WATER RESOURCES RESEARCH, VOL. 30, NO. 2, PAGES 483–497, FEBRUARY 1994

Characterization of aquifer properties using artificial neural networks: Neural kriging

Donna M. Rizzo and David E. Dougherty

Research Center for Groundwater Remediation Design, Department of Civil and Environmental Engineering
University of Vermont, Burlington

Abstract. A method for pattern completion based on the application of artificial neural networks and possessing many operational objectives of the ordinary kriging approach, neural kriging, is developed. A neural kriging (NK) network is described, implemented in a parallelizing algorithm, and applied to develop maps of discrete spatially distributed fields (e.g., log hydraulic conductivity). NK is, in the case of two discrete field values, similar to indicator kriging. It uses a feed-forward counterpropagation training approach because field observations are available and because fast yet reliable results are obtained. NK is data-driven and requires no estimate of a covariance function. The optimal design of the NK network is found to depend on the number of hidden units in a more complex way than expected. The quality of the estimate of each pixel of the NK maps can be presented as well, as in kriging, to help identify areas in which additional information will be most beneficial. A comparison with a reference field shows that the NK network produces unbiased errors relative to sample bias and reproduces the variogram of a quantized random field with reasonable accuracy. Ordinary kriging (OK) followed by quantization can also perform well; however, estimation errors in the variogram selected for use in OK (in this case the range coefficient in particular) must be carefully examined and treated. The NK method can provide multiple realizations of the estimated field, all of which respect observations; hence conditional simulation is demonstrably possible. The combination of simplicity, interpolation, reasonably accurate prediction statistics, ability to provide conditional simulations, and computational speed suggest that artificial neural networks can be useful tools in geohydrology when applied to specific well-defined problems for which they are well suited, such as aquifer characterization.

1. The Pattern Completion Problem

Patterns, organized distributions of a thing (quantity or quality) of interest, play an important role in groundwater hydrology. Examples include preferential paths of high hydraulic conductivity, surfeit of phreatophytes near shallow water tables or springs, and distribution of contaminant relative to possible sources. Some of these patterns are difficult to discern due to their randomness or the expense of sampling.

A prototypical example, the one we shall study here, is the identification of hydraulic conductivity fields (usually their logarithm is described). The log conductivity Y is spatially random and autocorrelated. Samples of Y values are obtained by in situ tests (such as slug tests), laboratory tests (such as permeameters), or correlation with other spatially distributed fields (such as piezometric head). Given a limited amount of information (which may, nevertheless, be quite extensive), it is desirable to determine a "best estimate" of the Y field. This is a pattern completion problem, as opposed to pattern recognition, which seeks to associate a sampled field with one or more reference "memories" (e.g., to recognize a letter in a handwritten text).

Kriging [*Journel and Huijbregts*, 1978] is an interpolating method of pattern completion that is widely used in the geohydrologic community. Kriging leads to unbiased estimates of the field variable at a given point and the variance of the estimation error at that point. The method can be developed by assuming that Y or one of its increments is weakly stationary (see Matheron's theory of regionalized variables [*Matheron*, 1971]), that the distribution of Y is known (typically multivariate Gaussian), that the correlation structure among variables is perfectly (a priori) known, and that an unbiased estimator is desired. The latter is an attractive property if the samples used to develop and justify the distribution and correlation structure are unbiased. The estimates are optimal if the underlying field is Gaussian. In practice the correlation structure is estimated; variogram estimation errors can be a major source of estimation error in the field variable of interest (Y).

In geohydrologic practice, information becomes available over the course of time, so estimation will not be done once and for all (as suggested above), but rather will evolve. As new information becomes available from field observations, (1) the variogram and its parameters should be reestimated [*Massmann and Freeze*, 1989], (2) the kriging system will grow larger and needs to be resolved based on this new conditioning information, and (3) field estimates (estimates of Y at unobserved points of interest) must be recomputed.

Recently, geostatistical estimation methods that are more nonparametric (distribution-free) have been introduced into the ore estimation literature. Among several methods are

Paper number 93WR02477.
0043-1397/94/93WR-02477$05.00

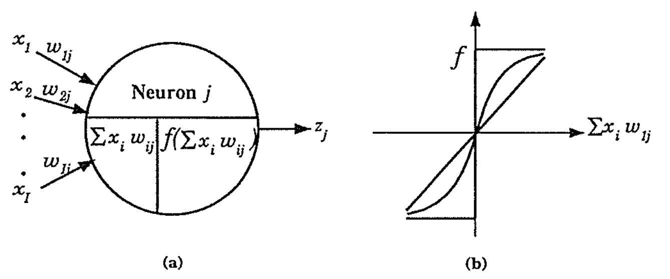

Figure 1. (a) Schematic of a single artificial neuron with inputs, x_i, weights, w_{ij}, and output, Z_j. (b) Several activation functions commonly used in artificial neural networks.

indicator kriging (IK) [*Journel*, 1984] and probability kriging (PK) [*Sullivan*, 1984]. The basic idea of IK is to estimate the amount of ore with grade in excess of a threshold value (which is either in excess of or below the cutoff value); PK looks at distributions as a function of the value of the threshold. Like ordinary kriging, both IK and PK require that the variogram be inferred from the observations before field estimation can begin. These methods are frequently used to discriminate the distribution of discretely valued classes of a property of interest (e.g., regions of high, medium, low, and negligible hydraulic conductivity). A method for conditional simulation based on indicator kriging has been presented by *Gomez-Hernandez and Srivastava* [1990]. A brief and intensive introduction to these and related geostatistical methods can be found in the work by *Journel* [1989].

A number of other methods are in use for characterization into discrete or quantized classes. The most common one is the "Crayola method," in which a professional sits down with a box of colored markers and a map that has observations noted on it and then colors in the map (or cross section) according to judgment, experience, intuition, and more. In spite of many criticisms, this method pervades the field.

We shall explore here a method of quantized pattern completion that imposes no statistical parameters on Y, is extensible as new information becomes available, and is, from a computational point of view, highly parallel. The objective is to develop a method for estimating a spatially distributed field that interpolates ("respects") observations, internally extracts spatial correlation of observations, provides a measure of the sureness of estimation at specific spatial locations, can be readily applied in two- or three-dimensional applications, and can be used as part of conditional simulation. Because of the similarity of our operational objectives to kriging, we refer to our ideas as neural kriging.

The organization of the paper is as follows. Section 2 introduces artificial neural networks. We move to a thorough description of the design and implementation of an artificial neural network (ANN) designed for geohydrological applications in section 3. (A sample program is available without cost from an anonymous ftp site. Details are given in the appendix.) Demonstrative examples are presented in section 4 to explain the internal behavior of the network, introduce the "sureness" of an estimation, establish a relationship of this ANN to some classical statistical methods, and provide an initial small-scale comparison with kriging. This section also provides a clear demonstration that conditional simulation is possible with this ANN. We do not illustrate in this paper the ability of the method to naturally and conveniently assimilate additional information as time advances. The paper ends with some summary remarks.

2. Artificial Neural Networks

An artificial neural network (ANN) is a biologically inspired computational system that (1) comprises individual processing or computational elements with associated memory, (2) is interconnected in some information-passing topology, (3) operates largely in parallel, and (4) has some ability to adapt its functioning to its inputs and outputs [see *Hertz et al.*, 1991]. The belief that an "intelligent" system might be developed from the collective behavior of many of these interconnected processing elements has led to the development of neural net models. First explored in the 1940s [*McCulloch and Pitts*, 1943], ANN research waned in the 1970s as a result of work reported by *Minsky and Papert* [1969] and has been recently revived. Its literature is widely distributed among engineering, computer science/artificial intelligence, biological science, cognitive science, physics, chemistry, and other disciplines. Recently, there have been a spate of papers in the earth sciences literature as well (from a recent electronic search [e.g., *Chiu et al.*, 1990; *Dowla et al.*, 1990; *Dysart and Pulli*, 1990; *Hepner et al.*, 1990; *Pulli and Dysart*, 1990; *Ritter and Hepner*, 1990; *Fitch et al.*, 1991; *Johansson et al.*, 1991; *Zitko*, 1991; *Aziz and Wong*, 1992; *Epping and L'Istelle*, 1992; *Kanellopoulos et al.*, 1992; *Ranjithan et al.*, 1993]).

We do not ascribe any intelligence to the neural network. We simply use the technology of ANNs to form a computational system. However, the heritage of ANNs causes us to use terminology that has come from cognitive studies and artificial intelligence.

Consider the single artificial neuron illustrated in Figure 1a with label j. Its function may be described as follows. Inputs $\mathbf{x} = x_i$, $i \in \{1, 2, \cdots, I\}$ are multiplied by weights $\mathbf{w}_j = w_{ij}$ (where $j = 1$ for this example) and summed in the neuron, forming $z_j = \sum_{i=1}^{I} x_i w_{ij}$. This result is then acted upon by an activation function, yielding the output of the jth neuron $Z_j = f(\sum_{i=1}^{I} x_i w_{ij})$. Typical activation functions are given in Figure 1b. It is important and necessary that the activation function not be both linear and C^1 continuous.

The weights w_{ij} for an artificial neuron are, at least to a certain degree, plastic. That is, their value may change based upon the inputs and outputs they produce. This adaptation to external and internal stimuli is an important characteristic of both real and artificial neurons. The self-adjustment of these weights is also called learning, training, or conditioning.

Such individual artificial neurons may be interconnected in some information-passing topology to form an ANN. An example of the topology of the ANN used in this work is shown in Figure 2. This network is an example of a feed-forward network that will be trained by counterpropagation. The details of its architecture and operation will be described in the next section.

Of the ANNs currently available, the counterpropagation network was chosen for the task at hand (developing maps of discrete spatially distributed fields) because of its ability to "learn" a mathematical mapping given nothing more than examples of the mapping action. ANNs that implement an approximation to a function or mapping are called "mapping neural networks." Other ANNs are also capable of learning the mapping at hand (e.g., back propagation). However, very few ANNs (e.g., counterpropagation [*Hecht-Nielsen*, 1987a, b] and probabilistic neural networks [*Specht*, 1990a, b]) are purely data driven and nonparametric and can achieve near-Bayesian performance in high-dimensional feature spaces. It is important to note that the lack of parameters does not imply a lack of structure; the architecture and the training algorithm chosen for the ANN impose structure on the observations that may be rigid or flexible.

3. Neural Kriging

In this section we describe the ANN designed for characterizing log conductivity, Y, into discrete classes. (Other ANNs can be used for floating point values of Y, but we do not address them here.) It may be applied to any other scalar field, including the components of a vector field. We restrict our investigations here to pattern completion for a scalar field Y using only observations of Y at known points $\mathbf{x}^m$ where $m = 1, 2, \cdots, M$ and M is the total number of observation points. For the purposes of this paper, observations are assumed to be error-free; independent errors can be accommodated by training to a root-mean-square (RMS) error consistent with the measurement error at a given point. No coestimation is considered here; this will be discussed in a sequel paper on the incorporation of "soft" data.

3.1. Overview

In its current form, neural kriging is performed using the three-layer network, as shown in Figure 2. The input to the ANN is a vector, x, of real-valued numbers representing the coordinates of any point of interest, whether a measurement has been recorded there or not. This information is transformed by the "hidden" Kohonen layer and the output Grossberg layer. The output of this ANN is a vector, Y, consisting of K logical values that constitute a code for identifying the discrete hydraulic conductivity class to which the spatial point belongs. For the given network, the input layer contains I units corresponding to the I components of the input training vector $\mathbf{x} = (x_1, x_2, \cdots, x_I)$. These I inputs may be the N spatial coordinates or a normalized vector of dimension $I = N + 1$ computed from the N spatial coordinates. A number of normalization procedures exist. We chose a method which preserves both the relative values of the vector components and the total length of the normalized vector. This is described in pseudocode 1 (Figure 3).

Output vectors Y represent one of three (= K) discrete values of log conductivity (e.g., high medium, low). We encode $\mathbf{Y} = \{1, 0, 0\}, \{0, 1, 0\}, \{0, 0, 1\}$ for high, medium, and low values of Y, respectively. Initially random weights w_{ij} and v_{jk} interconnect the input, hidden, and output layers. The dual subscript, for example w_{ij}, indicates a weight from unit i in the input layer to unit j in the hidden layer. If desired, one may also obtain a measure of the "sureness" of the prediction; this measure is described later in this paper.

The use of the ANN consists of two phases, a training phase and an interpolation phase. The network must first be trained on some observations or samples. Say we have M points $\mathbf{x}^m$ at which we measure and encode $\mathbf{Y}^m$. The vectors $\{\mathbf{x}^m, \mathbf{Y}^m\}$ are called training vectors, and the set of all training vectors $m = 1, \cdots, M$ is called the training set. Iteratively presenting the training vectors, a training algorithm adjusts the sets of weights

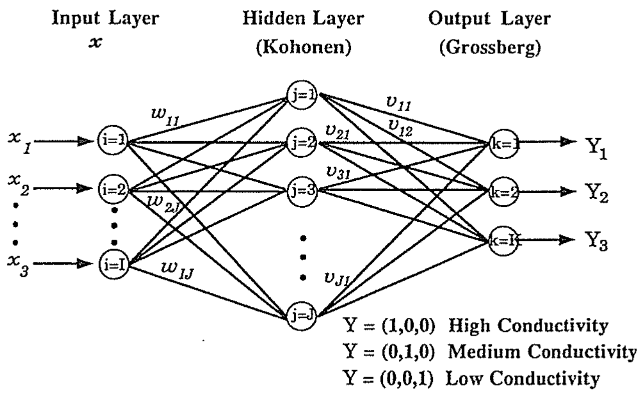

Figure 2. Example schematic of the neural kriging network for $K = 3$. For normalized inputs I equals one plus the number of spatial coordinates of interest.

associated with each interconnection (w and v of Figure 2) such that $\mathbf{x}^m$ is mapped to $\mathbf{Y}^m$ to within a specified degree of accuracy. When all training vectors are satisfactorily mapped, weights are fixed and the training is complete. This procedure is detailed below.

Once training is complete, the fixed weights may be used in the interpolation (or mapping or operational) phase. Y is estimated at a point of interest x by simply using x as input to the network and observing the output from the ANN.

In Figure 2, and in all of the examples presented in this paper, the number of spatial dimensions is $N = 2$ and the number of output log conductivity classes is $K = 3$. We have also extended the method to three-dimensional spatial coordinates with ease. In fact, we have used six-dimensional input data in a network like this to incorporate "soft data"; this work will be separately reported. In principle, any number of log conductivity classes may be predicted, although clearly no more than appear in the observation data (training set).

3.2. Interpolation or Operational Phase

Although a training phase must precede the interpolation phase, for purposes of simplicity, we have chosen to first present the interpolation phase. During interpolation the input patterns (spatial coordinates) are applied, and the trained network classifies that pattern into a particular (log conductivity) group. An algorithm to accomplish this is described in pseudocode 1 (Figure 3).

The final estimated output from the Grossberg layer, $\mathbf{Y}^p = \{Y_1^p, Y_2^p, \cdots, Y^p\}$, is a vector containing the value one in the entry for which $\mathbf{Y}^p$ is maximum and the value zero in all other entries. The superscript p indicates Y at a prediction point as opposed to a measurement point, m.

3.3. Training Phase

We use the counterpropagation training algorithm developed by *Hecht-Nielsen* [1987a, b, 1988], which is a combination of two well-known algorithms: the self-organizing map of *Kohonen*, [1988] and the Grossberg outstar [*Grossberg*, 1969, 1982]. The activity of the Kohonen units during training is the key to the success of the algorithm. Kohonen training has the useful and interesting ability to extract the statistical properties of the input data set [*Hecht-Nielsen*, 1987a, b].

Pseudocode 1: Interpolation algorithm (with normalization)

```
$\mathbf{xnorm}^p = norm(\mathbf{x}^p)$                /* Normalize input vectors (see below) */
For $p = 1 : P$                                         /* All prediction points */
    For $j = 1 : J$                                     /* All nodes in the Kohonen layer */
        $z_j = 1 - \sum_{i=1}^{N} xnorm_i^p wnorm_{ij}$  /* Calculate normalized distance between input
                                                            vectors and fixed Kohonen weights */
    end for $j$
    $c = index(min(\mathbf{z}))$                        /* Set index of minimum component of z to c */
    $Z_j = \begin{cases} 1.0 & \text{for } j = c \\ 0.0 & \text{for } j \neq c \end{cases}$
                                                        /* Pass Kohonen layer output, z, through a "winner-
                                                            take-all" activation function */
    For $k = 1 : K$                                     /* All nodes in the output layer */
        $y_k^p = 1 - \sum_{j=1}^{L} Z_j v_{jk}$          /* Calculate normalized distance between Kohonen
                                                            layer output and fixed Grossberg weights */
    end for $k$
    $c = index(min(\mathbf{y}))$                        /* Set index of minimum component of y to c */
    $Y_k^p = \begin{cases} 1.0 & \text{for } k = c \\ 0.0 & \text{for } k \neq c \end{cases}$
                                                        /* Pass estimated Grossberg layer output, y^p,
                                                            though a "winner-take-all" activation function */
end for $p$


function $norm(\mathbf{x}^p)$                           /* Function which normalizes vectors such that relative
                                                            length is preserved (Two dimensional vectors are
                                                            mapped to a unit sphere) */
For $p = 1 : P$                                         /* All prediction points */
    $\|\mathbf{x}^p\| = \sqrt{\sum_{i=1}^{N} x_i^2}$     /* Calculate length of each vector x^p */
end for $p$
$a = max\|x^p\| + 0.01$                                 /* Choose a value slightly greater than the length of
                                                            of the largest vector */
For $p = 1 : P$                                         /* All prediction points */
    $b = \sqrt{a^2 - \|\mathbf{x}^p\|^2}$
    $\mathbf{x}^{p'} = (x_1, x_2, \cdots x_I, b)$         /* Catenate b to each of the vectors */
    $norm(\mathbf{x}^p) = \mathbf{x}^{p'}/\textcircled{b}$  /* Normalize */
end for $p$                         $\downarrow$
                                   a
```

Figure 3. Pseudocode 1.

During training, the input layer receives the set of M input training vectors $\mathbf{x}^m$ (observation points for samples), while the output layer simultaneously receives the corresponding set of M target training vectors $\mathbf{Y}^m$ (see Figure 2). The input vectors and weight vectors are normalized in a manner similar to that detailed in the function norm of pseudocode 1. The normalized vectors are then passed to the middle layer known as the Kohonen layer or hidden layer. The Kohonen layer possesses J processing units. At each of these processing units the input vector, $\mathbf{x}^m$, is dotted with the weight vector, $\mathbf{w}_j$, associated with each Kohonen unit j, producing an output vector with elements $z_j = \sum_{i=1}^{I} x_i^m w_{ij}$ ($j = 1, 2, \cdots, J$). In general, these sums may then be passed through a nonlinear activation function as described earlier. However, in the counterpropagation algorithm, competition occurs in the hidden layer, and the output of the Kohonen units with the maximum dot product is set to one, while the output of all other hidden layer nodes is set to zero. These outputs are passed to the K processing units of the output layer, also known as the Grossberg layer. The Grossberg layer performs a dot product between the Kohonen layer output vector, $\mathbf{Z}$, and a weight vector, $\mathbf{v}_k$, associated with each Grossberg unit k, to produce an approximation vector, $\mathbf{y}^m$, with elements $y_k^m = \sum_{j=1}^{J} Z_j v_{jk}$ ($k = 1, 2, \cdots, K$) to the components of the original target vector $\mathbf{Y}^m = \{Y_1^m, Y_2^m, \cdots, Y_K^m\}$.

During training the counterpropagation algorithm adjusts the Kohonen and Grossberg weights, w_{ij} and v_{jk}, such that over each iteration of the training set, $(\mathbf{x}^m, \mathbf{Y}^m)$, the outputs from the Grossberg layer $\mathbf{y}^m = \{y_1^m, y_2^m, \cdots, y_K^m\}$ more closely approximate the components of the original target training vector $\mathbf{Y}^m = \{Y_1^m, Y_2^m, \cdots, Y_K^m\}$. Once the approximation outputs match the original target outputs to within some specified root-mean-square (RMS) error value, training is complete; the weights are fixed, and the second phase (the interpolation phase) may begin. The equations and other details of the counterpropagation training are presented in pseudocode 2 (Figure 4). For further details see *Hecht-Nielsen* [1987a, b].

This basic training loop is repeated until the output from the network, $\mathbf{y}^m$, matches all available measurements of $\mathbf{Y}^m$. Initially, the weights of the Kohonen and Grossberg layers are set to random values between zero and one (ideally, the components of each $\mathbf{v}_k$ should sum to one), and the network does not perform well. As training proceeds, the weights are adjusted and performance improves. When the error rate ε is very low, training is complete. Performance is measured by RMS error values

$$\varepsilon = \left(\frac{\sum_{m=1}^{M} \sum_{k=1}^{K} (Y_k^m - y_k^m)^2}{MK} \right)^{1/2},$$

where M is the number of measurements in the training set and K is the number of units (or components) in the output layer (also the number of discrete conductivity classes).

For the examples that follow, of the order of 10^2 iterations

Pseudocode 2: Training algorithm (with normalization)

```
$\mathbf{xnorm}^m = norm(\mathbf{x}^m)$                                  /* Normalize input vectors */
For $j = 1 : J$                                              /* All nodes in the Kohonen layer */
    $\mathbf{wnorm}_j = norm(\mathbf{w}_j)$                             /* Normalize Kohonen weight vectors */
end for $j$

Do until $\epsilon \leq$ tolerance                              /* Train until data is interpolated */
    For $m = 1 : M$                                          /* All observation points */
        For $j = 1 : J$                                      /* All nodes in the Kohonen layer */
            $z_j = 1 - \sum_{i=1}^{N} xnorm_i^m wnorm_{ij}$     /* Calculate normalized distance between input
        end for $j$                                             vectors and normalized Kohonen weights */
        $c = index(min(\mathbf{z}))$                          /* Set index of minimum component of $\mathbf{z}$ to $c$ */
        $Z_j = \begin{cases} 1.0 & \text{for } j = c \\ 0.0 & \text{for } j \neq c \end{cases}$   /* Pass Kohonen layer output, $\mathbf{z}$, through a
                                                               "winner-take-all" activation function */
        $\alpha$ = constant between 1 and 0                  /* Adjust weights connected to "winning"
        For $i = 1 : I$                                         Kohonen unit */
            $wnorm_{ij} = \begin{cases} wnorm_{ij} + \alpha(xnorm_i^m - wnorm_{ij}) & \text{for } j = c \\ wnorm_{ij} & \text{for } j \neq c \end{cases}$   /* Equation (1) */
        end for $i$
        $\mathbf{wnorm}_j = norm(\mathbf{wnorm}_j)$                  /* Renormalize adjusted weights */
        For $k = 1 : K$                                      /* All nodes in the output layer */
            $y_k^m = \sum_{j=1}^{L} Z_j v_{jk}$                    /* Calculate dot product between Kohonen
        end for $k$                                             layer output and Grossberg weights */
        $\beta = 0.1$                                          /* Learning constant */
        For $k = 1 : K$                                      /* All nodes in the output layer */
            $v_{jk} = \begin{cases} v_{jk} + \beta(Y_k^m - y_k^m) & \text{for } j = c \\ v_{jk} & \text{for } j \neq c \end{cases}$   /* Adjust Grossberg weights that connect
                                                               to the "winning" Kohonen unit*/
        end for $k$
    end for $m$
end do
```

Figure 4. Pseudocode 2.

were needed to learn the training set to a specified RMS error of 10^{-6}, which ensures the observations are interpolated. This training process may be thought of, in analogy, as iteratively constructing the variogram and weights in the ordinary kriging method. The rate of convergence of the training process to the correct solution can be assessed during training. Its prediction and enhancement are major topics of research interest, because the rate of convergence affects the size of the problem that may be computationally tackled.

4. Applications

In this section we present examples that are intended to explain the internal behavior of the network, establish a relationship of this ANN to some classical statistical methods, introduce the "sureness" of an estimation, and provide an initial, small-scale comparison with kriging. This section also provides a clear demonstration that conditional simulation is possible with this ANN.

4.1. Interpretation of Kohonen Weights

4.1.1. Kohonen layer weights. The activity (values) of the Kohonen and Grossberg weights during training is the key to the success of the algorithm. Choosing the optimal number of hidden neurons (i.e., Kohonen weights) and initial values of the weights is an area of much research. The following subsections will show that the number of hidden units (i.e., number of Kohonen and Grossberg weights) required can be significantly affected by the initial estimates of the weight vectors.

Two keys to understanding the Kohonen weights $\mathbf{w}_j$ are the facts that the dimension of every vector $\mathbf{w}_j$ is the same as $\mathbf{x}$ and that the distance between $\mathbf{x}$ and $\mathbf{w}_j$ (see computation for z_j) is used to predict $\mathbf{Y}$. Therefore we may interpret the vector $\mathbf{w}_j$ as a location in the same space as $\mathbf{x}$.

Consider the following example. An artificial distribution of Y was developed and quantized into $\mathbf{Y}$ values on a 20×20 mesh of points. This reference field, comprising three log conductivity classes, is shown in Figure 5. Twenty samples (observations) were drawn from the field in a uniformly random pattern and used to train a neural kriging network; these observation points are indicated by the letter X in the figure.

We first consider a network with 26 hidden units (nodes in the Kohonen layer) and three classes of $\mathbf{Y}$ to be identified (nodes in the Grossberg layer). The weights w and v were randomly initialized. The network was trained to the acceptable RMS error of 10^{-6} after 132 iterations. After training, the estimated field based on the 20 observations, $\mathbf{Y}^{20}$, shown in Figure 6a was determined.

The addition of 10 observations and retraining of the network ab ignorans (i.e., from random initial weights) led to the map of $\mathbf{Y}^{30}$ given in Figure 6b. Clearly, the neural kriging network adjusts to additional information and is able to respect spatial correlation among observations. Yet, these results are obtained in a completely distribution-free setting using a highly parallel, nonstandard computation.

No improvements in the output map are obtained if more than 35 samples are used together with ab ignorans training. The lack of improvement in estimation, in spite of increased information, is a result of the finite capacity of the 26-neuron network for assimilating information. The estimation map

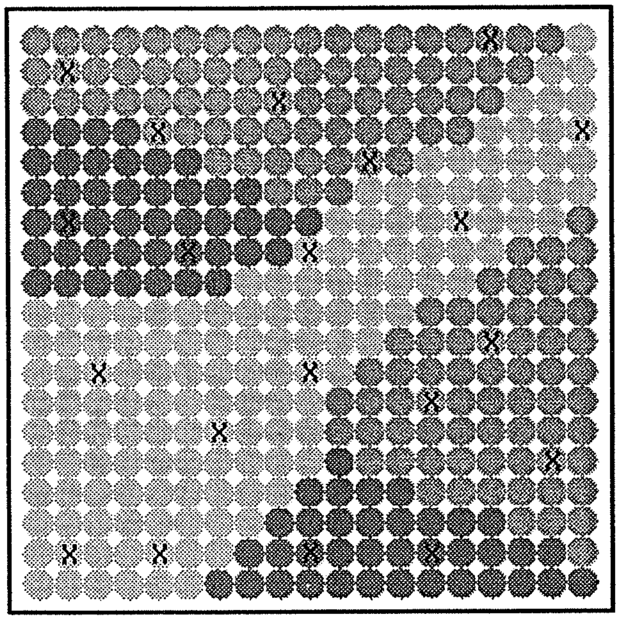
Figure 5. Reference field for the initial examples. Twenty observation points, obtained by uniformly random sampling, are indicated by the letter X.

could be improved provided the number of hidden nodes were increased.

Consider the iterative training process of this normalized ANN algorithm. Because the dot product in the Kohonen layer $z_j = \sum_{i=1}^{I} x_i w_{ij}$ is a measure of the similarity between the input vector and the Kohonen weight vector associated with the jth hidden node, training this network consists of selecting the unit in the Kohonen layer whose weight vector is closest to the input vector, and then adjusting it using equation (1) of pseudocode 2 to be a bit closer. The amount of adjustment is proportional to the difference between the weight and the input vector to which it connects by an amount α. The learning rate, α, usually starts out at approximately 0.7 [*Hecht-Nielsen*, 1988] and may gradually be reduced during training. However, for the examples in Figures 6 and 7 a constant value of $\alpha = 0.7$ was used. We now invoke the idea that $\mathbf{w}_j$ is a spatial vector like $\mathbf{x}$. If the initial density of weight vectors is too low in the vicinity of the training vectors, the network may not be able to assign a weight vector to each input vector category (i.e., separate the input vectors into different $\mathbf{Y}$ categories). That is, more hidden nodes can assimilate more information.

Conversely, if the density of weight vectors is too high near a group of input vectors that should be lumped together into the same category, each input vector may activate a different Kohonen neuron. For example, consider the same distribution of $\mathbf{Y}$ as indicated in Figure 5 and ab ignorans training but with new observation points. The result and location of the new observation points is given in Figure 7a. This clustered sampling is more representative of geohydrological practice than uniform sampling. A minimum of 94 hidden units are needed to learn the training set. The result shown in Figure 7a interpolates the data exactly, although the result is, in other respects, not similar to the reference field of Figure 5. This excessively large number of hidden nodes is a result of the random initial distribution of weight vectors, $\mathbf{w}_j$. A high density of weights is needed near the center of the field, where the observations are located, in order to separate the input vectors into different categories. However, because the initial values of the weights are randomly generated, the density of weight vectors is too high in the remainder of the field (where weight vectors could be lumped together and activate a single Kohonen neuron).

Thus the most desirable solution is to initially distribute the weight vectors according to the density of input vectors that must be separated. Ideally, a single weight vector would be associated with a cluster of observations that correspond to a single output class. Multiple weight vectors would be placed in the vicinity of a large number of input vectors that need to be separately categorized. Figure 7b shows the results of distributing weight vectors according to the density of the input vectors, with 20 observations and using 18 hidden units. The problem of weight distribution can seriously affect the accuracy of the result.

4.1.2. Conditional simulation. Examining the activity of the Kohonen weights before and after training using the two-dimensional maps of Figure 8 shows that conditional simulation is easily accomplished using the network. Input vectors $\mathbf{x}$ are arranged on the two-dimensional area as shown in Figure 8a. Before training, 13 Kohonen weight vectors are randomly placed on the two-dimensional area as indicated by the numbers of Figure 8a. The position of the Kohonen weight vectors after training and the corresponding interpolated map are shown in Figure 8b.

If different sets of initial random weights were selected and training then employed, a series of different maps may actually be produced by the network, all with the same mean value for $\mathbf{Y}$ and all interpolating the observations. The initial distribution of $\mathbf{Y}$ with a different set of initial weights is shown in Figure 8c. The associated interpolated map and location of weights after training is shown in Figure 8d. The interpolated maps of Figures 8b and 8d indicate that conditional simulation is a natural product of the counterpropagation network using random initial weights.

4.1.3. Relationship between Kohonen layer weights and the nearest-neighbor classifier. Over the last two decades, numerous pattern classification strategies in the form of neural networks have been developed which attempt to learn the structure of a classification problem from a finite training set. The back propagation algorithm is the most popular recent example. The underlying hope is that the classifier's performance can be made acceptable with a sufficiently large training set. Like other pattern classifiers an artificial neural network that uses counterpropagation can be viewed as a mapping from an N-dimensional feature space to the discrete set. We have recently compared the ANN developed in this paper to the multivariate statistical technique, discriminant analysis. The results, although favorable, will not be presented here. We will, however, discuss the relationship of the Kohonen layer to what is perhaps the simplest learning algorithm of this class, the nearest-neighbor classifier.

Consider the following highly idealized example. Let $\mathbf{Y}^m$ denote the output states of two discrete log conductivity values (high and low, respectively), and let the input vector, $\mathbf{x}^m = \{x_1, x_2\}$, represent the spatial coordinates of observation points m in two dimensions. Thus the observation points or training vectors are $\{(\mathbf{x}^1, \mathbf{Y}^1), (\mathbf{x}^2, \mathbf{Y}^2), \cdots, (\mathbf{x}^M, \mathbf{Y}^M)\}$ where $\mathbf{Y}^m \in \{(1, 0), (0, 1)\}$. The objective of the nearest-neighbor rule is to assign each input vector, $\mathbf{x}$, into

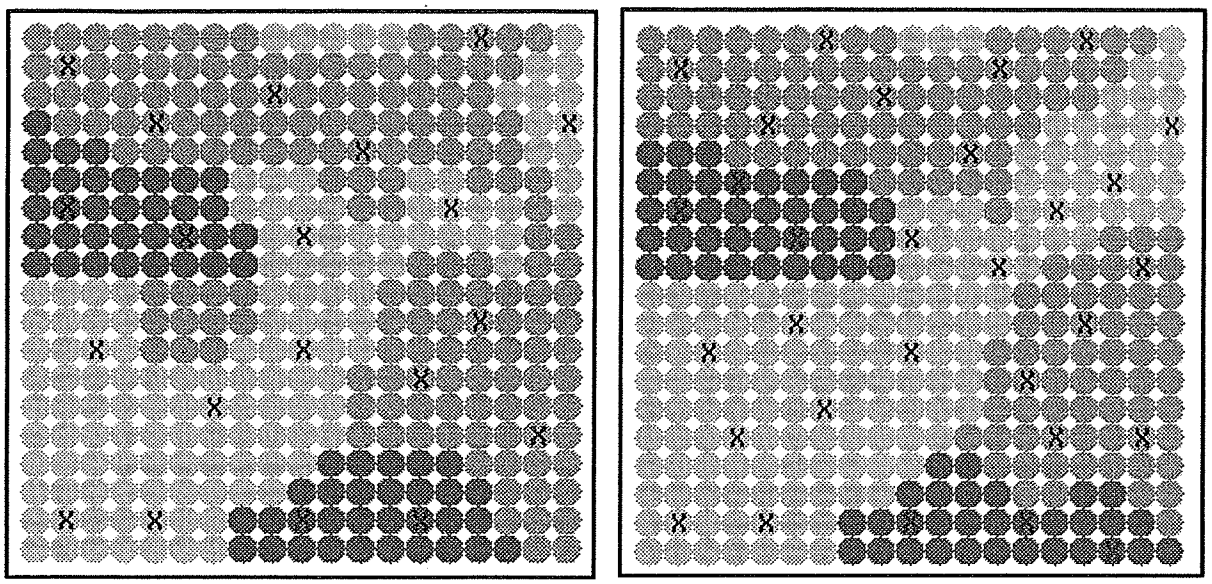

Figure 6. Estimated Y field from NK using (a) 20 observations and (b) 30 observations that are uniformly random in location, 26 hidden layer units, and random initial weights.

one of the two output classes, {(1, 0), (0, 1)}, based on the closest point, $\mathbf{x}^m$, in the training set.

In the special case where (1) the Kohonen weights of the counterpropagation network are initially chosen such that they are equal to the observation points and (2) the Grossberg weights are chosen such that the kth component of $\mathbf{Y}_k$ has a value of one corresponding to the class to which the associated input vector $\mathbf{x}$ belongs and a value of zero in the remaining $K - 1$ components, no iterations are necessary to train the network. For this special case the network of fixed weights may then be used to map results which are identical to those of the nearest-neighbor classifier. An example of the nearest-neighbor decision regions for the two-dimensional classification problem given a training set consisting of nine observations using the nearest-neighbor classifier and the neural network is shown in Figure 9. The difference in the two maps of Figure 9 is the result of using the dot product in the Kohonen layer as a measure of similarity between the normalized input vectors and Kohonen weight vectors, rather than the distance between two points on the two-dimensional (unnormalized) spatial plane.

Counterpropagation networks can be built without the

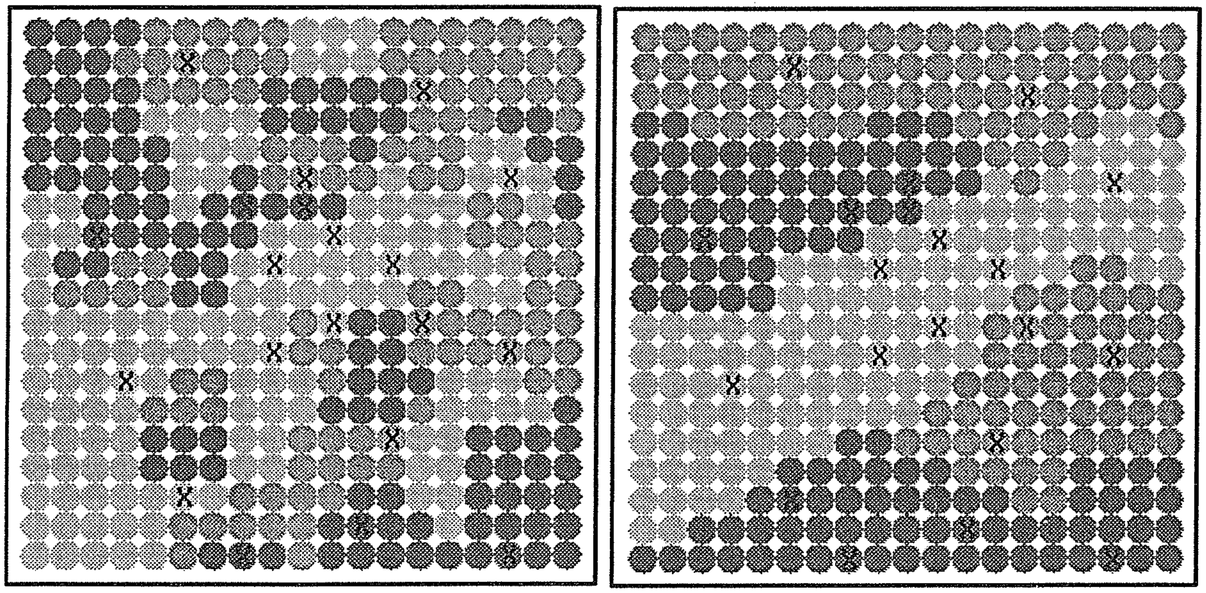

Figure 7. (a) Estimated Y field from NK using 20 observations that are clustered in location, 94 hidden layer units, and random initial weights. (b) Estimated Y field from NK using 20 observations that are clustered in location, 18 hidden layer units, and nonrandom initial weights.

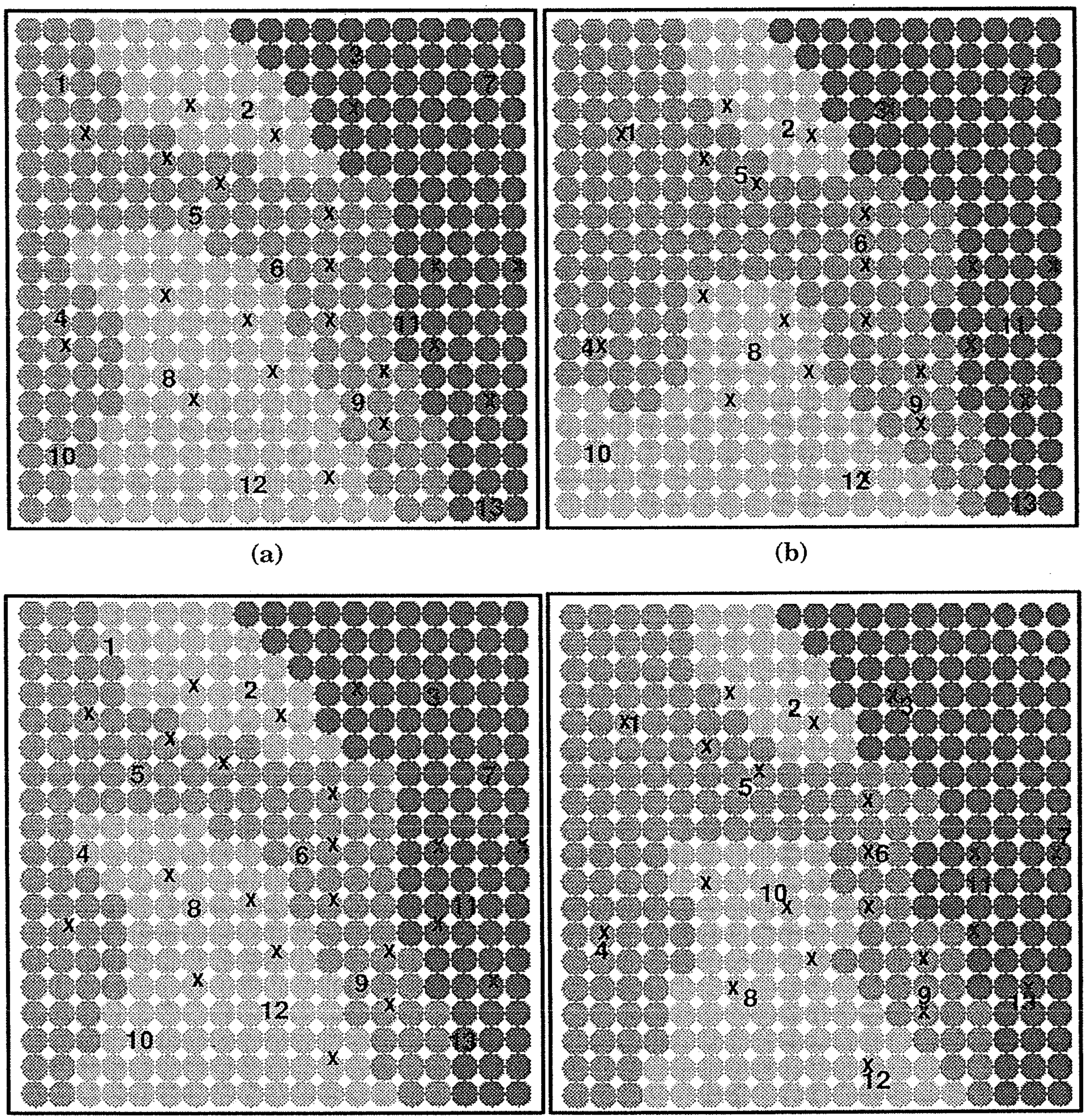

Figure 8. Map of observations showing (a and c) initially random distributed weights, and (b and d) corresponding predicted maps with position of weights after training.

restriction that the input vectors be normalized [*Hecht-Nielsen*, 1988]. Counterpropagation without normalization uses a competitive rule that is based on choosing a "winning" hidden node c such that

$$c = \text{index}\ (\min_j \|\mathbf{x}^p - \mathbf{w}_j\|),$$

where $\mathbf{x}^p$ is the input vector to be classified and $\|\ \|$ is the Euclidean norm. If this version of the counterpropagation network were used, the results of Figure 9b would be identical to those of the nearest-neighbor classifier of Figure 9a. This follows since we have assumed for this special case that the Kohonen weights equal the observation points. As a result the competitive rule shown above reduces to the nearest-neighbor classifier:

$$\|\mathbf{x}^p - \mathbf{w}_c\| = \min_j \|\mathbf{x}^p - \mathbf{x}^m\|.$$

That is, each input vector $\mathbf{x}^p$ is classified based on the $\mathbf{Y}$ class of the closest point, $\mathbf{x}^m$, in the training set. *Cover and Hart* [1967] show that for the two-class problem in the infinite sample limit the probability of error of a nearest-neighbor classifier is bounded by

$$P_B \le P_\infty(\text{error}) \le 2P_B(1 - P_B),$$

where P_B is the probability of error of a Bayes classifier.

The advantage of the counterpropagation algorithm is not its ability to emulate the nearest-neighbor classifier under the very special case of preprocessing the weights to some fixed

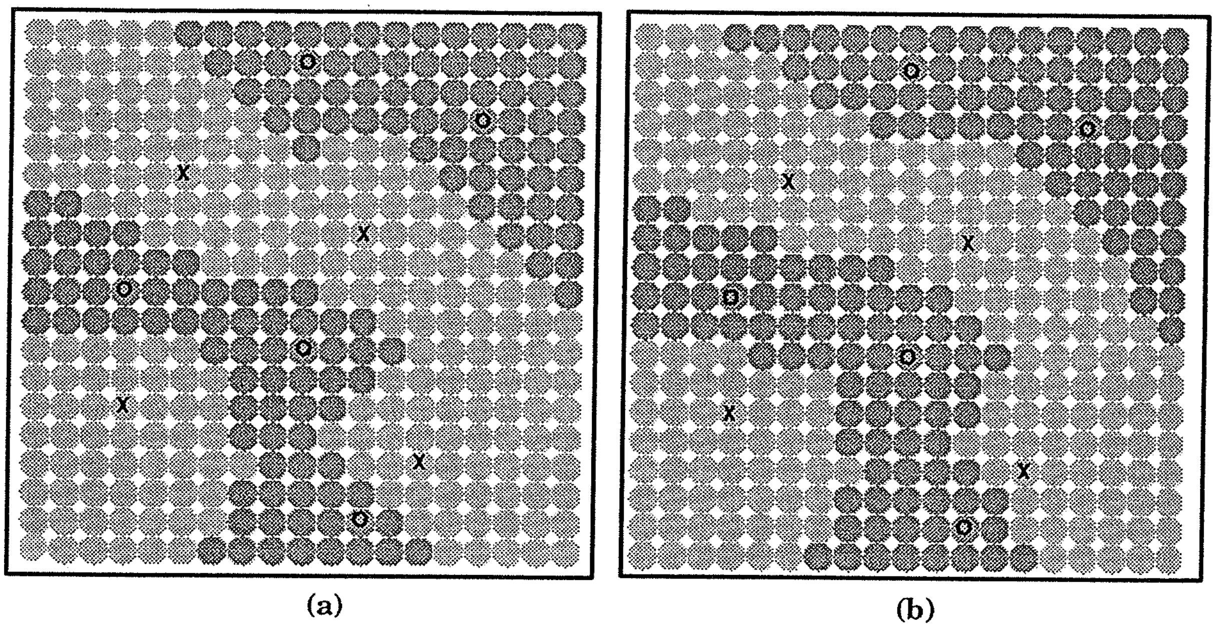

Figure 9. The decision regions produced with a nine-element training set by (a) the nearest-neighbor classifier and (b) the neural network. Observation points indicated by circles represent $\mathbf{Y}^m = (0, 1)$, while crosses correspond to $\mathbf{Y}^m = (1, 0)$.

positions. The beauty and strength of the ANN using the counterpropagation training algorithm are that it can, in certain cases, analyze the statistical structure of data automatically. It is the combination of the Kohonen layer acting as a nearest-neighbor classifier and the Grossberg layer's ability to approximate a Bayes classifier (discussed below) which enables the counterpropagation algorithm to statistically analyze the structure of the data.

4.2. Interpretation of Grossberg Weights

4.2.1. Bayes classifier. The objective of a pattern classifier is to assign the correct class d of the possible output classes to a given input vector $\mathbf{x}$. The Grossberg layer acts as a classifier where the output classes are the K possible log conductivity values.

Consider the case where the number of hidden units equals the number of observations, M, and the Kohonen weight vectors $\mathbf{w}_j$ equal the input vectors for these observations, $\mathbf{x}^m$. For each observation the Grossberg weights associated with each hidden unit $\mathbf{v}_j$ are set equal to (1, 0, 0), (0, 1, 0), or (0, 0, 1) depending on the observed $\mathbf{Y}$ for the case of $K = 3$ output classes of log conductivity. For cases when the Kohonen weights do not equal the input vectors of the observations, the Kohonen layer acts as a "fuzzy" nearest-neighbor classifier or "nearest-mean classifier"; and a single hidden node, c, may fire for more than one observation vector $\mathbf{x}_m$. Because each of these input vectors may have different observed values of Y, the training algorithm presented earlier causes the Grossberg vector associated with hidden unit c to converge to the observed discrete probability density function for the K log conductivity classes associated with Kohonen node c. The components of this vector, y_k, are the probabilities that log conductivity class k occurs given that Kohonen node c has fired. (Probabilities $P(\mathrm{Y}_k|\mathbf{w}_c)$ depend on $\mathbf{w}_c$ and are known as posterior probabilities.)

The winner-take-all algorithm in the Grossberg layer is used to select the class, d, identified with the raw Grossberg vector $\mathbf{y}^m$. Since class d is chosen based on

$$P(\mathrm{Y}_d|\mathbf{w}_c) > P(\mathrm{Y}_k|\mathbf{w}_c), \qquad \forall\, k \neq d,$$

the Grossberg layer acts as a Bayes classifier [*Hecht-Nielsen*, 1987a].

4.2.2. Relationship between Grossberg layer weights and the Bayes classifier. Once again, consider the normalized algorithm. During the iterative training process, the Kohonen weight vectors, $\mathbf{w}_j$, arrange themselves on a unit sphere such that they are approximately equiprobable in a nearest-neighbor sense with respect to input vectors, $\mathbf{x}^m$, on the same unit sphere in accordance with the probability density function.

Just as the weights of the Kohonen layer are trained to average values of the inputs, the weights of the Grossberg layer will converge to the average values of the selected output. During training the weights are adjusted such that they attempt to equal the target outputs, provided the allowed RMS error is low enough. The equation which adjusts the weights of the Grossberg layer causes the weight v_{jk} associated with the single nonzero input to become approximately equal to the average of the kth component of the $\mathbf{y}^m$ vectors that are encountered when this particular Kohonen unit wins the competition in the hidden layer.

Like the Bayes classifier discussed above (see *Duda and Hart* [1973] for details), the output of the Grossberg layer for each class tends to be proportional to the local probability density function value for that class. The reasons these outputs approximate these probability density function values are that the output of the winning unit of the Kohonen layer is 1 and the outputs of the Grossberg layer units tend to sum to 1. Each of the Grossberg weights become approximately equal to the average fraction of the time the points in the neighborhood of a particular Kohonen unit belong to a particular Grossberg unit's class. Thus the outputs of the Grossberg units are approxi-

Pseudocode 3: Sureness algorithm. Sureness is computed for each point of interest.

```
$hmax = \ln K$                           /* Find maximum possible entropy */
$h = \sum_{k=1}^{K}(-y_k \ln y_k)$       /* Compute entropy of $y$ */
$S = 1 - h/hmax$                         /* Compute measure of sureness $S$ */
```

Figure 10. Pseudocode 3.

mately equal to the probability density function values for their classes [*Hecht-Nielsen*, 1988].

It is critical to notice that the normalized counterpropagation algorithm treats local distance differences on an isotropic basis. Hence it will do a poor job of correctly representing the local probability density function values unless changes in coordinate values in one dimension are roughly as "important" as changes in all other dimensions. This means that (as with the nearest-neighbor technique) it is essential that the feature coordinates be scaled correctly [*Hecht-Nielsen*, 1988].

4.2.3. Sureness. The Grossberg weights for each hidden node will converge during training to the relative frequency distribution of the **Y** classes of all observations that cause that particular hidden node to "fire." After training, the winner-take-all rule causes a Bayesian classification based on the Grossberg weight vector $\mathbf{v}_c$. The probability of classification error is zero if the elements of $\mathbf{v}_c$ are 1 for the winning class and 0 for all other classes. That is, the more similar $\mathbf{v}_c$ is to the ideal binary vector, the more sure we are of having made an accurate classification.

For example, Grossberg vectors for points classified as high conductivity are ideally represented as (1,0,0). Two different Grossberg vectors, say $\mathbf{v}_1 = (0.9, 0.06, 0.04)$ and $\mathbf{v}_2 = (0.4, 0.3, 0.3)$, will both be classified as high conductivity after passing through the winner-take-all algorithm. Clearly, however, we are much more sure of this classification for $\mathbf{v}_1$ than for $\mathbf{v}_2$. Conversely, $\mathbf{v}_2$ represents little more information than an unbiased, uninformed prior estimate of (1/3, 1/3, 1/3).

We introduce a measure of "sureness," S, to quantify these ideas. Values of S range between zero and one. A value of one indicates perfect agreement between the Grossberg vector and one of the ideal patterns (1,0,0), (0,1,0), or (0,0,1), while a value close to zero indicates poor agreement. The sureness is assessed by a scaled entropy measure, computed using the algorithm in pseudocode 3 (Figure 10).

A typical result is shown in Figure 11, which shows graphically the distribution of the sureness associated with the first 20-observation log conductivity mapping example of Figure 5. The large circles represent the most decisive points and the smaller circles indicate less "sureness" in the estimate. The results show greatest unsureness in regions without measurements (which is intuitively comforting).

We are currently attempting to develop a theoretical understanding of the metric and its limitations. This quantity may be used, like the variance of estimation error in ordinary kriging, to indicate where additional observations may yield the greatest improvement in the estimation of the field.

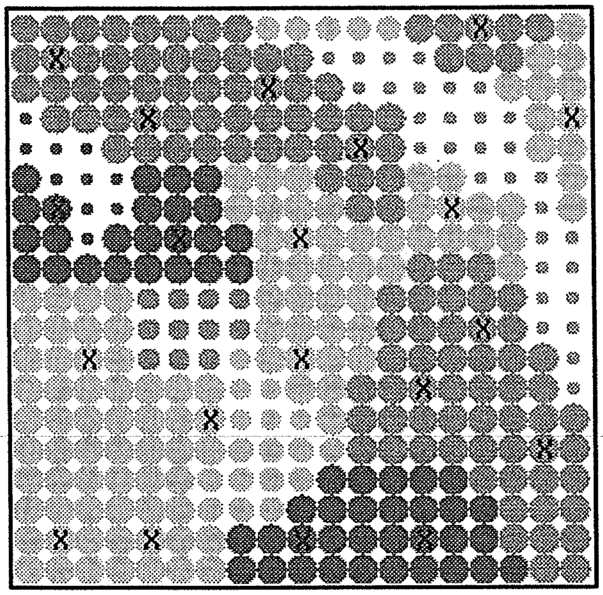

Figure 11. Estimated **Y** field with measure of sureness of prediction indicated by the size of the circle. This result was obtained from the NK network using 20 observations (uniformly random in location), 26 hidden layer units, and random initial weights.

4.3. Preliminary Comparison With Ordinary Kriging

A second two-dimensional reference field of Y was obtained by extracting a single slice from a $20 \times 20 \times 20$ random field. The latter was generated by the turning bands method [*Tompson et al.*, 1989] using 100 randomly oriented lines to avoid "striping." A Gaussian distribution with isotropic exponential covariance function was used, with variance $\sigma_Y^2 = 1$, mean value $\bar{Y} = 0$, and decorrelation scale λ. The lattice of points at which the random field was evaluated was selected such that $\Delta x = \Delta y = \Delta z$ and $\lambda = 4\Delta x$. The extraction of a single plane made it likely that this reference field would not be Gaussian itself (although an ensemble of samples would be).

Thirty points were selected at random as sample points in the plane. The observations were kriged using the $\sigma_Y^2 \exp(-h/\lambda)$ covariance function, with lag h, and the same decorrelation length, λ, used above. The reference field, the observed field, and the kriged field were quantized into the three Y classes: $Y < -\sigma_Y$, $-\sigma_Y \leq Y \leq \sigma_Y$, and $Y > \sigma_Y$. The resulting 20×20 maps are shown as Figures 12a–12c, where each observation is indicated by an M in the figure.

As mentioned previously, for good results the number of Kohonen units should be as large as the number of observation points. As a result, for this example the Kohonen weights were initialized to the location of the observation points plus a small random vector. The 30 observation points and quantized sample values were used to train a neural kriging network with 31 hidden nodes and a constant learning rate of $\alpha = 0.2$. About 100 iterations were required for an allowable RMS error of

Table 1. Evaluation of Biasedness of Quantized Fields Based on the Distribution of Hydraulic Conductivity Classes: Figure 12 Reference Field

	Low *K*		Medium *K*		High *K*	
Quantized Field	Count	Percent	Count	Percent	Count	Percent
Reference	130	32.50	137	34.25	133	33.25
Sample	6	20.00	12	40.00	12	40.00
Ordinary kriging	48	12.00	212	53.00	140	35.00
Neural kriging	73	18.25	184	46.00	143	35.75

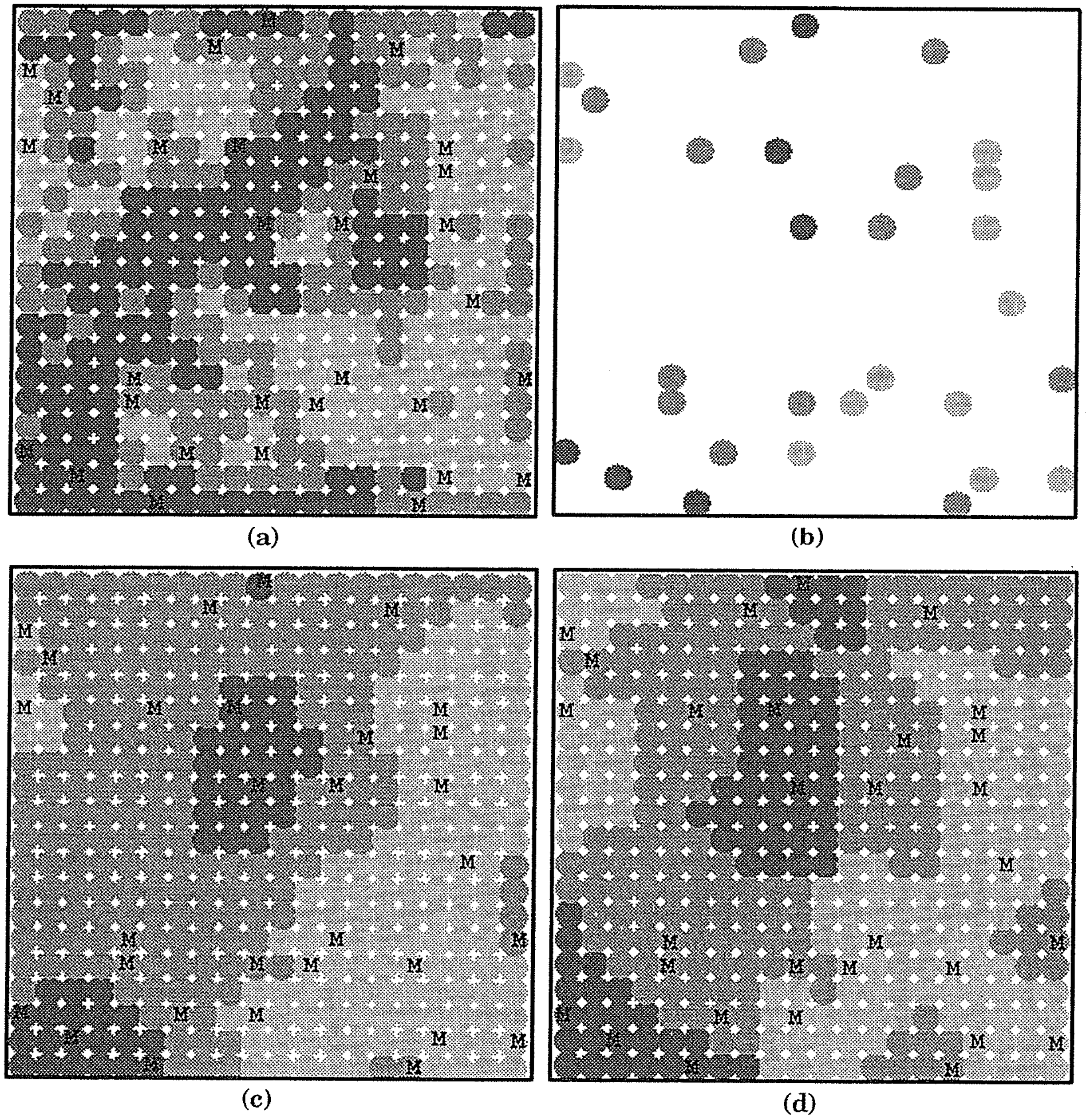

Figure 12. Comparison among (a) a quantized reference random field, (b) a quantized sampled field with 30 observations (M), (c) a quantized field obtained by ordinary kriging of the quantized observations, and (d) a quantized field obtained by neural kriging of the quantized observations.

Table 2. Evaluation of Biasedness of Quantized Fields Based on the Distribution of Hydraulic Conductivity Classes: Additional Reference Field

	Low *K*		Medium *K*		High *K*	
Quantized Field	Count	Percent	Count	Percent	Count	Percent
Reference	38	9.50	288	72.00	74	18.50
Sample	2	6.67	22	73.33	6	20.00
Ordinary kriging	5	1.25	366	91.50	29	7.25
Neural kriging	33	8.25	294	73.50	73	18.25

10^{-6}. The resulting estimated field is shown in Figure 12d; the sureness measure is suppressed here for clarity.

Both the quantized ordinary kriging (OK) and neural kriging (NK) approaches interpolate the observations, and both have many of the gross features of the reference field. The quantized OK result shows more localized effects of observations than does NK. On the other hand, OK does not represent the variability present in the reference field as well as NK. One measure, D, of error is obtained by summing over all estimation points (including observation points) the absolute value of the difference in cardinality between the Y class of the reference field and the estimated field. In this example $D_{\text{OK}} = 265$ while $D_{\text{NK}} = 267$. Both OK and NK give the correct results at every observation point.

Bias was checked in the reference field, sample field, OK field, and NK field (Table 1). Because a finite field was generated, the reference field is slightly biased toward the medium conductivities. The sample field, which was chosen at random with no attempt to reproduce the bias in the reference field, is slightly biased toward medium and high conductivities. The OK field is biased toward medium conductivities, with less low conductivity than is present in either the reference or sample fields. On the other hand, the NK approach provides a better distribution of the three conductivity classes (relative to the samples and reference field).

Variograms were constructed from the continuous reference and OK fields. These continuous values were then

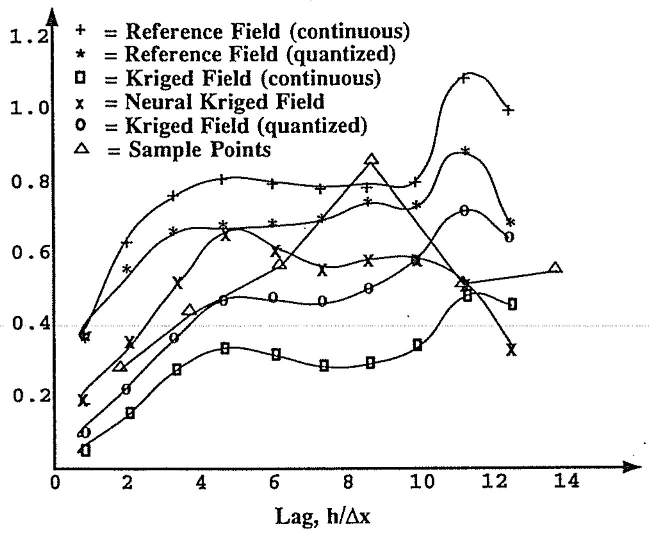

Figure 13. Variograms constructed from the quantized reference, NK, and OK fields as well as the continuous OK and reference fields.

quantized to obtain variograms for the quantized OK, NK, and quantized reference fields. Figure 13 shows that the continuous reference field, which is a single slice taken from a three-dimensional object, has a variance of about 0.8 and a decorrelation length of $4\Delta x$, as compared to the values 1.0 and 4 which were input to the random field generator. The variogram for the continuously valued samples exhibits about 80% of the variance of the complete reference field. The quantized reference field shows smaller variance but roughly the same decorrelation scale. NK shows about 60% of the variance of the quantized reference field while providing a reasonably similar decorrelation scale. On the other hand, quantized OK shows insufficient variance, consistent with the observed excessive smoothing (see Figure 12c). Oscillations in the variograms result from quantization and the finite extent of the domain.

Because the experimental (sample) variogram is not identical with the variogram input to the random field generator, comparisons were made between NK and OK with a number of estimated variograms [see *Stein and Handcock*, 1989; *Stein*, 1990, 1991]. If we assume an exponential decorrelation function, then it is apparent from Figure 13 that the exponential (range) coefficient is not well described by the data. Hence different OK runs were made using fixed variance of 0.8 and variable range coefficient uniformly distributed between $4\Delta x$ and $20\Delta x$. Figure 14 shows sample random fields produced by OK followed by quantization for two different range coefficients. Clearly, the range of possible OK results contains predicted maps that are very similar to the NK result. This provides a further qualitative demonstration that NK is a candidate method for aquifer characterization.

The NK network was compared with three additional quantized reference fields. The maps of Figure 12 and evaluation of bias reported in Table 1 show results of the comparison most favorable for OK. Table 2 shows the evaluation of biasedness for an additional two-dimensional reference field. This field was generated by extracting a new slice from the same 20 × 20 × 20 random field produced earlier by the turning bands method. The resulting field is less Gaussian than the previous example, with a large bias toward medium conductivity and a somewhat smaller bias toward high conductivity. A new set of 30 sample points were selected at random; again no attempt was made to reproduce the bias of the reference field. The samples are even more biased than the original reference field. For this example the OK field is strongly biased toward medium conductivity, with less low and high conductivity than is present in either the reference or sample fields. However, NK provides an excellent distribution of the three conductivity classes. Results of the remaining two comparisons

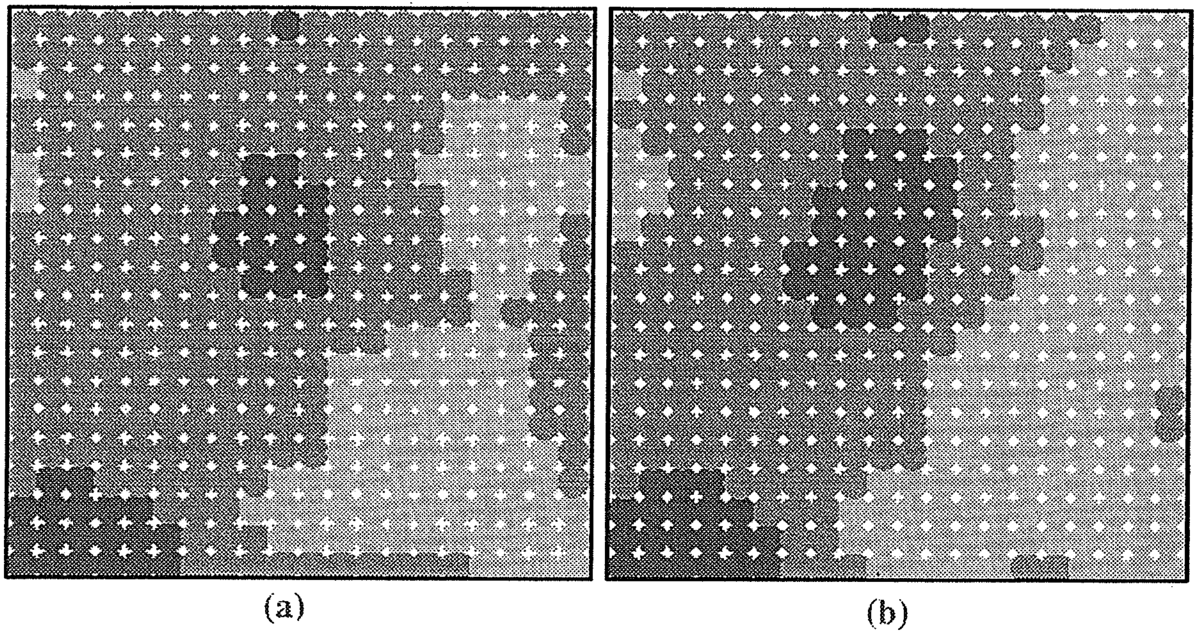

Figure 14. Sample random fields produced by OK followed by quantization for two different range coefficients: (a) $\lambda = 2$ and (b) $\lambda = 10$.

were similar. There is a need for further tests of NK with other methods. Since the Gaussian distribution is assumed rather than observed in most geohydrologic settings, it will be particularly useful to run tests on non-Gaussian fields.

5. Summary and Discussion

A new method of pattern completion for geohydrological applications has been introduced. Using artificial neural networks (ANNs), spatial estimations of discrete "quantized" functions for correlated fields given limited observations have been obtained using a highly parallel and nonparametric description of the field.

The resulting neural kriging (NK) network has been applied to two-dimensional examples. NK provides an estimate of the field in two dimensions and a measure of the sureness of the estimates. A sample plane was drawn from a three-dimensional Gaussian field generated with an isotropic exponential correlation structure; the plane itself was small, and its statistics are not Gaussian. Using a set of randomly located samples from within the plane, NK was trained and used to provide a map of discrete conductivity values. NK shows significantly less bias than OK in an example using the known distribution used to generate the sample field, very closely approximates the bias present in the reference field and in the set of 30 samples, and produces a variogram that matches the variogram of the quantized reference field very well. When the variogram for OK was estimated from the observations, rather than taken as known, the quantized OK predictions show a range of possible characters. The NK patterns are apparently similar to some members of this range of OK patterns.

The quality of the results is dependent on several factors. First, the NK network can be required to interpolate the observations by imposing a sufficiently stringent RMS error tolerance. If the allowed RMS error is too large, the interpolating property is lost. Our experience with such cases to date is that the network retains its ability to construct reasonable maps in spite of the loss of accuracy at the observation points.

Second, there must be a sufficient number of hidden layer nodes to assimilate the quantity of information being presented to the NK network. When an insufficient number are used, some portion of the information presented to the NK network is ignored; which information is ignored and in what way is dependent upon the details of the training protocol.

Third, the number of hidden units required can be significantly affected by the initial estimates of the weight vectors. Experiments show that the number of hidden nodes required to attain a given RMS error can be an order of magnitude more if random initial weight values are used, rather than carefully distributing the weight vectors according to the density of input vectors.

Many of the neural computing features used here are widely known. The use of two winner-take-all activation functions, one in the hidden Kohonen layer (trained by feed forward) and one in the output Grossberg layer (trained by back propagation), is unusual, but has appeared before [*Rumelhart and Zipser*, 1985]. A novel feature of our NK network is the use of the raw output of the Grossberg layer (before the activation function) to determine the sureness with which an estimate is being made. The measure of sureness introduced above, S, is easily computed and seems to provide information with the same utility as the variance of the estimation error that is produced in ordinary kriging.

A number of developments, extensions, and tests of the method are required and envisioned. There is no need to normalize the inputs and the weighting vectors, as shown above. A robust algorithm for setting the initial values of the weighting vectors is needed; work by *DeSieno* [1988], for example, provides a point of departure. Careful testing of NK with various kriging approaches (e.g., indicator kriging) for a variety of fields, especially for non-Gaussian ones, is necessary. We have already successfully performed extensions of the NK to more quanta (more output classes) and other pattern completion applications that incorporate "soft data" (e.g., fence diagrams in geohydrology); these will be reported elsewhere. The more formal development of a method of conditional simulation using the NK would make it much more useful. Finally, because the method requires simple computations and predetermined connectivity among artificial neurons, a VLSI implementation is possible.

Appendix

A simple MATLAB implementation of the neural kriging network described in this paper is available by anonymous ftp. A network connection to the Internet is necessary.

To retrieve the MATLAB program, ftp to ftp.emba.uvm.edu, using "anonymous" as the user and your email address as the password. Then type "cd pub/CME/NKN". Finally, to acquire all of the germane files, type in succession "get README", "get NKSP1.m", "get results.1", and "mget *.dat.1". The program is implemented for version 4.0 of MATLAB and requires no toolboxes. (If you do not know about MATLAB or its toolboxes, contact someone in your computer support group.) The program is self-documented.

This program is copyrighted and permission is granted for the use of this program for individual study. Commercial use is explicitly not allowed. The authors and the University of Vermont provide this code on an "as is" basis and accept no responsibility for errors, mistakes, or misrepresentations that may occur as a result of its use. A complete disclaimer is provided at the beginning of the program.

Our use of MATLAB is for product identification purposes and does not represent an endorsement of the product.

Notation

c	"winning" hidden unit (node for which the dot product between the normalized input vector and Kohonen weight vector is maximum).
h	lag distance.
I	number of input units (also equals the number of components in the normalized training vector $\mathbf{x}$).
J	number of Kohonen (hidden) layer units.
K	number of output units (also equals the number of log conductivity classes).
N	number of spatial coordinates.
M	total number of measurement points.
P	total number of prediction points.
P_B	probability of error of a Bayes classifier.
$P(\mathbf{Y}_k\|\mathbf{x})$	posterior probability.
S	measure of sureness.
v	Grossberg weight matrix.
v_{jk}	elements of the Grossberg weight matrix ($j = 1, 2, \cdots, J$) ($k = 1, 2, \cdots, K$) (subscript indicates a weight from hidden unit j to output unit k).
w	Kohonen weight matrix.
w_{ij}	elements of the Kohonen weight matrix ($i = 1, 2, \cdots, I$) ($j = 1, 2, \cdots, J$) (subscript indicates a weight from input node i to hidden node j).
$\mathbf{w}_j$	vector containing the jth column of the Kohonen weight matrix.
$\mathbf{x}$	input vector containing spatial information.
$\mathbf{x}^m$	input vector of M observation points.
$\mathbf{x}^p$	input vector at interpolation point (p indicates points at which we would like to predict some output variable).
x_i	elements of the input vector ($i = 1, 2, \cdots, I$) (for normalized vectors, I equals one plus the number of spatial coordinates, N).
Y	log conductivity.
Y^m	log conductivity measured at M observation points ($m = 1, 2, \cdots, M$).
$\bar{Y}$	mean value of measured log conductivity.
$\mathbf{Y}$	quantized vector of outputs (Grossberg layer output Y passed through a "winner-take-all" activation function).
$\mathbf{Y}^m$	target output vectors (quantized log conductivity) at M observation points.
$\mathbf{Y}^p$	quantized vector of outputs approximated by the network at prediction points.
$\mathbf{y}$	raw Grossberg layer output.
$\mathbf{y}^m$	raw vector of outputs approximated by the network at M observation points.
$\mathbf{y}^p$	raw vector of outputs approximated by the network at P prediction points.
y_k^m	elements of the output vector estimated for M observation points.
y_k^p	elements of the output vector estimated for P prediction points.
$\mathbf{z}$	Kohonen (or hidden) layer output vector.
z_j	elements of Kohonen layer output ($j = 1, 2, \cdots, J$) (dot product between $\mathbf{x}$ and w)
$\mathbf{Z}$	Kohonen layer output, $\mathbf{z}$ passed through a "winner-take-all" activation function.
Z_j	elements of the Kohonen layer output vector, $\mathbf{Z}$.
α	learning constant ranging between zero and one.
β	learning constant ranging between zero and one.
$\Delta x = \Delta y$	grid spacing.
ε	root-mean-square training error.
λ	decorrelation scale.
σ_Y	standard deviation of Y.
σ_Y^2	variance of Y.

Acknowledgments. Computational aspects of this work were supported by National Science Foundation grant ASC-9100262. Additional support for the work was provided by NSF grant MSS-9214698 and by Lawrence Livermore National Laboratory under subcontract B-218791. We are pleased to acknowledge discussions with Leah L. Rogers of LLNL and Robert A. Snapp of the Department of Computer Science and Electrical Engineering at UVM. We also thank Michael Stein of the University of Chicago for suggestions that strengthened the comparison with ordinary kriging.

References

Aziz, A. R. A., and K. F. V. Wong, A neural-network approach to the determination of aquifer parameters, *Ground Water*, *30*, 164, 1992.

Chiu, C., C.-Y. Maa, and M. A. Shanblatt, An artificial neural network algorithm for dynamic programming, *Int. J. Neural Syst.*, *1*, 211, 1990.

Cover, T. M., and P. E. Hart, Nearest neighbor pattern classification, *IEEE Trans. Inf. Theory*, *IT-13*, 21, 1967.

DeSieno, D., Adding a conscience to competitive learning, in *IEEE International Conference on Neural Networks*, pp. 117–124, Institute of Electrical and Electronics Engineers, New York, 1988.

Dowla, F. U., S. R. Taylor, and R. W. Anderson, Seismic discrimination with artificial neural networks: Preliminary results with regional spectral data, *Bull. Seismol. Soc. Am.*, *80*, 1346, 1990.

Duda, R. O., and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, New York, 1973.

Dysart, P. S., and J. J. Pulli, Regional seismic event classification at the NORESS array: Seismological measurements and the use of trained neural networks, *Bull. Seismol. Soc. Am.*, *80*, 1920, 1990.

Epping, W. J. M., and A. R. L'Istelle, Data compression of seismic images by neural networks, *Rev. Inst. Fr. Pet.*, *47*, 243, 1992.

Fitch, J. P., S. K. Lehman, and F. U. Dowla, Ship wake detection procedure using conjugate gradient trained artificial neural networks, *IEEE Trans. Geosci. Remote Sens.*, *29*, 718, 1991.

Gomez-Hernandez, J. J., and R. M. Srivastava, ISIM3D: An ANSI-C three-dimensional multiple indicator conditional simulation program, *Comput. Geosci.*, *16*(4), 395, 1990.

Grossberg, S., Some networks that can learn, remember, and reproduce any number of complicated space-time patterns, *J. Math. Mech.*, *19*, 53–91, 1969.

Grossberg, S., *Studies of Mind and Brain*, D. Reidel, Norwell, Mass., 1982.

Hecht-Nielsen, R., Counterpropagation networks, *Appl. Opt.*, *26*(23), 4979, 1987a.

Hecht-Nielsen, R., Counterpropagation networks, in *IEEE International Conference on Neural Networks*, vol. 2, pp. 19–32, Institute of Electrical and Electronics Engineers, New York, 1987b.

Hecht-Nielsen, R., Applications of counterpropagation networks, *Neural Networks*, 1, 131, 1988.

Hepner, G. F., T. Logan, and N. Ritter, Artificial neural network classification using a minimal training set: Comparison to conventional supervised classification, *Photogramm. Eng. Remote Sens.*, *56*, 469, 1990.

Hertz, J., A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, Mass., 1991.

Johansson, E. M., F. U. Dowla, and D. M. Goodman, Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method, *Int. J. Neural Syst.*, *2*, 291, 1991.

Journel, A. G., The place of non-parametric geostatistics, in *Geostatistics for Natural Resources Characterization*, part 1, edited by G. Verly, M. David, A. G. Journel and A. Marechal, pp. 307–335, D. Reidel, Norwell, Mass., 1984.

Journel, A. G., *Fundamentals of Geostatistics in Five Lessons*, *Short Course Geol. Ser.*, vol. 8, 38 pp., AGU, Washington, D. C., 1989.

Journel, A. G., and C. J. Huijbregts, *Mining Geostatistics*, Academic, San Diego, Calif., 1978.

Kanellopoulos, I., A. Varfis, and G. G. Wilkinson, Land-cover discrimination in SPOT HRV imagery using an artificial neural network—A 20-class experiment, *Int. J. Remote Sens.*, *13*, 917, 1992.

Kohonen, T., *Self-Organization and Associative Memory*, 2nd ed., Springer-Verlag, New York, 1988.

Massmann, J. A., and R. A. Freeze, Updating random hydraulic conductivity fields: A two-step procedure, *Water Resour. Res.*, *25*(7), 1763, 1989.

Matheron, G., *The Theory of Regionalized Variables and Its Applications*, *Cah. Centre Morphol. Math.*, vol. 5, Ecole Nationale Superieure des Mines de Paris, Fontainebleau, France, 1971.

McCulloch, W. S., and W. Pitts, A logical calculus of ideas immanent in nervous activity, *J. Math. Biophys.*, *5*, 115–133, 1943.

Minsky, M. L., and S. Papert, *Perceptrons*, MIT Press, Cambridge, Mass., 1969.

Pulli, J. J., and P. S. Dysart, An experiment in the use of trained neural networks for regional seismic event classification, *Geophys. Res. Lett.*, *17*, 977, 1990.

Ranjithan, S., J. W. Eheart, and J. H. Garrett, Jr., Neural network–based screening for groundwater reclamation under uncertainty, *Water Resour. Res.*, *29*(3), 563, 1993.

Ritter, N. D., and G. F. Hepner, Application of an artificial neural network to land-cover classification of thematic mapper imagery, *Comput. Geosci.*, *16*, 873, 1990.

Rumelhart, D. E., and D. Zipser, Feature discovery by competitive learning, *Cognit. Sci.*, *9*, 75, 1985.

Specht, D. F., Probabilistic neural networks, *Neural Networks*, *3*, 109, 1990a.

Specht, D. F., Probabilistic neural networks and the polynomial as complementary techniques for classification, *IEEE Trans. Neural Networks*, *1*(1), 111, 1990b.

Stein, M. L., Bounds on the efficiency of linear predictions using an incorrect covariance function, *Ann. Stat.*, *118*, 1116, 1990.

Stein, M. L., A kernel approximation to the kriging predictor of a spatial process, *Ann. Inst. Stat. Math.*, *43*, 61, 1991.

Stein, M. L., and M. S. Handcock, Some asymptotic properties of kriging when the covariance function is misspecified, *Math. Geol.*, *21*, 171, 1989.

Sullivan, J., Conditional recovery estimation through probability kriging—Theory and practice, in *Geostatistics for Natural Resources Characterization*, part 1, edited by G. Verly, M. David, A. G. Journel, and A. Marechal, pp. 365–384, D. Reidel, Norwell, Mass., 1984.

Tompson, A. F. B., R. Ababou, and L. W. Gelhar, Implementation of the three-dimensional turning bands random field generator, *Water Resour. Res.*, *25*(10), 2227, 1989.

Zitko, V., Prediction of biodegradability of organic chemicals by an artificial neural network, *Chemosphere*, *23*, 305, 1991.

D. E. Dougherty and D. M. Rizzo, Research Center for Groundwater Remediation Design, Department of Civil Engineering, 213 Votey Building, University of Vermont, Burlington, VT 05405 (tel.: (802)656-1920, (802)656-1941; fax: (802)656-8446; e-mail: ddougher@emba.uvm.edu, drizzo@emba.uvm.edu).

(Received February 3, 1993; revised July 23, 1993;
accepted August 27, 1993.)