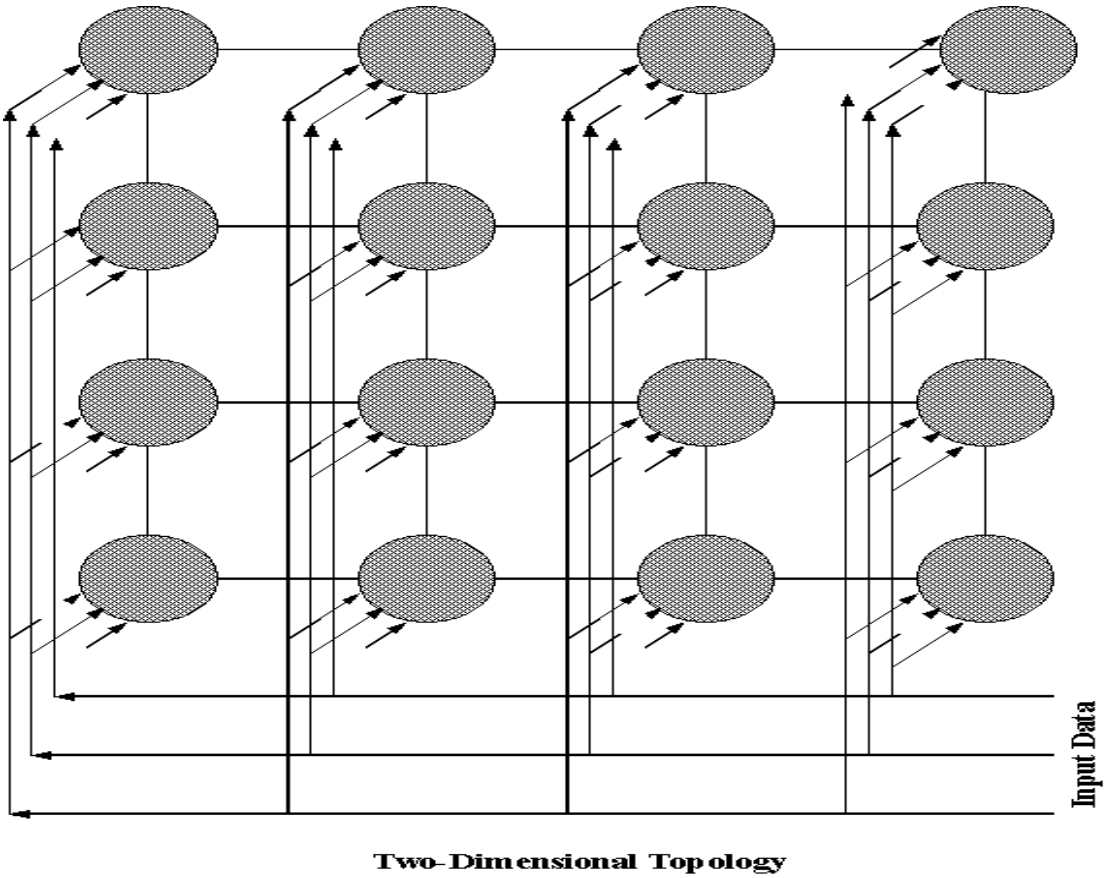

Figure 3. Input to 2-Dimensional Neural Topology

COMPUTATIONAL PROCEDURE:

I will outline the computational procedure step by step including some discussion associated with each step.

1. ***Data Initialization***: Once the composition of the attributes is selected, each attribute set will be normalized (scaled) to RMS=1. This will prevent one attribute from overpowering in clustering and classification. If, however, we consider some attributes more diagnostic than the others, we can upward adjust their RMS amplitudes. This will make this particular attribute's distance be more prominent than the rest, hence minimization will be affected by it.

2. ***Initialization of Neural Weights***: Each neuron will have one weight for each attribute. To begin, we assign small random numbers to each of the weights, as in the case of other Neural Network computations. In the beginning we will also initialize the neighborhood radius. This radius will be large at the beginning to cover a larger number of adjacent nodes. As the iteration proceeds, this radius may be systematically reduced to a unit distance. We will also initialize a set of counters that will indicate the number of consecutive times a Neural node is declared as the winner. In the "*conscience*" algorithm, if a node wins excessively, it will feel bad and allows others to win. This is accomplished by artificially reducing the computed weights, causing other nodes to win.

3. ***Selection of Input data***: Select input data at random from the given set of seismic data traces and their attributes. This will prevent similar neighboring attributes from over influencing the weight-updating scheme. All of the data samples are input at each iteration cycle.

4. ***Computation of Euclidean Distance or Vector Dot Product and determining the winning Node***: The Euclidean distance or Vector dot product is computed between the input data set (vector) and each of the Neural Nodes (as shown on equations 1 and 2). The Node with the minimum Euclidean distance or maximum vector dot product is declared the winner. This determines the node with the most similar set of weights to the input data attributes. Before the winning node is determined, the number of consecutive wins is checked. If the node had exceeded the limit, then the results are reduced proportionally to allow another node to win. The losing node counters are reset to zero, and the winning node counter is incremented by one.

   In some instances one node may continually win, thus overpowering all the other nodes. To prevent this DeSieno (1988) has introduced an algorithm called the "conscience". This algorithm keeps track of the winning neurons and if they exceed some level of number of winning , their Euclidean distances are increased or the dot products are decreased to allow the other neurons a chance to win.

   Let the output of winning **i**'th element given by;

$$y_i = 1 \quad \text{if} \quad \left\| w_i - X \right\|^2 < \left\| w_j - X \right\|^2 \text{ for all j values, except , } i \neq j$$

$$y_i = 0 \quad \text{otherwise.} \qquad \textbf{(3)}$$

A bias is developed for each neuron based on the number of times it has won the competition shown on eq. (3) Let p represent the fraction of time a neuron wins, then we can update the bias factor by;.

$$p_i^{new} = p_i^{old} + b[y_i - p_i^{old}], \qquad \textbf{(4)}$$

where $0 < b << 1.$ We are using $b = 0.0001$.. We compute now, the conscience factor;

$$b_i = C(1/N - p_i), \qquad \textbf{(5)}$$

where C is the bias factor, and N is the number of neurons. Some of the examples on various book use C=10. We have to experiment with this value. Once the conscience factor is determined, new (conscience modified) Euclidean distances will be computed as;

$$z_i = 1 \quad \text{if} \quad \|w_i - X\|^2 - b_i < \|w_j - X\|^2 - b_j \text{ for all j values, except , } i \neq j$$

$$z_i = 0 \quad \text{otherwise.} \qquad \textbf{(6)}$$

This will determine the new winner. We will continue the process by making the weight vector of the winning neuron closer the input data, as described below.

5. **Updating the weights of winning node and its neighbors**: The winning nodes weights will be modified by ;

$$w_{(n+1)}(k,j) = w_n(k,j) + h(n)[x(j) - w_n(k,j)] \qquad \textbf{(7)}$$

where **n** is the number of the iteration, **k** is the winning node and **j** is the j'th attribute. $h(n)$ is the modification weight as a decreasing function of the number of iterations. This is also called the learning rate. All of the nodes within the associative radius will also be modified, inversely proportional to the distance on the map. If we assign one, two or more dimensionality to each node, we can compute the relative distance from the winning node to each other node and can easily compute the modifications. Let the radius ***d(m,k,n)*** be represented as a decreasing function of the iteration number.

6. ***Update neighboring nodes according to their distances to the winning node;***

$$w_{(n+1)}(m,j) = w_n(m,j) + d(m,k,n)h(n)[x(j) - w_n(m,j)] \qquad \textbf{(8)}$$

where ***d(m,k,n)*** is a function of iteration number ***n***, and the distance between winning node ***k*** and its neighbor m. All other nodes outside the associative radius will have no correction.

7. Update associative correction radius;
In the beginning the initial radius may be kept long enough to cover most of the neural nodes. This is reduced as the iteration number increases. ***d(m,k,n)*** can be made any function inversely proportional to the iteration number n, and the distance between the winning node k and the node to be adjusted m.

8. Check convergence rate to continue or to go to finish iteration:
We compute RMS error (square root of the sum of error squares, which is the square of the distance between the input and the winning neural nodes weights divided by the number of inputs) and check if it is below some prescribed level. If it is, then we have developed the desired weights, which can be used for classification. If not, then we go back for another set of iteration (back to step 3).

9. Quality control:
One of the main characteristics of Kohonen's mapping is the determination neurons with topologically associative weight sets. Therefore, a good quality check is to see if the Euclidean distance between neurons computed from their weights do satisfy the original topological design. That is, computed distances between neural nodes should be proportional to the topological distances of the desired map. In other words, final map coordinates should resemble shape of the user specified map. If this condition is not satisfied, some additional iteration may be needed.

10. Save computed weights as classifiers or as reference attributes:
The computed weights of the neural nodes are the memory functions, which are used for classification. They should have smaller differences between adjacent neurons. In our case they represent the Attribute values of

each classification. (For actual values we have to remove the scaling we applied to the attributes during normalizing.)

11. Use computed reference attributes to classify input data set:
In the classification stage for each input, we compute the Euclidean distance to each neural node and classify the input as a member of the class represented by the winning node. This is very similar to the actual iteration, except we no longer adjust the weights.

12. Validity check of the results:
The "conscience" condition tries to develop a uniform or equally balanced clustering. Therefore we should expect that each classification should have about the same number of members and similar variance. During the classification process, the number of members of each class and their statistics are accumulated , which could then be used for a check of validity. These results, as well as other quality control statistics, are output to the user for proper control of the process..

CONCLUSIONS:

Kohonen's Self-Organizing Map method can be used in several different ways for lithologic interpretation. One of the most obvious ways is to use a typical portion of a 2-D or 3-D data set to train the network; then use the reference weights as classifiers for the rest of the data set. If we have a lithological column, we can experiment with a number of different and logical attribute combinations to see which clusters are most similar to the lithological column.

In this elegant method, there is no restriction to the dimension of the input. It could be one, two or many dimensional input vectors, defining various physical attributes or geometrical conditions. This means that, we may be able to use the SOM for visual type classifications. This we will have to experiment with in the near future.

ACKNOWLEDMENTS:

I would like to express my thanks to James Schuelke and John Quirein of Mobil R&D Corporation for their invaluable help and direction. To them and to Graham Jago goes many thanks for gentle editing of the text. The value of the SOM technique in interpretation became more obvious in our conversations.

REFERENCES:

Proceedings of the IEEE, 1990, Special issue on Neural Networks;
1**.** ***Neural Networks, theory and modeling*** (**September** issue)
2.***Neural networks, analysis, techniques and applications*** (**October** issue)
(These issues have extensive literature and background articles)

Bharath, R, and Drosen, J, 1994, "***Neural Network Computing"***, Published b Windcrest/McGraw-Hill, New York (This book contains examples of some primitive functions used in Neural computation. C-code programs are included on a diskette)

Chester, M, 1993, "***Neural networks, A Tutorial"***, Published by PTR Prentice Hall, Englewood Cliffs, New Jersey. (This book is recommended to the beginners of the Neural Networks)

DeSieno, D., 1988. ***"Adding a Conscience to competitive learning."*** *IEEE International Conference on Neural Networks*, Vol. 1, pp. 117-124, San Diego CA.

Haykin, S. 1994, "***Neural Networks, A Comprehensive Foundation"***, Published by Macmillan College Publishing Company, New York. (This book contains one of the most comprehensive discussions of all types of Neural Networks including the Self-Organized Maps).

Kohonen, T., 1988, "***Self-Organizing and Associative Memory"***, 3rd. ed., Published by Springer Verlag, New York.

Pao Y. H., 1989***, "Adaptive Pattern Recognition and Neural Networks"***, Addison-Wesley Publishing Company.