Mahalanobis clustering, with applications to AVO classification and seismic reservoir parameter estimation

Brian H. Russell and Laurence R. Lines

ABSTRACT

A new clustering algorithm, Mahalanobis clustering, is proposed as an improvement on traditional *K*-means clustering. We present applications of this method to both AVO classification and seismic reservoir parameter estimation using multiple attributes. In the latter application, we use the radial basis function neural network (RBFN) with centres, and apply Mahalanobis clustering to find the cluster centres that are used in the training of the network. We also show that this method allows us to improve the estimate of the covariance matrix parameters used in the general form of the RBFN approach.

INTRODUCTION

Clustering has been long been discussed in the field of multivariate statistics (Johnson and Wichern, 1998), but has only recently found application in seismic analysis. In clustering, we seek to find natural groupings, or clusters, within a particular data set. These clusters could pertain to different lithologies, fluid content, etc. Although closely related to classification, clustering is more basic in that we normally do not know how many clusters we have, or what form these clusters will take. Classification is usually supervised by the knowledge of what our classes should look like, whereas clustering is unsupervised, without any such knowledge.

A key question is where we should do the clustering. In this study, we will apply the clustering in multi-dimensional parameter space. The simplest example of this is two-dimensional space, and our study will involve A-B crossplots from the AVO technique (Ross, 2000). The advantage of using two-dimensional space is that it is easily visualized and the results can be checked by eye. The method can also be applied to higher dimensional spaces, which cannot be visualized. For this example, we chose multi-attribute space, in which multiple seismic attributes are grouped and used to predict reservoir parameters (Hampson et al., 2001). In the following discussion, we will refer to the group of L attributes for a single recorded time sample in L-dimensional space as an attribute vector. In general, there will be N recorded time samples.

We will start by considering the simplest clustering approach, called *K*-means clustering (MacQueen, 1967). This method is based on the Euclidean distance between points. We will show that this method works well for the case of well-separated, roughly spherical clusters, but not when the clusters become elliptical in shape. We then propose a new method called Mahalanobis clustering, in which statistical distance, rather than Euclidean distance, is used for the clustering. Although the use of this distance metric was discussed by Johnson and Wichern (1998), they state that it is not used in practice, and do not discuss a method for implementing this procedure.

K-MEANS CLUSTERING

In the *K*-means clustering technique, we start with a random estimate of the cluster centres, and iterate toward a solution by minimizing the distance from each input cluster centre to the points surrounding it. As pointed out by Haykin (1998) this has the desirable property of placing the centres of the clusters in those regions of the input space where significant amounts of data are present. The steps involved in *K*-means clustering are as follows:

1. Decide on a number of clusters, *K*, and divide the input data points randomly into these *K* clusters. If we have *N* *L*-dimensional input vectors of attributes, $\boldsymbol{x_i}$, we initially set the number of points in each of the first *K-1* clusters to $N_k = int(N/K)$, and the last cluster equal to $N - (N_k*(K-1))$. Also, the decision of what value to assign to *K* is important and will affect the result.

2. Compute the means $\boldsymbol{\mu_k}$, where $k = 1, 2, \ldots, K$. The means are simply the sum of the N_k attribute vectors divided by N_k, where N_k is the number of points in the k^{th} cluster.

3. Compute the matrix of distances $d_{ik} = | \boldsymbol{x_i} - \boldsymbol{\mu_k} |$, and assign each input attribute vector to the cluster for which this distance is a minimum. Note that the N_k values are now updated.

4. Re-compute the means based on the new cluster assignments.

5. Iterate through steps 2-4 until convergence.

Obviously, the *K*-means clustering algorithm can be computed for any number of input points, attributes and clusters. Before considering real data application, let us consider the following numerical example in which we have the eighteen input vectors given by:

$$\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \boldsymbol{x}_2 = \begin{bmatrix} 8 \\ 1 \end{bmatrix}, \boldsymbol{x}_3 = \begin{bmatrix} 4 \\ 9 \end{bmatrix}, \boldsymbol{x}_4 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \boldsymbol{x}_5 = \begin{bmatrix} 8 \\ 2 \end{bmatrix}, \boldsymbol{x}_6 = \begin{bmatrix} 5 \\ 5 \end{bmatrix},$$

$$\boldsymbol{x}_7 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \boldsymbol{x}_8 = \begin{bmatrix} 9 \\ 3 \end{bmatrix}, \boldsymbol{x}_9 = \begin{bmatrix} 6 \\ 6 \end{bmatrix}, \boldsymbol{x}_{10} = \begin{bmatrix} 1 \\ 6 \end{bmatrix}, \boldsymbol{x}_{11} = \begin{bmatrix} 9 \\ 1 \end{bmatrix}, \boldsymbol{x}_{12} = \begin{bmatrix} 7 \\ 7 \end{bmatrix},$$

$$\boldsymbol{x}_{13} = \begin{bmatrix} 2 \\ 8 \end{bmatrix}, \boldsymbol{x}_{14} = \begin{bmatrix} 10 \\ 2 \end{bmatrix}, \boldsymbol{x}_{15} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}, \boldsymbol{x}_{16} = \begin{bmatrix} 3 \\ 7 \end{bmatrix}, \boldsymbol{x}_{17} = \begin{bmatrix} 10 \\ 3 \end{bmatrix}, \boldsymbol{x}_{18} = \begin{bmatrix} 8 \\ 5 \end{bmatrix}.$$

These eighteen vectors are plotted in Figure 1, with the input vector number labelled on the graph. Note that we see four distinct clusters, although the order of the input points is random, and bears no relationship to the cluster order.

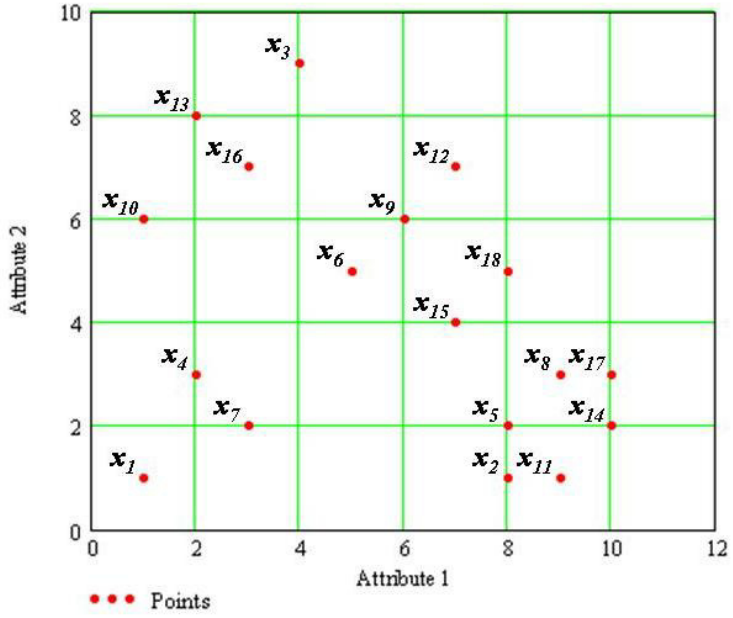


FIG. 1. The red dots show a set of eighteen points, grouped in four clusters, with the labels indicating the input order of the two-dimensional attribute vectors.

We will perform the first step of the *K*-means clustering by assuming that we have four clusters, although the optimal number of clusters will not be obvious in a real data set. This will give us the result that the first three clusters have four points each, and the last cluster has six points. The resulting means are

$$\mu_1 = \begin{bmatrix} 3.75 \\ 3.5 \end{bmatrix}, \mu_2 = \begin{bmatrix} 6.25 \\ 3 \end{bmatrix}, \mu_3 = \begin{bmatrix} 5.75 \\ 5 \end{bmatrix}, \text{and } \mu_4 = \begin{bmatrix} 6.667 \\ 4.833 \end{bmatrix}.$$

The means of the four clusters are plotted using blue crosses on Figure 2(b). (Figure 2(a) is simply a repeat of Figure 4, for comparison purposes). Notice that all of the means are grouped together in the centre of the plot and do not do a very good job of defining the actual cluster means. By the central limit theorem, we know that as the number of points increases, and they continue to be random, the initial means should be grouped together near the total population mean. After the first iteration, we find that the number of points in each cluster is $n_1 = 4$, $n_2 = 6$, $n_3 = 5$, $n_4 = 3$, and their means become

$$\mu_1 = \begin{bmatrix} 1.75 \\ 3 \end{bmatrix}, \mu_2 = \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \mu_3 = \begin{bmatrix} 4 \\ 7 \end{bmatrix}, \text{and } \mu_4 = \begin{bmatrix} 7.333 \\ 5.333 \end{bmatrix}.$$

Although the algorithm hasn't quite got the number of points right yet, the means have spread out and become more reasonable. In fact, the second cluster, with six points and a mean of (9, 2), has been computed exactly.

Since the algorithm has not found the solution yet, we will go to a second iteration, which leads to the values n1 = 4, n2 = 6, n3 = 4, n4 = 4, with the means

$$\mu_1 = \begin{bmatrix} 1.75 \\ 3 \end{bmatrix}, \mu_2 = \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \mu_3 = \begin{bmatrix} 3.5 \\ 7.55 \end{bmatrix}, and\ \mu_4 = \begin{bmatrix} 7 \\ 5.5 \end{bmatrix}.$$

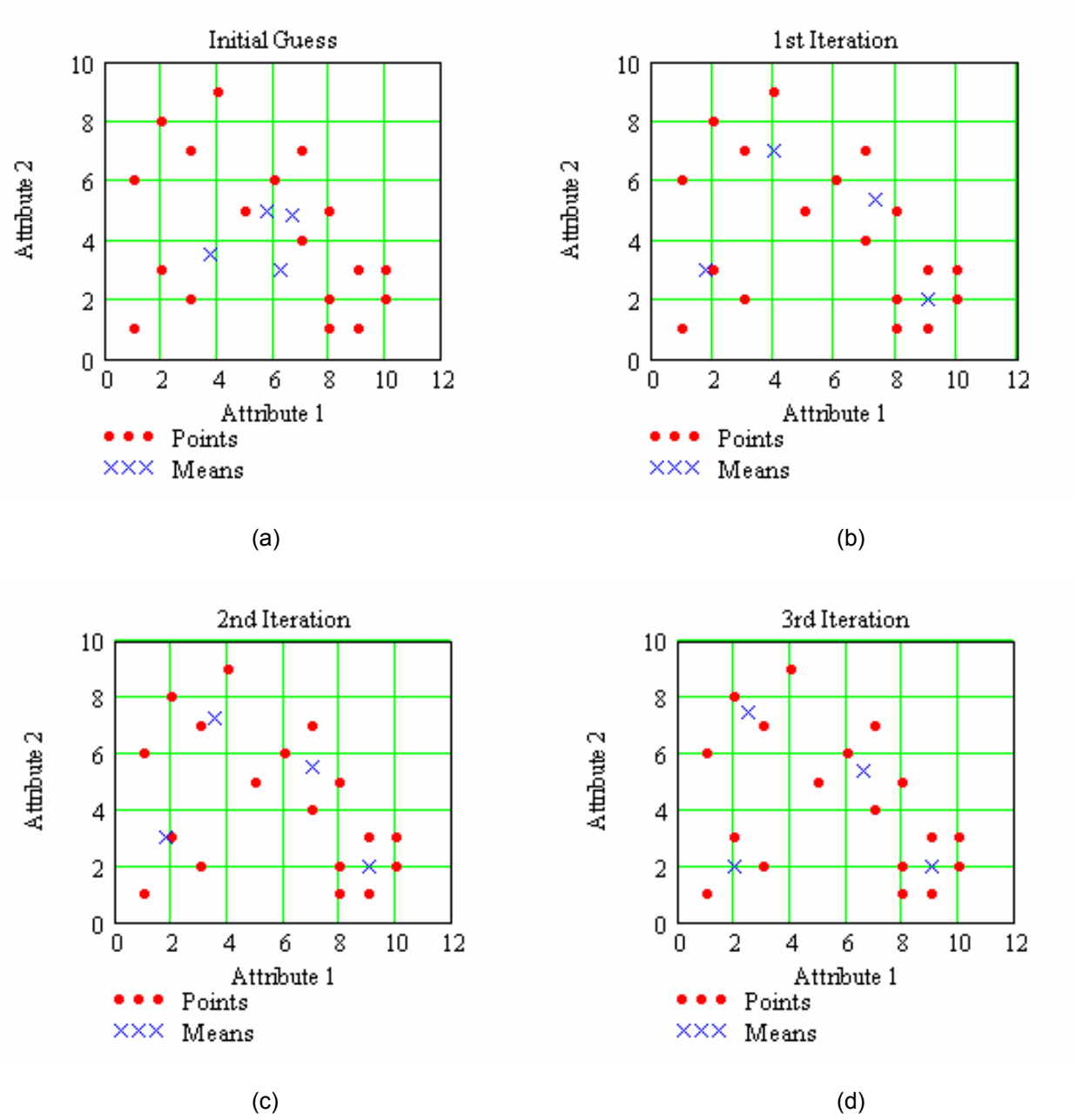


FIG. 2. The results of applying the *K*-means clustering example to a simple example, which consists of eighteen points in four clusters, where (a) shows the initial calculation, (b) shows the result of the first iteration, (c) shows the result of the second iteration, and (d) shows the result of the third, and last, iteration. After the three iterations, the algorithm has converged perfectly to the right answer.

The results of the second iteration are shown in Figure 2(c). Although cluster 1 hasn't improved, note that clusters 3 and 4 have moved closer to their correct positions. Also, it is encouraging to note that cluster 2, which was correct, has not changed. Finally, notice that the number of points in each cluster is different, but still wrong. So we will go to a third iteration, which leads to the values $n_1 = 3$, $n_2 = 6$, $n_3 = 4$, $n_4 = 5$, with means:

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \boldsymbol{\mu}_2 = \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \boldsymbol{\mu}_3 = \begin{bmatrix} 2.5 \\ 7.5 \end{bmatrix}, \text{and } \boldsymbol{\mu}_4 = \begin{bmatrix} 6.6 \\ 5.4 \end{bmatrix}.$$

Comparing these to the known right answers, notice that the results are now perfect. This is also obvious when referring to Figure 2(d). In this case, we have converged after three iterations.

Now we will consider a second example, shown in Figure 3. In this case we have created three elliptical clusters trending at about -45 degrees. As we will see, this is a synthetic example of a class 3 AVO anomaly (Russell et al., 2002).

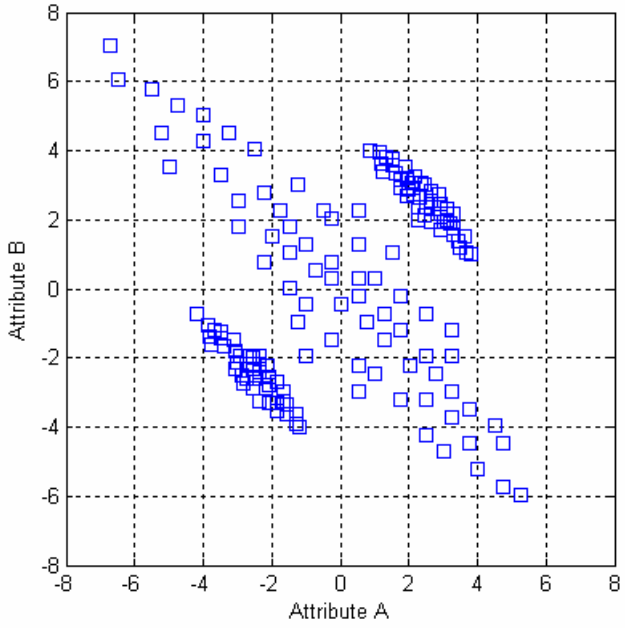
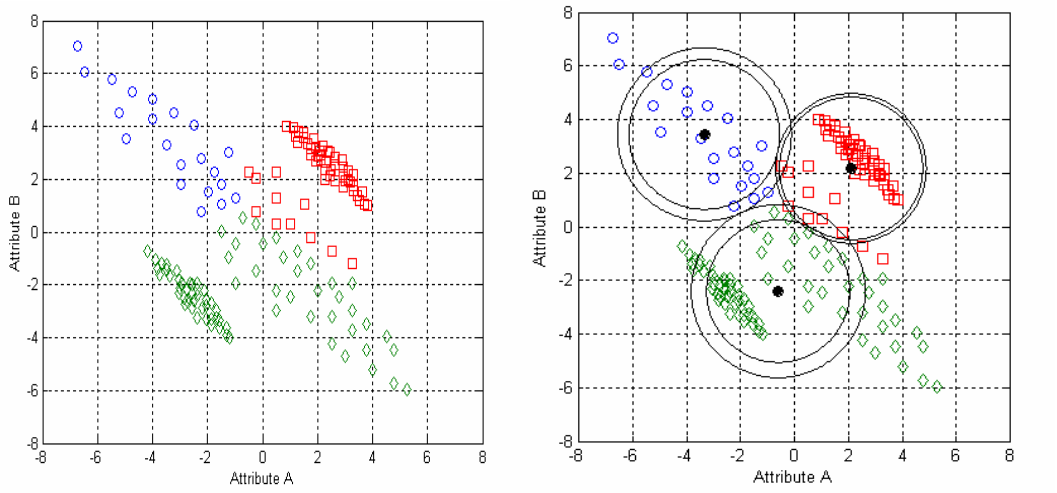


FIG. 3. A second input data set to the K-means clustering algorithm. This data set simulates a typical AVO A-B crossplot.

It would appear that the job of partitioning this set of points into three clusters would be fairly trivial. The results of 20 iterations of *K*-means clustering on the example shown in Figure 3 are shown in Figure 4. Figure 4(a) shows the classified points, where the blue circles, red squares, and green diamonds show, respectively, the three clusters. Notice that the three obvious clusters from Figure 3 have not been correctly classified. The reason for this is shown in Figure 4(b), where black dots show the three means, and two circles have been drawn around each of the means. The circles have radii equal to half the distance between each pair of means. Thus, the *K*-means algorithm has classified points into clusters that fall into circular groups, not the elongated ellipses seen in

Figure 3. As we shall show in the next section, there is a theoretical reason for this, and we can make use of this theory to develop an algorithm that will correctly classify this example.



(a) (b)

FIG. 4. The application of the *K*-means clustering algorithm to the input data set of Figure 3, where (a) shows the three output clusters (blue circles, green diamonds, and red squares), and (b) shows the cluster centres (black circles) with circles indicating the mid-point distance between cluster centres.

MAHALANOBIS CLUSTERING

As discussed in the last section, *K*-means clustering is based on the Euclidean distance, which can be written

$$d_{ik}^2 = \left|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right|^2 = (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T(\boldsymbol{x}_i - \boldsymbol{\mu}_k),\ i = 1,\ldots,N;\ k = 1,\ldots,K, \tag{1}$$

where

$$\boldsymbol{x}_i^T = [x_{i1}, x_{i2}, \ldots, x_{iL}], \text{and } \boldsymbol{\mu}_k^T = [\mu_{i1}, \mu_{i2}, \ldots, \mu_{iL}].$$

Another type of distance is the statistical, or Mahalanobis, distance (Johnson and Wichern, 1998) from **x** to **μ**, which can be written

$$\Delta_{ik}^2 = (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k), \tag{2}$$

where Σ is the covariance matrix given by

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1L} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{L1} & \sigma_{L2} & \cdots & \sigma_{LL} \end{bmatrix},$$

and the individual covariance values of Σ are computed from the outer product sum given by

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T .$$

Note that the Mahalanobis distance can also be interpreted as the exponent in the multivariate Gaussian distribution, which is given by

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{\Sigma}|^{1/2}} exp\left[-\frac{\Delta^2}{2}\right]. \tag{3}$$

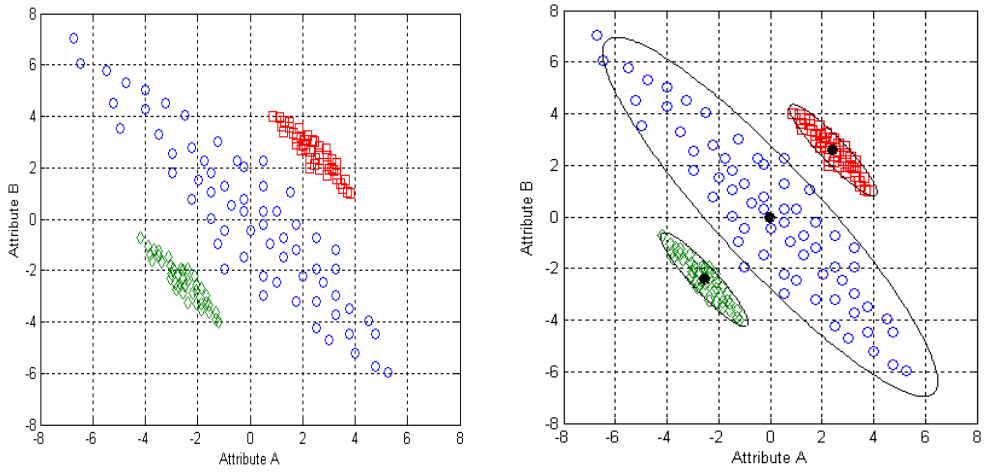
Also note that the covariance matrix with statistically independent variates and unit variances is equal to the identity matrix. That is, if

$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{4}$$

then

$$\Delta_{ij}^2 = (\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) = (\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T(\boldsymbol{x}_i - \boldsymbol{\mu}_j), \tag{5}$$

which is the Euclidean distance. Thus, Mahalanobis distance can be seen as the generalization of Euclidean distance, and can be computed for each cluster if the covariances of the cluster are known.



(a) (b)

FIG. 5. The application of the Mahalanobis clustering algorithm to the input data set of Figure 3, where (a) shows the three output clusters (blue circles, green diamonds, and red squares), and (b) shows the cluster centres (black circles) with the ellipses showing lines of constant variance.

After the application of the *K*-means clustering algorithm shown in Figure 4, we can then compute the means and covariances of each cluster, which gives us an initial estimate. Then, by iterating through the same steps given at the start of the last section, but using Mahalanobis distance rather than Euclidean distance, we can improve the clustering. We refer to this as Mahalanobis clustering. The result of applying 20 iterations of Mahalanobis clustering is shown in Figure 5(a). Notice that the cluster values are now correctly assigned. Figure 5(b) shows the lines of equal bivariate gaussian amplitude, illustrating that the elliptical clusters have indeed been captured.

Next, we will consider a real data example using the AVO crossplot method.

AVO CROSSPLOT CLUSTERING

An obvious application of the theory presented in the previous two sections is to AVO crossplot analysis (Ross, 2000). To illustrate the method, we will use a data example from the Colony sand play in central Alberta. A seismic section across a known gas sand is shown in Figure 6. Although the gas sand is an obvious "bright spot", there are many false "bright spots" in the area due to thin limestone stringers. In Figure 6, the sonic log from the discovery well has been inserted at its correct location, after conversion from time to depth and the application of a check-shot correction.

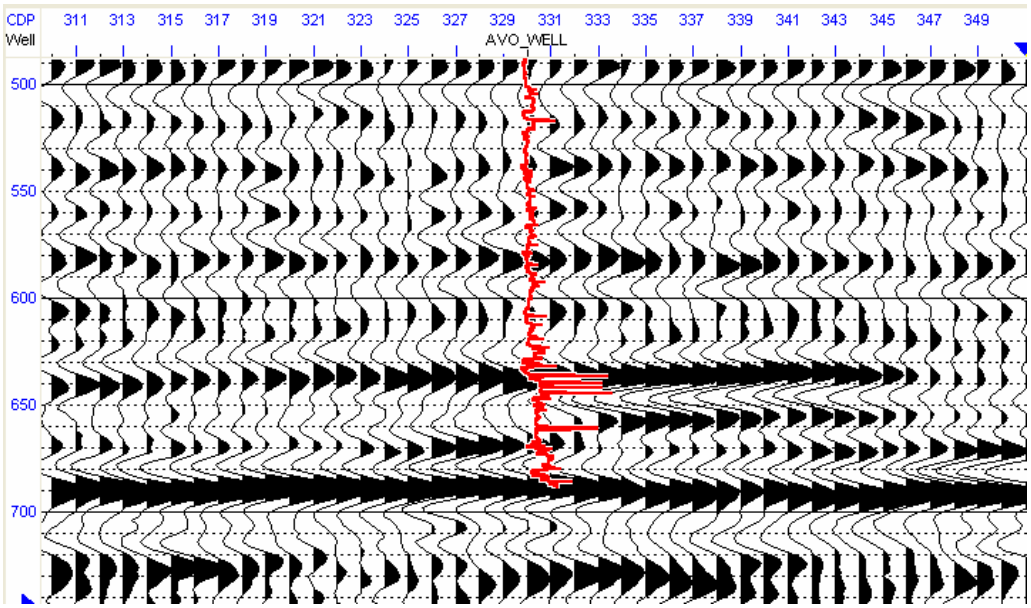
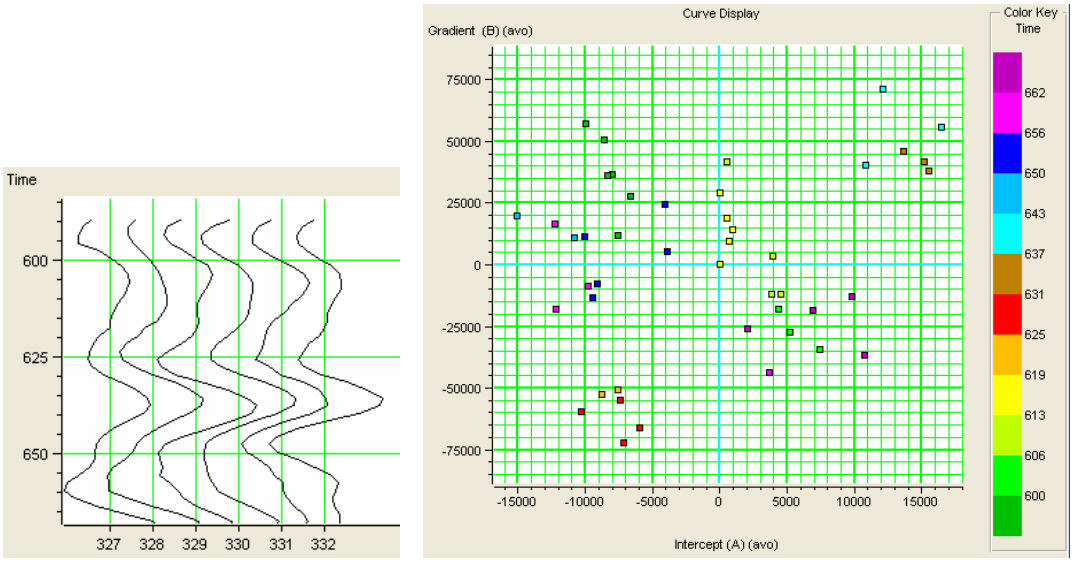


FIG. 6. A seismic line over a known gas zone, with the sonic log from the discovery well overlain at CDP 330. The gas sand is indicated by the "bright spot" at a time of 630 ms.

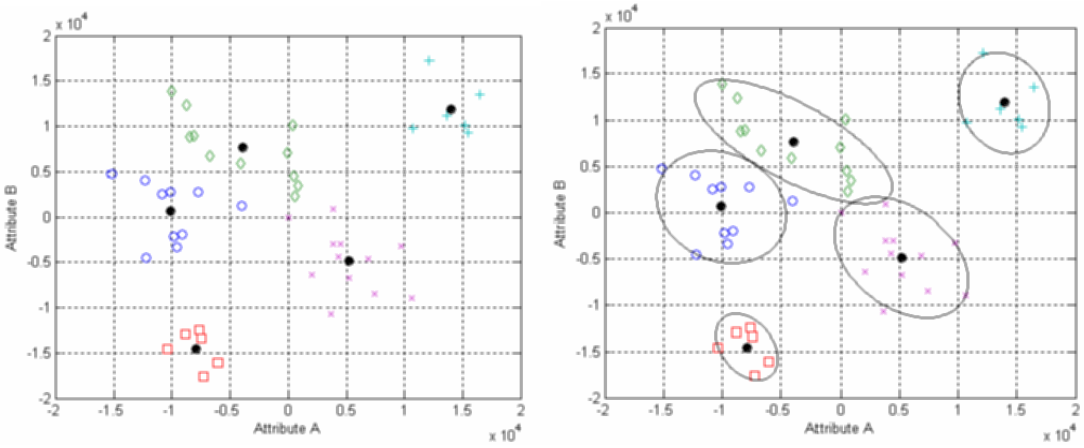
A small window of the seismic section shown in Figure 6 is shown in Figure 7(a), with the intercept (A) versus gradient (B) crossplot from the peaks and troughs in this window shown in Figure 7(b). This window is between CDP 327 and 332 and encompasses a time window of 80 ms, centred on a time of 630 ms.



(a) (b)

FIG. 7. AVO intercept/gradient crossplot analysis over a window from the seismic section of Figure 6, where (a) is the seismic window, and (b) is the uninterpreted crossplot.

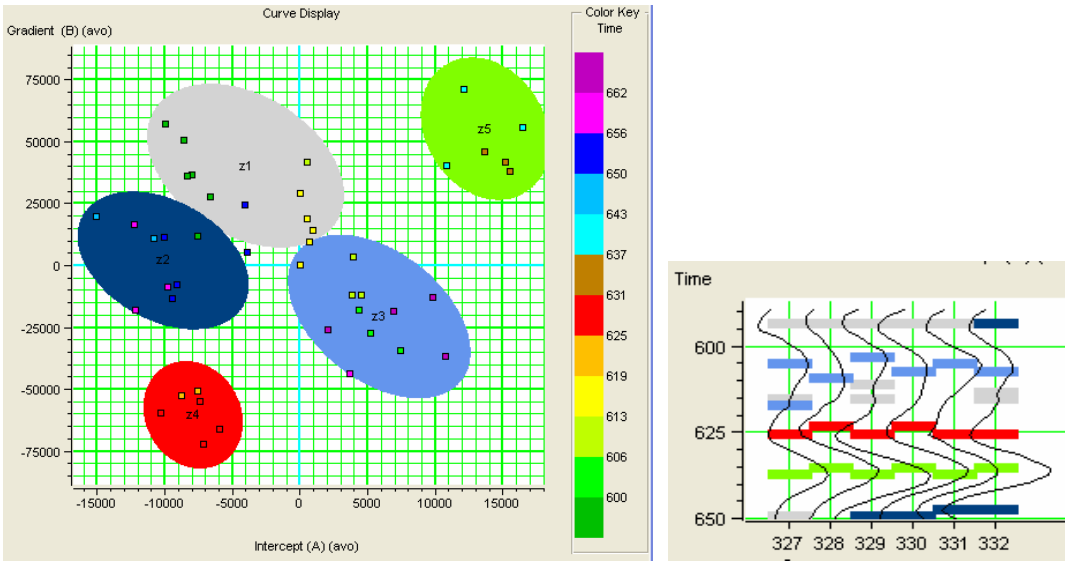
In the crossplot shown in Figure 7(b), the colour scale represents time. As discussed by Ross (2000) and Russell et al. (2002), the human interpreter would interpret this crossplot as a Class 3 AVO anomaly, with a "wet trend" visible as a cluster of points running along a -90 degree line, and two anomalous clusters in the first and third quadrants of the crossplot. These anomalies represent the top (third quadrant) and base (first quadrant) of the gas sand.



(a) (b)

FIG. 8. The application of (a) the *K*-means clustering algorithm, and (b) the Mahalanobis clustering algorithm to the crossplot of Figure 7(b), where the different shapes and colours indicate the five clusters, the black dots show the cluster centres, and the ellipses are equal variance lines enclosing the Mahalanobis clusters.

We will next apply the clustering algorithms described earlier to the crossplot shown in Figure 7. This results are shown in Figure 8, where Figure 8(a) shows the application of twenty iterations of the K-means algorithm, and Figure 8(b) shows the application of a further twenty iterations of the Mahalanobis algorithm to the output of the K-means algorithm. In Figure 8, notice that five clusters have been used in the training of the algorithm. This was due to the fact that there appeared to be more than three obvious clusters on the crossplot. (It should be pointed out that the number of clusters, or centres, to use is not a trivial issue, and has been discussed by many authors. For example, Chen et al. (1991) use an orthogonal least-squares learning algorithm to determine the number of centres.) The K-means clustering algorithm (Figure 8(a)) has done an excellent job in finding five distinct clusters of points, and these are indicated by five different shapes and colours on the figure. When the output from the K-means algorithm was processed through twenty iterations of the Mahalanobis algorithm, as shown in Figure 8(b), the points in the clusters and the cluster centres do not change. This confirms our feeling that, in this case, the K-means algorithm has done the optimum job. However, the advantage of the Mahalanobis method in this case is that it defines the elliptical shapes (and their quantitative parameters) that can be used for the application of clustering back to the crossplot of Figure 7. The results of this clustering are shown in Figure 9.



(a) (b)

Figure 9: The application of the Mahalanobis clustering results shown in Figure 8(b) to the original crossplot of Figure 7(b), where (a) shows the crossplot with the ellipse superimposed, and (b) shows the application back to the seismic traces of Figure 7(a).

In Figure 9(a) we have plotted the result of applying the elliptical shapes from the Mahalanobis clustering to the original crossplot of Figure 7(b), and in Figure 9(b) we have shown the equivalent peaks and troughs on the seismic window shown in Figure 7(a). Notice that the identified clusters appear to define coherent events in the seismic window. Although we have derived the crossplot clustering parameters over a very small window, we can apply the results of the clustering to the complete data set. Figure 10

shows the application of the training done in Figures 8 and 9 to a portion of the complete line shown in Figure 6.

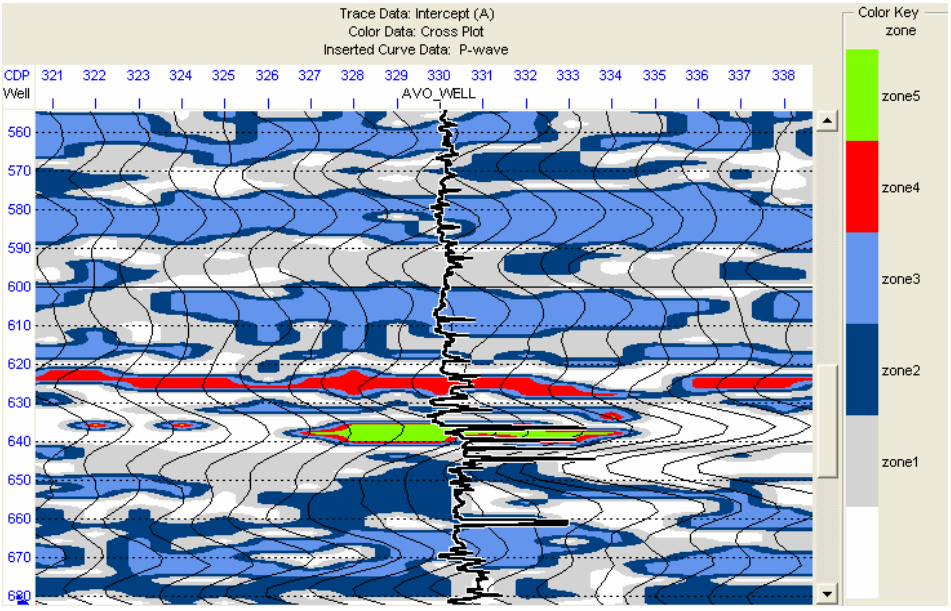


FIG. 10. The application of the Mahalanobis clustering results shown in Figure 9(a) to the complete seismic line of Figure 6. Only a portion of the line is shown in this figure.

In Figure 10, we clearly see the top of the gas sand (in red, called zone 4 on the colour bar) and the base of the gas sand (in green, called zone 3 on the colour bar). Also, the other three clusters, which are clearly on the wet trend in Figures 8 and 9, all plot on parts of the seismic line that are non-anomalous, as expected. Thus, the *K*-means/Mahalanobis clustering method has produced an excellent result in this case. However, before we get too excited about this result, it is important to note that a human interpreter could do at least as good a job, and probably better, in identifying these anomalies on the crossplot. Also, the method depends on having a certain amount of separation between the anomalies and the wet trend, which is not always the case.

Thus, two-dimensional clustering is probably best left to human beings. The real power of this method is when we apply it to higher dimensional spaces, especially those having more than three dimensions, the limit of our visualization skills. This extension of the *K*-means clustering algorithm will be discussed in the next section, in which the method is applied to the radial basis function neural network used for predicting reservoir parameters (Russell et al., 2002a), along with a suggestion of how Mahalanobis clustering can be applied.

RADIAL BASIS FUNCTION NEURAL NETWORK APPLICATION

As we have just seen, the Mahalanobis clustering method can be applied successfully to clustering in two-dimensional space, but an argument can be made that the human interpreter can still outperform the algorithm. However, clustering techniques come into their own in *N*-dimensional space, where *N* is greater than 3, since there is no possibility of visualizing such high dimensional space. One area of research that can benefit from cluster analysis is the radial basis function neural network, or RBFN. In last year's

CREWES report, several papers were presented describing the RBFN method (Russell et. al, 2002a, Russell et al. 2002b). In the first paper (Russell et. al, 2002a), the strict interpolation RBFN method was applied to the prediction of reservoir parameters using seismic attributes. In this approach, N training points are used in the prediction, and the solution to the problem involves inverting an $N \times N$ matrix. In the second paper (Russell et. al, 2002b), the RBFN method with centres was applied to a simple AVO classification problem. In this approach, M centres are found for the N training points, where $M << N$. Although the centres used by Russell et al. (2002b) were found by trial and error, a better approach is to use a technique such as the K-means clustering algorithm. We will therefore apply the K-means clustering method to the case study shown by Russell et al. (2002a). We will finish with a suggestion as to how the Mahalanobis method can be used to improve the traditional RBFN with centres algorithm.

We will start with a brief review of the paper by Russell et al. (2002a). This study involved the prediction of porosity in the Blackfoot field of central Alberta. A 3C-3D seismic survey was recorded in October 1995, with the primary target being the Glauconitic member of the Mannville group. The reservoir occurs at a depth of around 1550 m, where valleys filled with Glauconitic sands and shales are incised into the regional Mannville stratigraphy. The objectives of the survey were to delineate the channel and distinguish between sand-fill and shale-fill. The well log input consisted of twelve wells, each with sonic, density, and calculated porosity logs, shown in Figure 11.

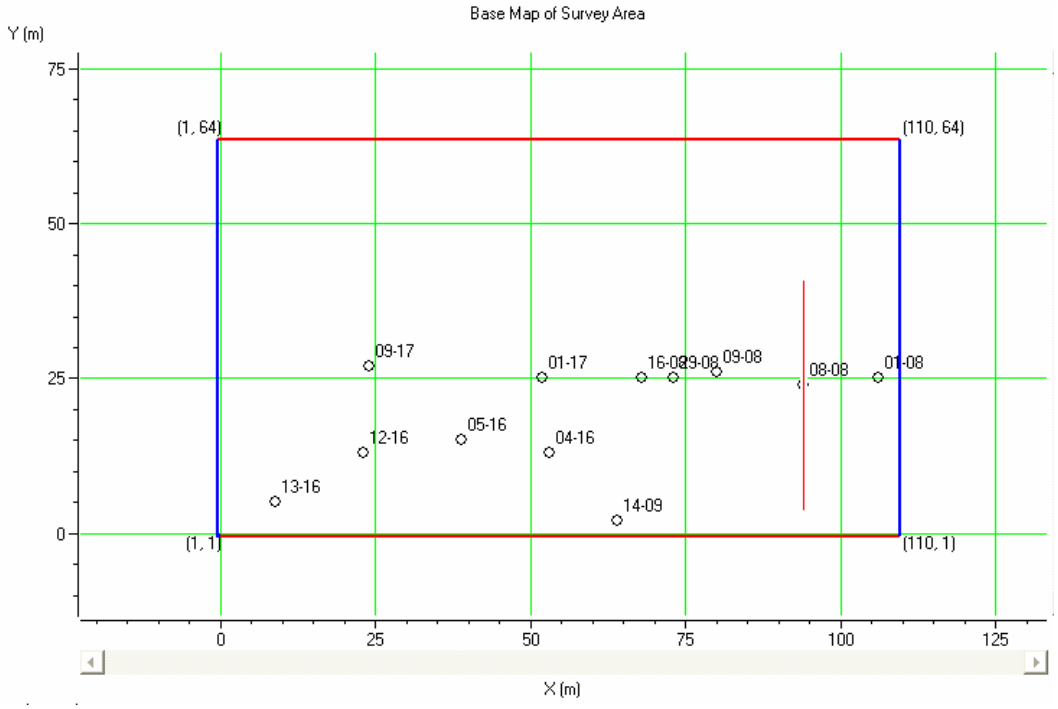


FIG. 11. The distribution of wells within a 3D seismic survey over the Blackfoot channel sand play in southern Alberta, showing seismic inline 95.

One of the lines from the 3D survey, inline 95, is indicated on Figure 11, and shown in Figure 12. Figure 12 shows the impedance section, which was inverted using a model-based inversion approach. The P-wave sonic log from well 08-08 has been inserted at cross-line 25, and shows the extra resolution present in a well log.

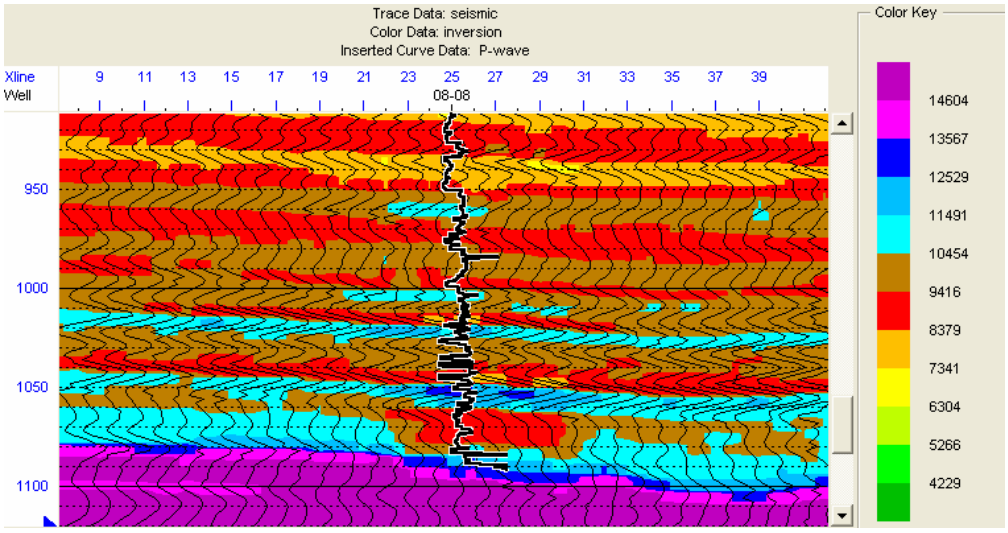


FIG. 12. Inline 95 from the 3D seismic survey of Figure 11, with the P-wave sonic from well 08-08 overlain at cross-line 25. The wiggle traces are the stacked seismic data and the colour underlay shows the impedance inversion values.

Russell et al. (2002a) used the P-wave sonic as the target on which to train the seismic attributes. Their paper was a comparison between the generalized regression neural network, or GRNN (also known in statistical estimation theory as the Nadaraya-Watson estimator) and the radial basis function neural network, or RBFN. In this paper, we will not discuss the GRNN algorithm, but will briefly review the RBFN method (Bishop, 1995). Our main objective is to use some type of neural network, in this case the RBF neural network, to perform the supervised prediction of reservoir parameters. Our training data set consists of a set of N known training samples t_i, which in our case will be some well-log derived reservoir parameter such as V_P, SP, S_W, etc. Each training sample, which is a scalar quantity, is in turn dependant on a vector of L seismic attribute values, correlated in time with the training samples. These seismic attribute vectors can be written $\boldsymbol{s}_i = (s_{i1}, s_{i2}, \ldots, s_{iL})^T$, $i = 1, 2, \ldots, N$. Thus, we can write the training samples more formally as $t(\boldsymbol{s}_i)$. The objective of a neural network is to find some function y such that

$$y(\boldsymbol{s}_i) = t_i(\boldsymbol{s}_i)\ ,\ i = 1, 2, \ldots, N. \qquad (6)$$

Obviously, the equality in equation (6) will usually not hold, so our actual objective is to find a set of y values such that we approximate the t values with the least possible error. Once this function has been found, it can be applied to an arbitrary set of M seismic attribute vectors $\boldsymbol{x}_m$, $m=1, 2, \ldots, M$, where the attributes in the $\boldsymbol{x}_m$ vectors are identical to those in the $\boldsymbol{s}_i$ vectors.

In the RBFN approach, which was originally developed as a method for performing exact interpolation of a set of data points in multi-dimensional space (Powell, 1987), the solution to equation (6) is found by solving the least-squares system given by

$$t(\boldsymbol{s}_i) = \sum_{j=1}^{N} w_j \phi_{ij}, \quad i = 1,2,\ldots,N \qquad (7)$$

where the ϕ_{ij} functions are called basis functions. Although the basis function can take many forms (Bishop, 1995), its most common form is that of the Gaussian, given by

$$\phi_{ij} = exp\left[-\frac{\left| \boldsymbol{s}_i - \boldsymbol{s}_j \right|^2}{\sigma^2} \right]. \qquad (8)$$

Notice that the term within the brackets of the basis function consists of a distance function in the numerator (the distance between each pair of attribute vectors) and a smoothing term in the denominator (similar to the variance in a Gaussian distribution). To solve equation (7), notice that it can be written as a set of N equations in N unknowns, or

$$\begin{aligned} t_1 &= w_1 \phi_{11} + w_2 \phi_{12} + \ldots + w_{1N} \\ t_2 &= w_1 \phi_{21} + w_2 \phi_{22} + \ldots + w_{2N} \\ &\vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ t_N &= w_1 \phi_{N1} + w_2 \phi_{N2} + \ldots + w_{NN} \end{aligned}. \qquad (9)$$

Equation (9) can be written more compactly as the matrix equation

$$\boldsymbol{t} = \boldsymbol{\Phi}\, \boldsymbol{w}, \qquad (10)$$

where

$$\boldsymbol{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}, \text{ and } \boldsymbol{\Phi} = \begin{bmatrix} \phi_{11} \cdots \phi_{1N} \\ \vdots \ \ddots \ \vdots \\ \phi_{N1} \cdots \phi_{NN} \end{bmatrix}.$$

The solution to equation (10) is simply the matrix inverse

$$\boldsymbol{w} = \left[\boldsymbol{\Phi} + \lambda \boldsymbol{I} \right]^{-1} \boldsymbol{t}, \qquad (11)$$

where λ is a pre-whitening factor and I is the identity matrix. This equation can be solved efficiently by noting that the matrix is symmetric. Once the weights have been computed, they are applied to the full data set using the equation

$$y(\boldsymbol{x}_m) = \sum_{j=1}^{N} w_j \, exp\left[\frac{\left| \boldsymbol{x}_m - \boldsymbol{s}_j \right|^2}{\sigma^2} \right], \quad m = 1,\ 2,\ \ldots, M \qquad (12)$$

The methodology for the preconditioning of the multiple seismic attributes has been extensively covered by Hampson et al (2001) and will only be briefly reviewed here. The

two key problems in the analysis can be summarized as follows: **which attributes should we use, and which of these attributes are statistically significant?**

To start the process, we compute as many attributes as possible. These attributes can be the classical instantaneous attributes, frequency and absorption attributes based on windowed estimates along the seismic trace, integrated seismic amplitudes, model-based trace inversion, or AVO intercept and gradient. Once these attributes have been computed and stored, we measure a goodness of fit between the attributes and the training samples from the logs. Using the notation of the previous section, we can compute this goodness of fit by minimizing the least-squares error given by

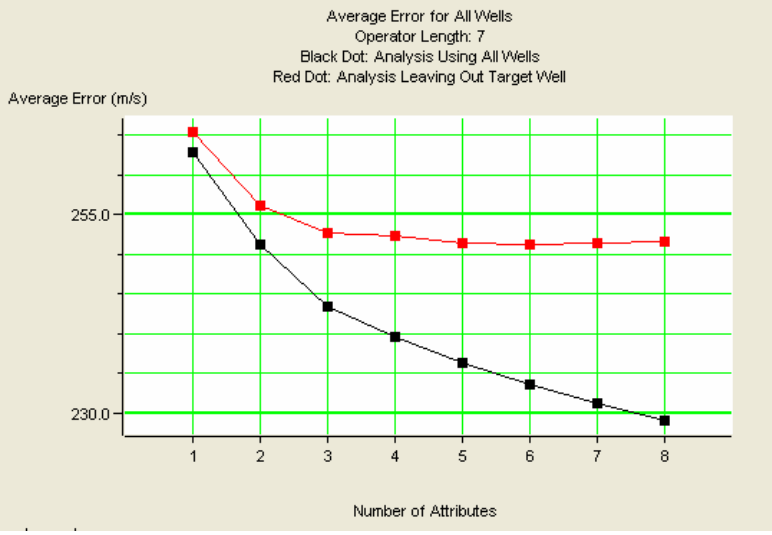
$$E^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - w_0 - w_1 s_{1i} - \ldots - w_L s_{Li})^2 \quad . \tag{13}$$

Note that the weights can also be convolutional, as discussed by Hampson et al. (2001), which is equivalent to introducing a new set of time-shifted attributes. The attributes to use in the neural network computations, and their order is then found by a technique called stepwise regression, which consists of the following steps:

1. Find the best attribute by an exhaustive search of all the attributes, using equation (1) to compute the prediction error for each attribute (i.e. $L = 1$) and choosing the attribute with the lowest error.
2. Find the best pair of attributes from all combinations of the first attribute and one other. Again, the best pair is the pair that has the lowest prediction error from equation (18), with $L = 2$.
3. Find the best triplet, using the pair from step (2) and combining it with each other attribute.
4. Continue the process as long as desired.

This will give us a set of attributes that is guaranteed to reduce the total error as the number of attributes goes up. To find the significant attributes, we use a technique called cross-validation, in which we leave out a training sample and then predict it from the other samples. We then re-compute the error using equation (13), but this time from the training sample that was left out. We repeat this procedure for all the training samples, and average the error, giving us a total validation error. This computation is done as a function of the number of attributes, and the resulting graph usually shows an increase in validation error past some small number of attributes such as five or six.

Figure 13(a) shows this initial step applied to the data of Figure 12, where we have used all twelve wells in the training, a maximum of eight attributes, and a seven point convolutional operator. Notice from Figure 13(a) that the validation curve (the top curve, in red, shows that only the first six attributes actually reduce the error. This is confirmed in Figure 13(b), which shows the actual attributes, and the numerical values of the validation process.



(a)

	Target	Final Attribute	Training Error	Validation Error
1	Sqrt(P-wave)	1 / (Impedance)	262.795094	265.408161
2	Sqrt(P-wave)	Filter 15/20-25/30	251.229374	256.100616
3	Sqrt(P-wave)	Filter 35/40-45/50	243.316259	252.696652
4	Sqrt(P-wave)	Filter 55/60-65/70	239.606546	252.268152
5	Sqrt(P-wave)	Amplitude Weighted Phase	236.356838	251.459963
6	Sqrt(P-wave)	Amplitude Weighted Frequency	233.497895	251.186200
7	Sqrt(P-wave)	Integrate	231.205176	251.451810
8	Sqrt(P-wave)	Second Derivative	229.012163	251.506805

(b)

FIG. 13. Cross-validation results for the 12 wells shown in Figure 11, where (a) shows the cross-validation curve (top, in red), and (b) shows the attributes and numerical results.

Once the type and order of the attributes have been determined, we can train the RBFN algorithm. This is shown in Figure 14, where only the first three wells are displayed. The black lines show the original curves and the red lines show the predicted curves. Notice that the correlation coefficient is 0.7994, and that the observed fit is very good.

The cross-validation of the RBFN result is shown in Figure 15, where the each log has been left out, in turn, of the training. The black lines show the original curves and the red lines show the predicted curves. Only the first three P-wave sonic logs have been shown, although all twelve were used in the training. Notice that the cross-validation error is 0.6577, which is lower that the training error without cross-validation, but still quite good.

The results of the training are then applied to the seismic line shown in Figure 12, and is shown in Figure 16. Notice the excellent fit of the predicted P-wave sonic log at the well tie, and also the high frequency detail of the results as we move away from the well.

There is also good lateral continuity of the predicted events. This detail should be compared with the result shown in Figure 12.

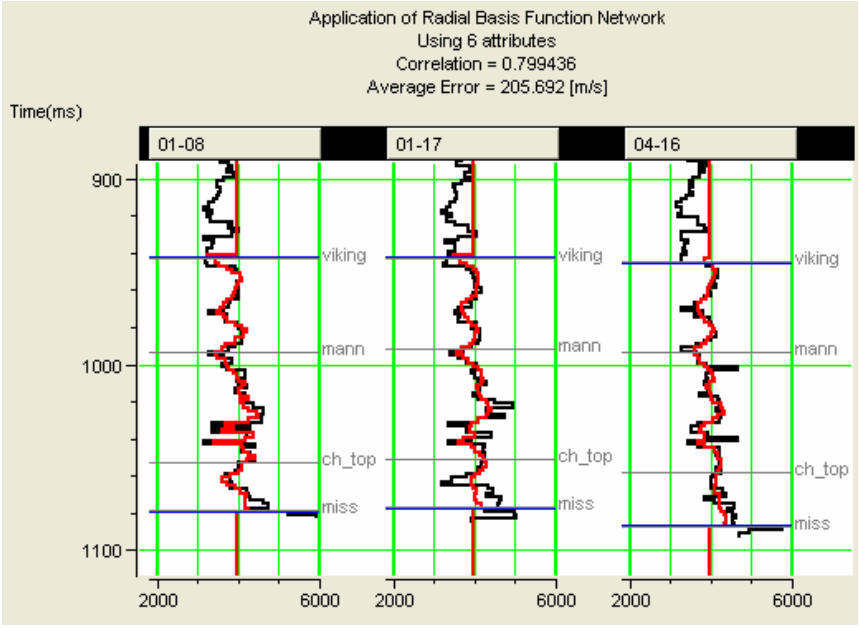


FIG. 14. Training the RBFN algorithm, where all the training samples are used in the prediction. The black lines show the original curves and the red lines show the predicted curves. Only the first three P-wave sonic logs have been shown, although all twelve were used in the training.

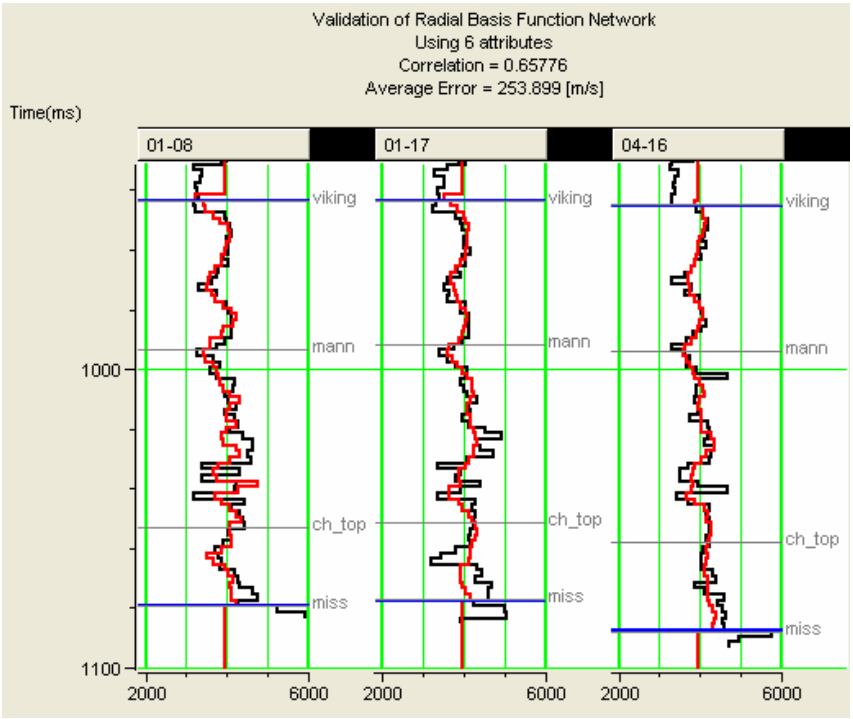


FIG. 15. Validation of the RBFN, where the each log has been left out, in turn, of the training. The black lines show the original curves and the red lines show the predicted curves. Only the first three P-wave sonic logs have been shown, although all twelve were used in the training.

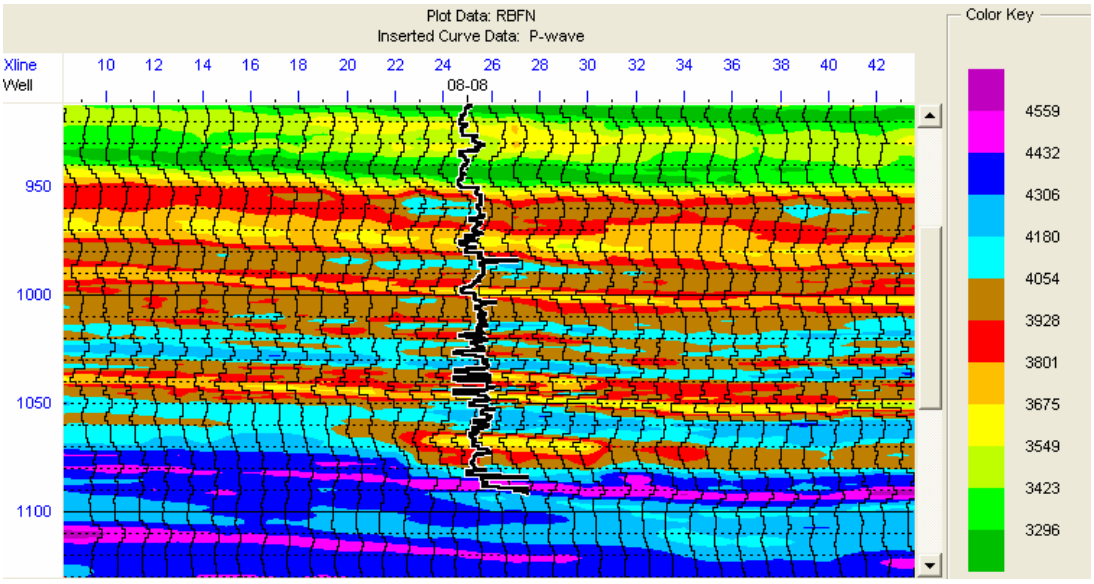


FIG. 16. Application of the full RBFN algorithm to line 95 of the 3D volume.

Having discussed the exact interpolation RBFN, which uses all the training points in the calculation of the weights, we will now discuss a modification to the method which uses a reduced set of weighting functions. There are two reasons to modify the RBFN method in this way. The first is that we may reach a point at which we have so many training points that the solution of an $N \times N$ matrix becomes prohibitively expensive and slow, even when optimization methods such as conjugate-gradient are used. The second is that, in the case of noisy and inconsistent training data, the solution of the $N \times N$ set of linear equations may produce an unstable result. Indeed, for the result just shown in Figures 14 through 16, the prewhitening factor of equation (11) had to be set to 10% to avoid instability in the matrix inversion.

The solution to this problem is to define a modified version of equation (7) given by

$$t(\boldsymbol{s}_i) = \sum_{k=1}^{K} w_k \exp\left[-\frac{|\boldsymbol{s}_i - \boldsymbol{\mu}_k|^2}{\sigma^2}\right], \quad i = 1,2,\ldots,N \qquad (14)$$

where the μk terms are K cluster centres, or means, and $K << N$. Notice that these are vector valued means, with a dimension equal to L, the number of attributes. Equation (14) can be written as the set of linear equations given by

$$\begin{aligned} t_1 &= w_1 \phi_{11} + w_2 \phi_{12} + \ldots + w_{1K} \\ t_2 &= w_1 \phi_{21} + w_2 \phi_{22} + \ldots + w_{2K} \\ &\vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ t_N &= w_1 \phi_{N1} + w_2 \phi_{N2} + \ldots + w_{NK} \end{aligned} \qquad (15)$$

Since the set of equations given in equations (14) and (15) represent an over-determined system, the solution is the least-squares Moore-Penrose inverse given by

$$\boldsymbol{w} = \left[\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda I\right]^{-1} \boldsymbol{\Phi}^T \boldsymbol{t}, \tag{16}$$

where

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}, \boldsymbol{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}, \text{ and } \boldsymbol{\Phi} = \begin{bmatrix} \phi_{11} \cdots \phi_{1N} \\ \vdots \ \ddots \ \vdots \\ \phi_{N1} \cdots \phi_{NN} \end{bmatrix}.$$

Next, we will apply the RBFN method with centres to the problem we have just considered. It should first be pointed out that in the training of the full RBFN algorithm, we used a total training data set of approximately 1000 points. We will reduce this number of points to 25 centres and see how well the results compare to the full RBFN. The same attributes will be used as shown in Figure 13. These centres are found using the *K*-means clustering algorithm. Since the vectors are 6-dimensional, it will be impossible to actually visualize the clusters.

Also note that the scaling functions, σ, given in equation (14) are constants. These are optimized by using a range of σ values and choosing the one that gives the lowest cross-validation error. An alternate approach to the optimization of these values will be given in the last section.

This result of the training with 25 centres is shown in Figure 17, where only the first three wells are displayed. The black lines show the original curves and the red lines show the predicted curves. Notice that the correlation coefficient is 0.634, less than the full RBFN value of 0.7994, but still a very good fit.

The cross-validation of the RBFN result with 25 centres is shown in Figure 18, where the each log has been left out, in turn, of the training. The black lines show the original curves and the red lines show the predicted curves. Only the first three P-wave sonic logs have been shown, although all twelve were used in the training. Notice that the cross-validation error is 0.602, which is less than the full RBFN result of 0.6558, but actually much closer than the full training result. That is, the difference between the training result with all wells and the cross-validation result is much smaller for RBFN with 25 centres than for full RBFN.

The results of the training are then applied to the seismic line shown in Figure 12, and is shown in Figure 19. Notice the excellent fit of the predicted P-wave sonic log at the well tie, and also the high frequency detail of the results as we move away from the well. There is also good lateral continuity of the predicted events. This detail should be compared with the results shown in Figures 12 and 16. Although the full result of Figure 16 is better, we have achieved almost as good a result using roughly 2.5 % of the original number of data values.

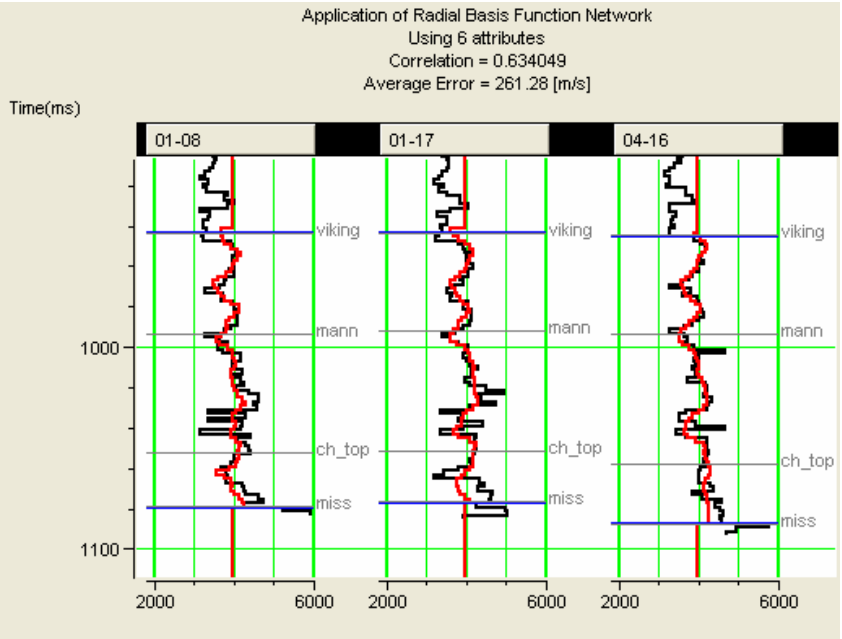


FIG. 17. Training the RBFN algorithm using 25 centres. The black lines show the original curves and the red lines show the predicted curves. Only the first three P-wave sonic logs have been shown, although all twelve were used in the training.

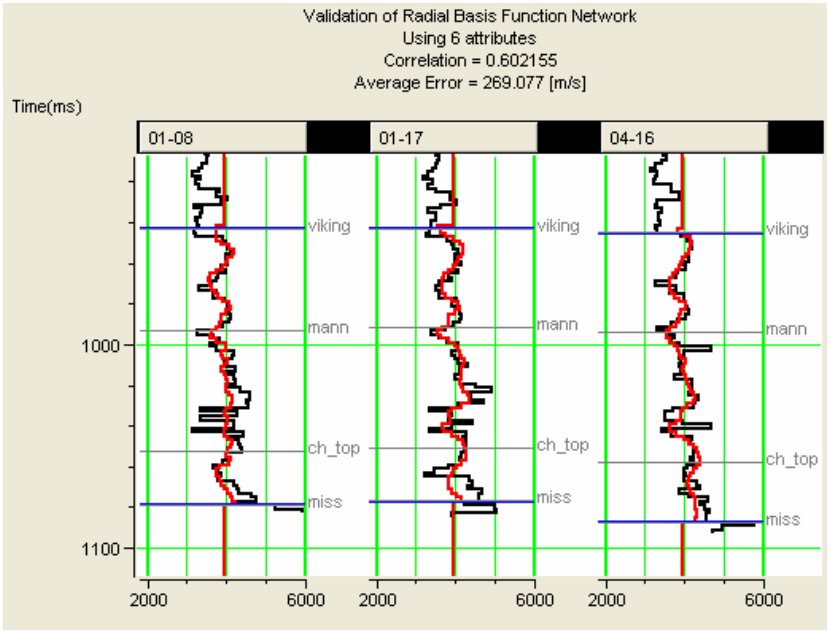


FIG. 18. Validation of the RBFN using 25 centres, where the each log has been left out, in turn, of the training. The black lines show the original curves and the red lines show the predicted curves. Only the first three P-wave sonic logs have been shown, although all twelve were used in the training.

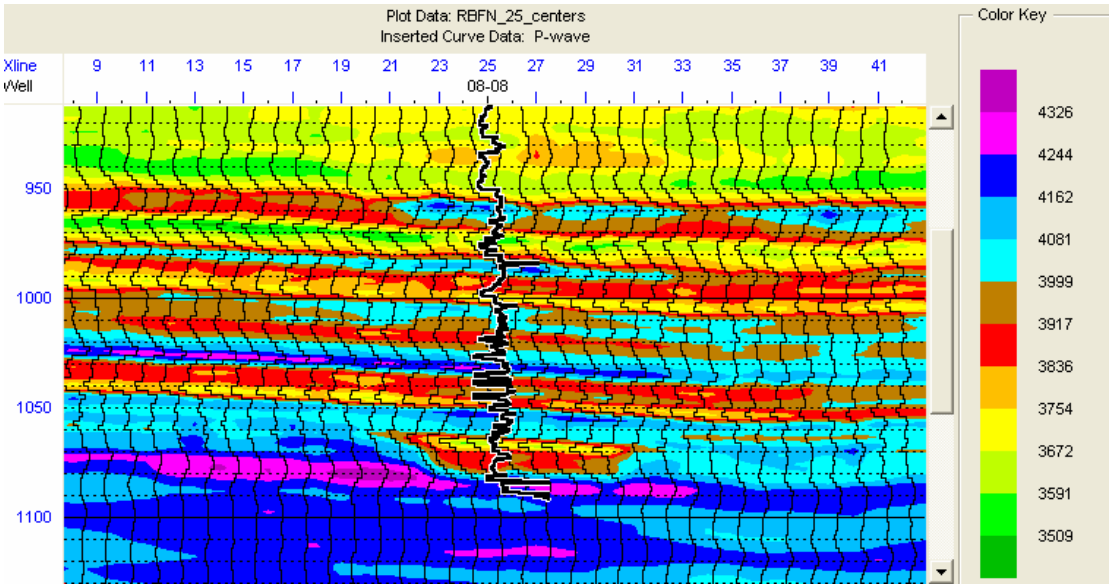


FIG. 19. Application of the RBFN with 25 centres to line 95 of the 3D volume.

RBFN APPLICATION USING MAHALANOBIS CLUSTERING

In the previous section, we showed how *K*-means clustering could be used to improve the run time and stability of the RBFN algorithm by reducing the full interpolation algorithm to an algorithm that uses a limited number of centres (or mean vectors) in the design of the weight values. You will also recall that the scaling functions, σ, given in equation (14) were constant and were optimized by using a range of σ values and choosing the one that gives the lowest cross-validation error. If we return to the concept of statistical distance, notice that the full basis function equation can actually be written as follows (Bishop, 1995), using statistical distance rather than Euclidean distance:

$$\phi_{ik} = exp\left[-\frac{1}{2}(\boldsymbol{s}_i - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{s}_i - \boldsymbol{\mu}_k)\right], \tag{17}$$

where

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \sigma_{11}^{(k)} & \sigma_{12}^{(k)} & \cdots & \sigma_{1L}^{(k)} \\ \sigma_{21}^{(k)} & \sigma_{22}^{(k)} & \cdots & \sigma_{2L}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{L1}^{(k)} & \sigma_{L2}^{(k)} & \cdots & \sigma_{LL}^{(K)} \end{bmatrix}.$$

Equation (17) can be thought of as analogous to the multivariate Gaussian distribution (Johnson and Wichern, 1998). The Σ_k matrix represents the covariance matrix for the k^{th} cluster. The basis function in equation (14) can thus be seen to be a simplification of equation (17), in which the covariance matrix can be written in the simplified form,

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix} = \sigma^2 I \quad . \tag{18}$$

The reason that this form of the RBFN is not used is that we normally do not have an estimate of the full covariance matrix for each cluster. However, using the Mahalanobis clustering method proposed in the previous section, we do obtain an estimate of each of these covariance matrices. Although this procedure has not yet been implemented, we are in the process of updating the source code, and intend to test this idea on real and model data.

CONCLUSIONS

In this paper, we have presented a new approach to clustering, which we call Mahalanobis clustering. This method is an extension of the *K*-means clustering method in which we apply a second iteration which uses statistical, or Mahalanobis, distance to perform the clustering.

After a review of the *K*-means method, in which we illustrated the approach using a simple two-dimensional problem, we then presented a model data set for which the *K*-means method did not converge to a correct answer. This model data set consisted of elongated elliptical clusters. We then described the Mahalanobis clustering method and showed that this method converged to a correct answer for our model data set.

We then applied the *K*-means and Mahalanobis clustering methods to a two-dimensional AVO crossplot example. Although both methods converged to the same answer, we showed that the Mahalanobis method was able to give us the correct elliptical shapes for interpreting the AVO crossplot. This was confirmed by the interpretation of the crossplot, in which the top and base of a gas zone were identified.

We then used *K*-means clustering to find the clusters to be used in the radial basis function neural network (RBFN) method with centres, and found that we could achieve a result that was very close to the full RBFN method using clusters representing only 3% of the original number of points. Although the *K*-means method performed well, it gave us no way of finding and refining the values for the covariance matrices of each cluster, which would allow us to optimize the scaling parameters in the weight determination. In the last section, we proposed a method which uses Mahalanobis clustering to perform this task. This method has not yet been implemented, but will be applied in the near future.

REFERENCES

Bishop, C. M., 1995, Neural Networks for Pattern Recognition: Oxford University Press.

Chen, S., Cowan, C. F. N., and Grant, P. M., 1991, Orthogonal least squares learning algorithm for radial basis function neural networks: IEEE Transactions on Neural Networks, **2**, 302-309.

Hampson, D. P., Schuelke, J. S., and Quirein, J. A., 2001, Use of multiattribute transforms to predict log properties from seismic data: Geophysics, **66**, 220-236.

Haykin, S. S., 1998, Neural Networks: A Comprehensive Foundation, 2nd Edition: Macmillan Publishing Company.

Johnson, R. A. and Wichern D. W., 1998, Applied Multivariate Statistical Analysis, 4th Edition: Prentice Hall.

Powell, M. J. D., 1987, Radial basis functions for multivariable interpolation: a review, *in* Mason, J. C., and Cox, M. G. Eds., Algorithms for Approximation: Oxford Clarendon Press, 143-167.

Ross, C. P., 2000, Effective AVO crossplot modeling: A tutorial: Geophysics, **65**, 700-711.

Russell, B. H, Lines, L. R., and Hampson, D.P., 2002a, Application of the radial basis function neural network to the prediction of log properties from seismic attributes: CREWES Research Report, **14**.

Russell, B. H, Lines, L. R., and Ross, C. P., 2002b, AVO classification using neural networks: A comparison of two methods: CREWES Research Report, **14**.

ACKNOWLEDGEMENTS

We wish to thank our colleagues at the CREWES Project and at Hampson-Russell Software for their support and ideas, as well as the sponsors of the CREWES Project.