Unsupervised seismic facies classification using *K*-means and Gaussian Mixture Modeling

Brian Russell[1]

[1] HampsonRussell, A CGG GeoSoftware Company, Calgary, Alberta, brian.russell@cgg.com

ABSTRACT

In this study, I will apply the techniques of *K*-means clustering and Gaussian Mixture Modeling (GMM) to the task of multidimensional unsupervised seismic facies classification. The *K*-means algorithm is the most popular and well understood clustering algorithm. For *N M*-dimensional points, *K*-means is implemented by dividing the input data points randomly into *K* clusters, computing the *M*-dimensional means of the clusters, and then assigning each point to the cluster for which its distance to the mean is a minimum. The means are then re-computed based on the new cluster assignments and this process is iterated until convergence. Traditionally, the distance metric used in *K*-means is Euclidean, but it can be modified to use a Mahalanobis, or statistical, distance. The Gaussian Mixture Model (GMM) is a mixture pdf of *N M*-dimensional feature vectors which are grouped into *K* classes. GMM starts with an initial guess of the means and covariance matrices of each class and determines the correct values by iterating to a solution. Therefore, GMM always uses statistical distance as its metric. Also, unlike the *K*-means algorithm, the data is never physically re-ordered during the process. I will start by illustrating the *K*-means and GMM methods with several two-dimensional synthetic clustering examples and a two-dimensional real data example that uses inverted elastic parameters (V_P/V_S ratio versus acoustic impedance) extracted from a Gulf of Mexico dataset. I will then move to a higher dimensional example of *K*-means clustering and GMM that performs unsupervised facies classification of a Cretaceous channel sand play in the Blackfoot area of Alberta.

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 dataset. 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, a related technique, is supervised by the knowledge of what our classes should look like, whereas clustering is unsupervised, without any such knowledge.

I will start by considering the simplest clustering approach, called *K*-means clustering (Bishop, 2006). This method is based on the Euclidean distance between points. I will show that this method works well for the case of well-separated, roughly circular clusters, but not when the clusters become elliptical in shape. I will then describe a method called Mahalanobis clustering, in which statistical distance, rather than Euclidean distance, is used for the clustering.

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. If we start with *N M*-dimensional data points (e.g., *M* attributes) the *K*-means algorithm divides these points into *K* clusters. The *K*-means algorithm is implemented as follow:

1. Pick the number of clusters, *K*, and divide the input data points randomly into these *K* clusters, each with approximately *N/K* points.
2. Compute the *M*-dimensional means of the clusters.
3. Compute the distances between each point and each cluster and assign each point to the cluster for which this distance is a minimum.
4. Re-compute the means based on the new cluster assignments.
5. Iterate through the above three steps until convergence.

Mathematically, the initial means are computed in the following way for each cluster:

$$\boldsymbol{\mu}_k = \frac{1}{N}\sum_{i=1}^{N_k} \boldsymbol{x}_i \,, \tag{1}$$

where $\boldsymbol{\mu}_k$ are the *M*-dimensional means, x_i are *M*-dimensional input points, and N_k is the number of points in the k^{th} cluster. Most *K*-means clustering algorithms perform the ordering using Euclidean distance, which can be written

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

where

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

In this section, I will use Euclidean distance as our distance metric, but in the next section I will discuss a method that uses a different distance metric, statistical, or Mahalanobis, distance.

Obviously, the *K*-means clustering algorithm can be computed for any number of input points, attributes and clusters. Before considering a real data application, let us consider the following numerical example in which we have the eighteen input points given by: $\boldsymbol{x}_1 = \begin{bmatrix}1\\1\end{bmatrix}$, $\boldsymbol{x}_2 = \begin{bmatrix}8\\1\end{bmatrix}$, $\boldsymbol{x}_3 = \begin{bmatrix}4\\9\end{bmatrix}$, $\boldsymbol{x}_4 = \begin{bmatrix}2\\3\end{bmatrix}$, $\boldsymbol{x}_5 = \begin{bmatrix}8\\2\end{bmatrix}$, $\boldsymbol{x}_6 = \begin{bmatrix}5\\5\end{bmatrix}$, $\boldsymbol{x}_7 = \begin{bmatrix}3\\2\end{bmatrix}$, $\boldsymbol{x}_8 = \begin{bmatrix}9\\3\end{bmatrix}$, $\boldsymbol{x}_9 = \begin{bmatrix}6\\6\end{bmatrix}$, $\boldsymbol{x}_{10} = \begin{bmatrix}1\\6\end{bmatrix}$, $\boldsymbol{x}_{11} = \begin{bmatrix}9\\1\end{bmatrix}$, $\boldsymbol{x}_{12} = \begin{bmatrix}7\\7\end{bmatrix}$, $\boldsymbol{x}_{13} = \begin{bmatrix}2\\8\end{bmatrix}$, $\boldsymbol{x}_{14} = \begin{bmatrix}10\\2\end{bmatrix}$, $\boldsymbol{x}_{15} = \begin{bmatrix}7\\4\end{bmatrix}$, $\boldsymbol{x}_{16} = \begin{bmatrix}3\\7\end{bmatrix}$, $\boldsymbol{x}_{17} = \begin{bmatrix}10\\3\end{bmatrix}$, $\boldsymbol{x}_{18} = \begin{bmatrix}8\\5\end{bmatrix}$.

Thus, $M = 2$ in our example and we can plot the results in two dimensions. These eighteen points 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.

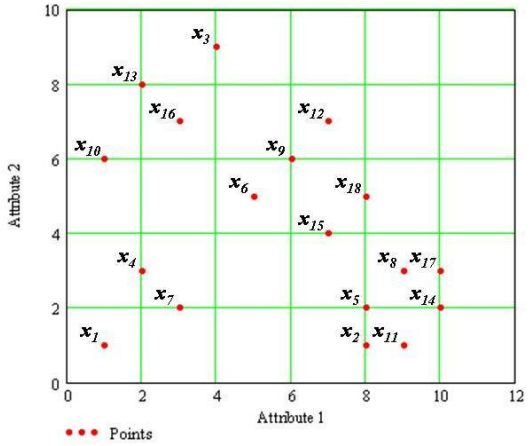


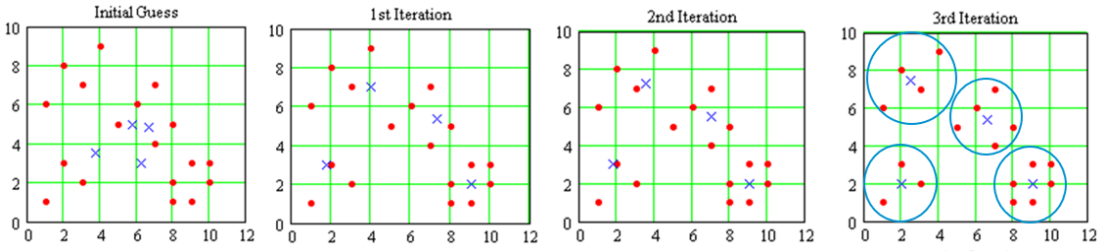
Figure 1: The red dots show a set of eighteen points, grouped in four clusters, with the labels indicating the input order of the two-dimensional attribute vectors.

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

The initial means are

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 3.75 \\ 3.5 \end{bmatrix}, \boldsymbol{\mu}_2 = \begin{bmatrix} 6.25 \\ 3 \end{bmatrix}, \boldsymbol{\mu}_3 = \begin{bmatrix} 5.75 \\ 5 \end{bmatrix}, \text{ and } \boldsymbol{\mu}_4 = \begin{bmatrix} 6.667 \\ 4.833 \end{bmatrix}.$$

The means of the four clusters are plotted using blue crosses on Figure 2(a). Notice that all 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. As the number of points increases, and they continue to be random, the initial means should be grouped together near the total population mean.



(a) (b) (c) (d)

Figure 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 means, (b) shows the result of the first iteration, (c) shows the result of the second iteration, and (d) shows the result of the third iteration, with circles indicating the Euclidean distance. After the three iterations, the algorithm has converged perfectly to the right answer.

After the first iteration, we find that the means become

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 1.75 \\ 3 \end{bmatrix}, \boldsymbol{\mu}_2 = \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \boldsymbol{\mu}_3 = \begin{bmatrix} 4 \\ 7 \end{bmatrix}, \text{and } \boldsymbol{\mu}_4 = \begin{bmatrix} 7.333 \\ 5.333 \end{bmatrix},$$

which is shown in Figure 2(b). Since the algorithm has not found the solution yet, we will go to a second iteration, which gives the means

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 1.75 \\ 3 \end{bmatrix}, \boldsymbol{\mu}_2 = \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \boldsymbol{\mu}_3 = \begin{bmatrix} 3.5 \\ 7.55 \end{bmatrix}, \text{and } \boldsymbol{\mu}_4 = \begin{bmatrix} 7 \\ 5.5 \end{bmatrix}.$$

The results of the second iteration are shown in Figure 2(c), and the solution is much closer. Finally, we go to a third iteration, which leads to the correct means of:

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

which is shown in Figure 2(d). In this case, we have converged to the correct answer in three iterations.

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

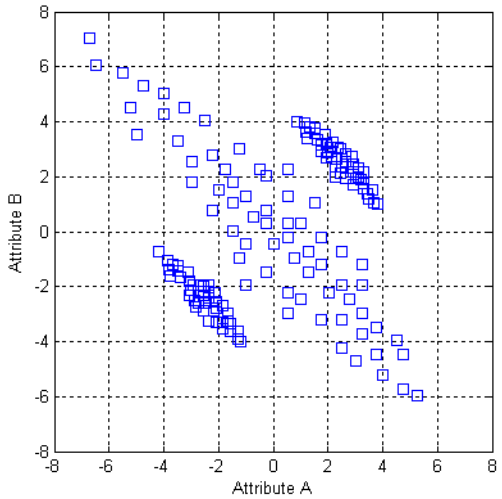
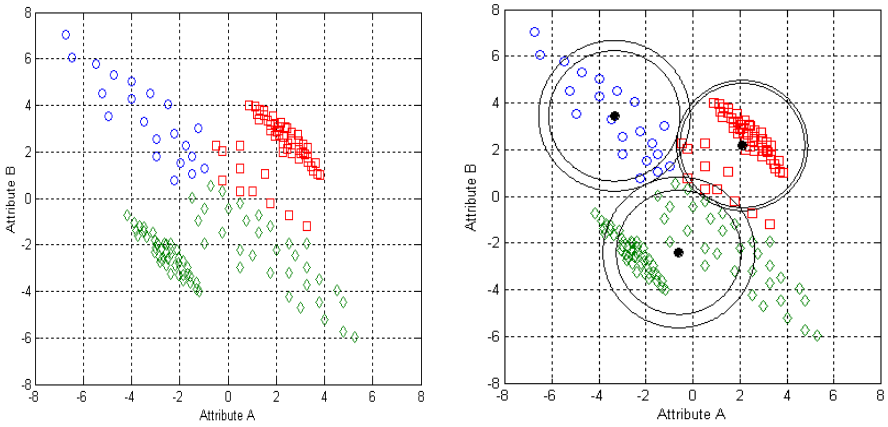


Figure 3: A second input dataset to the K-means clustering algorithm. This dataset simulates a typical AVO A-B crossplot.

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. In the next section, I will consider a more generalized approach to *K*-means clustering which will address this problem.



(a) (b)

Figure 4: The application of the *K*-means clustering algorithm to the input dataset 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 *K*-MEANS CLUSTERING

In the previous section, the Euclidean distance shown in equation 2 was used as a distance metric in the K-means algorithm. In this section, following Russell and Lines (2003), we will use the statistical, or Mahalanobis, distance as our metric. Mahalanobis distance from point $\boldsymbol{x_i}$ to cluster mean $\boldsymbol{\mu_k}$, which can be written as (Johnson and Wichern, 1998):

$$\Delta_{ik}^2 = (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)\,, \tag{3}$$

where $\boldsymbol{\Sigma}_k$ is the covariance matrix of the k^{th} cluster given by

$$\Sigma_k = \frac{1}{N} \sum_{i=1}^{N_k} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T\,. \tag{4}$$

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

$$p(\boldsymbol{x}_i) = \frac{1}{(2\pi)^{M/2} \left|\Sigma_k\right|^{1/2}} \exp\left[-\frac{\Delta_{ik}^2}{2}\right]. \tag{5}$$

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

That is, if

$$\Sigma_k = \Sigma_k^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{6}$$

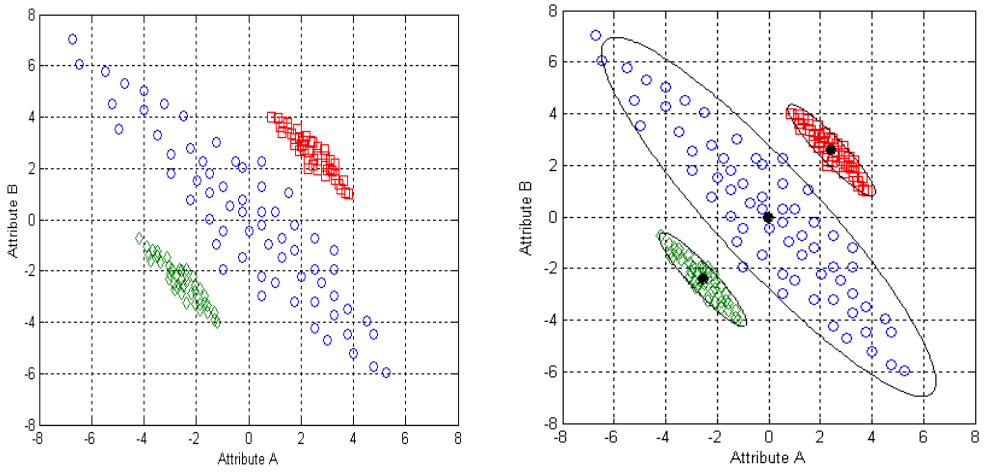
then

$$\Delta_{ik}^2 = (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) = (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T (\boldsymbol{x}_i - \boldsymbol{\mu}_k), \tag{7}$$

which is the Euclidean distance. Thus, the Mahalanobis distance is the generalization of Euclidean distance and can be computed for each cluster if the covariances of the cluster are known. This suggests a modification of the *K*-means algorithm as follows:

1. Pick the number of clusters, *K*, and divide the input data points randomly into these *K* clusters.
2. Compute the *M*-dimensional means, $\boldsymbol{\mu}_k$, of the clusters, as well as the covariance matrices Σ_k within each cluster, where $k = 1, 2, \ldots, K$.
3. Compute the Mahalanobis distances between each point and cluster and assign each point to the cluster for which this distance is a minimum.
4. Re-compute the means and covariances based on the new cluster assignments.
5. Iterate through the above three steps until convergence.

Now, let's see how well this algorithm works on our dataset. The result of applying 20 iterations of Mahalanobis *K*-means 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.



(b) (b)

Figure 5: The application of the Mahalanobis clustering algorithm to the input dataset 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.

Now, let's move to a real data example. Figure 6 shows a plot of one inline from the pre-stack inversion of a Gulf Coast dataset, which is intersected by a well that found gas at a time of 2550 ms. The traces represent acoustic impedance (I_P) and the colour represents V_P/V_S ratio (the green is low V_P/V_S at the gas). Three zones have been picked on the line, from the gas sand, shallow wet sands and shales and deeper carbonates.

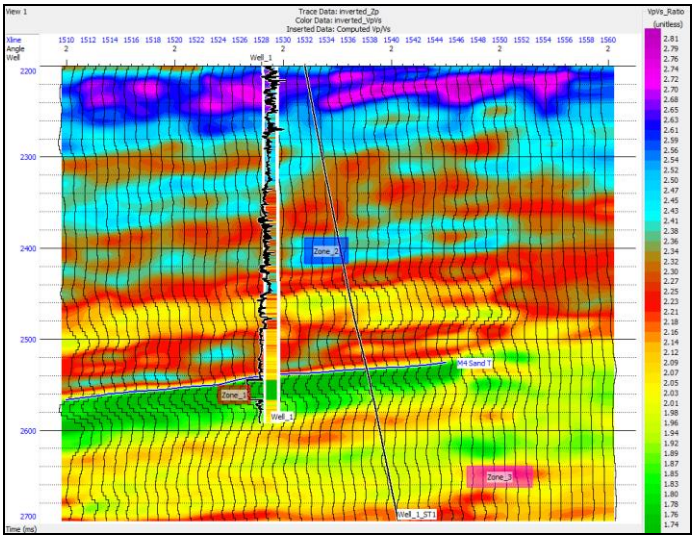


Figure 6: A plot of one line from a pre-stack inverted volume from the Gulf of Mexico, where color represents Vp/Vs ratio and wiggle trace represent P-impedance and three zones have been picked: Zone 1 in a gas sand,, Zone 2 in a wet sand-shale area and Zone 3 in a carbonate.

Figure 7 is a cross-plot of the V_P/V_S ratio versus I_P values of the three zones. The blue squares are from the shallow, wet sands and shales, the red circles are from the gas sand and the cyan triangles are from the deeper carbonates. There are 97 points in total: 35 blue (upper left), 20 red (low Vp/Vs) and 42 cyan (middle right). Let's now see how *K*-means will handle these clusters.

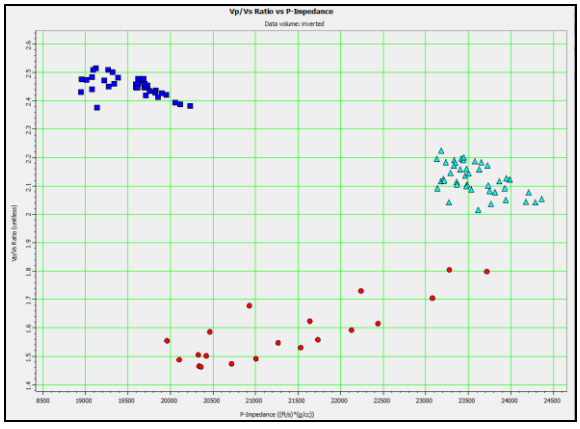


Figure 7: A cross-plot of the three zones from Figure 6, with Vp/Vs on the vertical axis and P-impedance on the horizontal axis, where the red points show Zone 1, the blue points show Zone 2 and the cyan points show Zone 3.

We will present these points to the *K*-means algorithm in an unsupervised way, as shown in Figure 8. As human interpreters, we have no problem seeing the three distinct clusters.

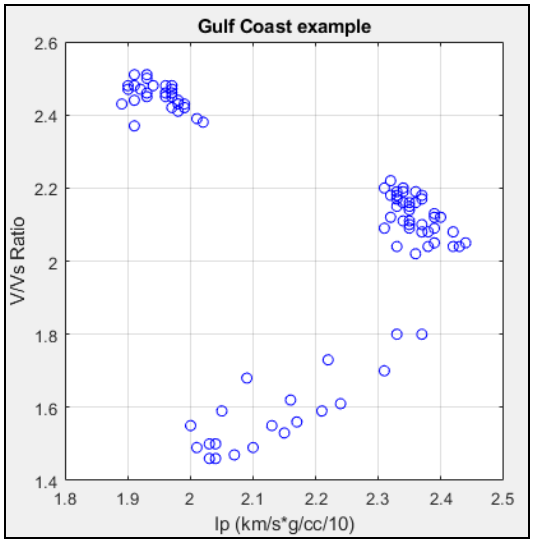
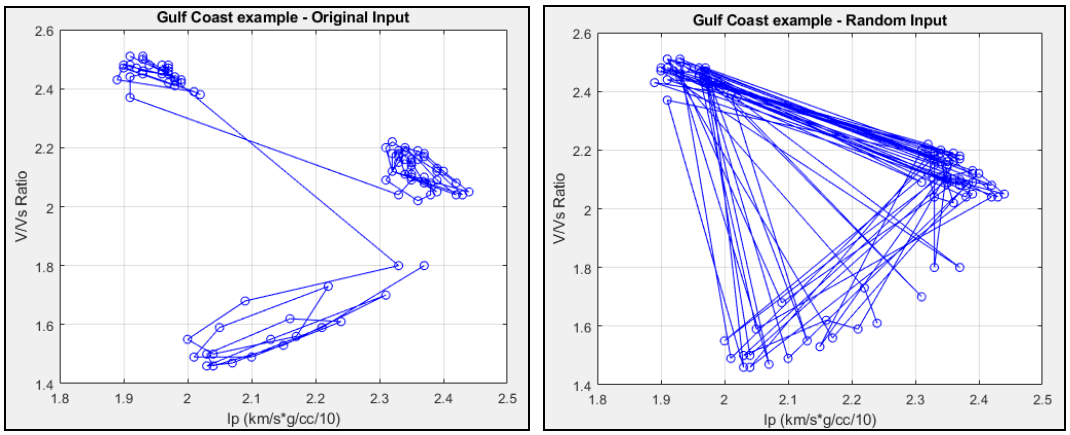


Figure 8: The points from Figure 7 as they are presented to an unsupervised network.

However, a computer algorithm will not "see" the clusters immediately. In fact, what a computer "sees" depends on the input order of the points. Figure 9(a) shows the connections between the points if we present the points to the algorithm as they were picked. Figure 9(b) shows the points presented in random order to make it more difficult for the computer to find the clusters.



(a) (b)

Figure 9: (a) Shows the original order of the data points, and (b) shows the randomized order.

If initiate the *K*-means algorithm by letting the number of clusters $K = 3$, the algorithm starts by dividing the inputs into three approximately equal groups (in this case: 32, 32, 33 points, respectively). Because of the random order, the initial guess of the clusters is not very good, as shown in Figure 10. The algorithm then computes

the means of the three clusters, which are shown by the blue, red and cyan crosses. As expected, they are all close to the centre of the cross-plot.

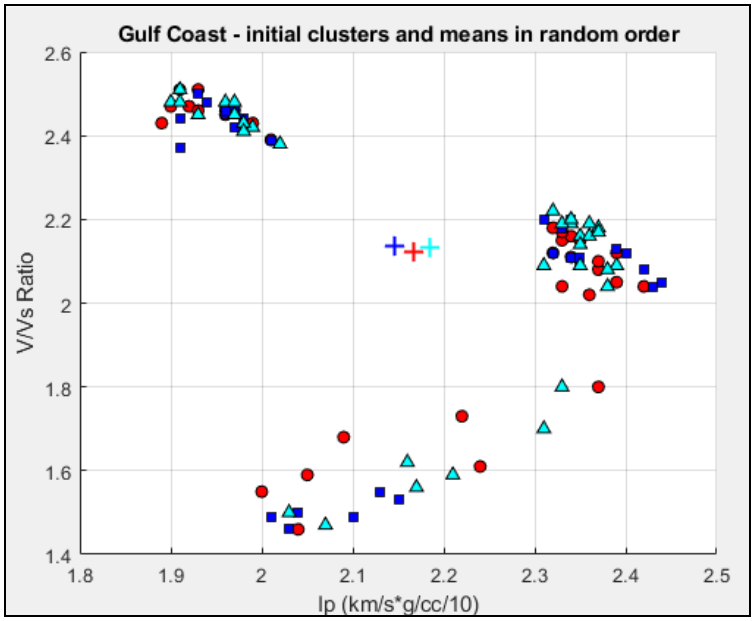


Figure 10: The initial estimate of the means and clusters from K-means.

The algorithm then assigns the points to their closest means, re-groups them and re-computes the means. This procedure is iterated until convergence, which is shown in Figure 11, with the means as black crosses. Because of the wide separation of these clusters, the algorithm converges in one iteration. However, notice that one point on the gas cluster has been miss-classified. This is an example of a limitation in the standard *K*-means algorithm, which can be corrected with Mahalanobis *K*-means.

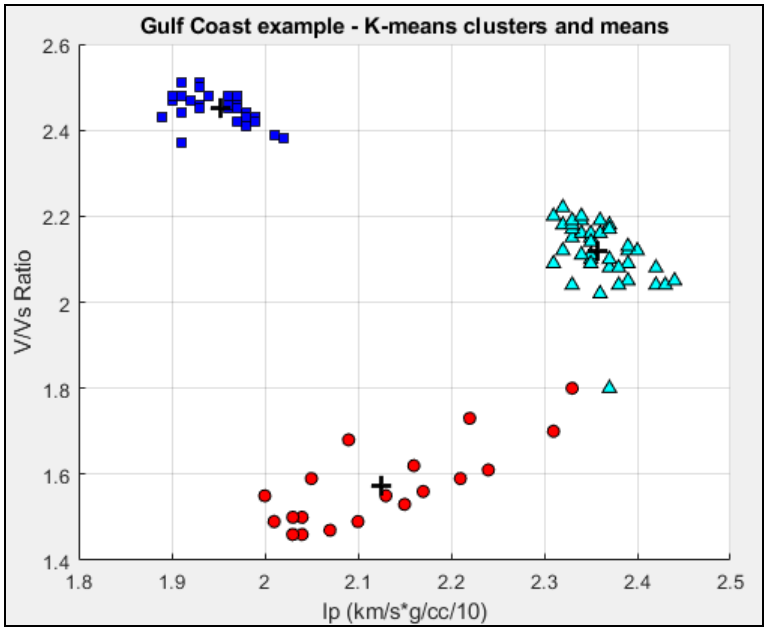


Figure 11: The final estimate of the means and cluster points from *K*-means, where one point has been miss-classified.

I therefore applied Mahalanobis *K*-means to this dataset, and the result is shown in Figure 12, showing the elliptical contours for the first 3 standard deviations, where the red points show cluster 1, the blue points show cluster 2 and the cyan points show cluster 3.

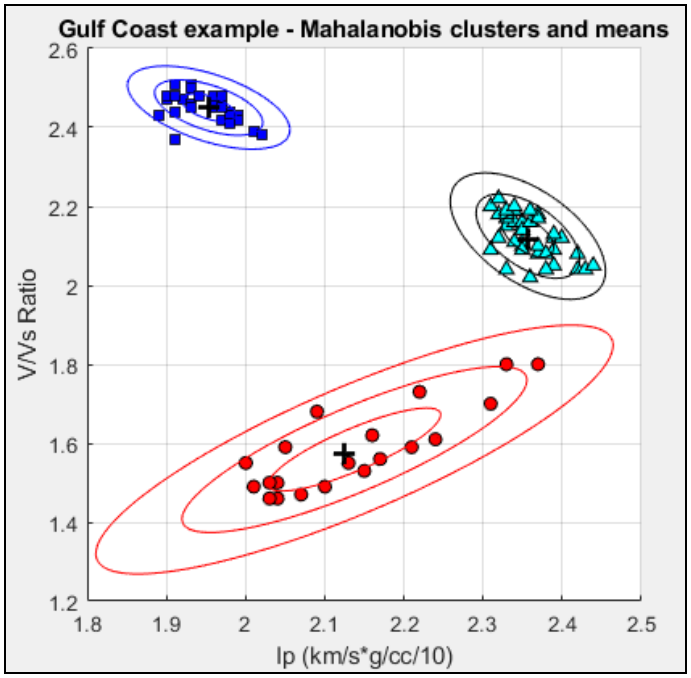


Figure 12: The final estimate of the means and cluster points from Mahalanobis *K*-means, showing the bivariate Gaussian contours, where all the points have been correctly classified.

Here are the statistics for the three clusters:

$$\mu_1 = \begin{bmatrix} 2.1375 \\ 1.5840 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 0.0119 & 0.0096 \\ 0.0096 & 0.0110 \end{bmatrix}, \mu_2 = \begin{bmatrix} 1.9529 \\ 2.4503 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.0012 & -0.0006 \\ -0.0006 & 0.0012 \end{bmatrix},$$

and $\mu_3 = \begin{bmatrix} 2.3571 \\ 2.1248 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 0.0011 & -0.001 \\ -0.001 & 0.0029 \end{bmatrix}.$

Note that Mahalanobis *K*-means is a term used by Russell and Lines (2003) to describe this process, and we were unaware of any other references to this technique at the time we wrote that report. However, Bishop (2006) points out that the technique was proposed initially by Sung and Poggio (1994), who called it the elliptical *K*-means method.

THE GAUSSIAN MIXTURE MODEL (GMM)

The Gaussian Mixture Model, or GMM, (Bishop, 2006) is a mixture pdf of N M-dimensional feature vectors $\boldsymbol{x}_n$, which are grouped into K classes C_K. Each feature vector has a conditional probability given by:

$$p(\boldsymbol{x}_n \mid C_k) = \frac{1}{(2\pi)^{M/2}\left|\Sigma_k\right|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)\right], \tag{8}$$

where $k = 1,\ldots,K$, $n = 1,\ldots,N$, $\boldsymbol{x}_n = \begin{bmatrix} x_{1n} \\ \vdots \\ x_{Mn} \end{bmatrix}$, $\boldsymbol{\mu}_k = \begin{bmatrix} \mu_{1k} \\ \vdots \\ \mu_{Mk} \end{bmatrix}$ and $\Sigma_k = \begin{bmatrix} \sigma_{11} & \ldots & \sigma_{1M} \\ \vdots & \ddots & \vdots \\ \sigma_{M1} & \ldots & \sigma_{MM} \end{bmatrix}$.

GMM starts with an initial guess of the means and covariance matrices of each class and determines the correct values by iterating to a solution. Unlike K-means, the data is never physically re-ordered during the process.

To illustrate the Gaussian Mixture Model, or GMM, I will use the example shown in Figure 8 and in the random order shown in Figure 9(b). The Gaussian functions can have full, diagonal or spherical covariance matrices. I will initialize the GMM with three means and covariances given by:

$$\mu_1 = \begin{bmatrix} 2.25 \\ 2.25 \end{bmatrix}, \mu_2 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu_2 = \begin{bmatrix} 1.75 \\ 1.75 \end{bmatrix}, \text{ and } \Sigma_1 = \Sigma_2 = \Sigma_3 = \frac{1}{4}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The spherical Gaussian functions for these initial estimates have been superimposed on the cross-plot in Figure 13.

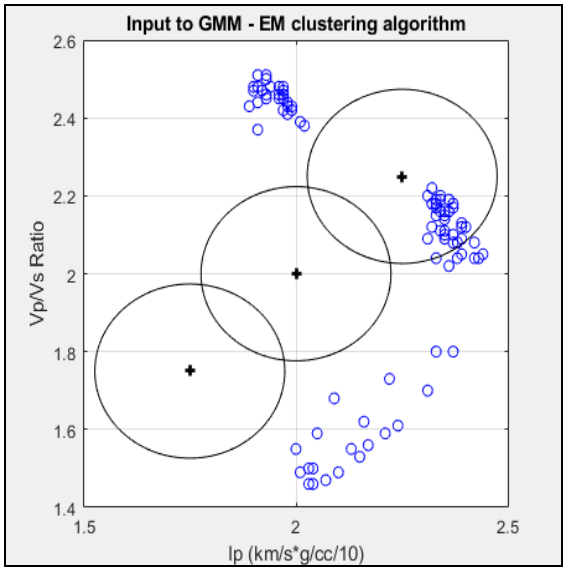


Figure 13: The initial estimate of the means and covariances for the GMM algorithm.

The first step of GMM is the expectation, or *E*, step and involves first determining the "responsibility" for each training point and in each class using Bayes' Theorem:

$$p\left(C_k \mid \boldsymbol{x}_n\right) = \frac{p\left(\boldsymbol{x}_n \mid C_k\right) p(C_k)}{p\left(\boldsymbol{x}_n\right)}, \tag{9}$$

where $p\left(\boldsymbol{x}_n\right) = \sum_{k=1}^{K} p\left(\boldsymbol{x}_n \mid C_k\right) p(C_k)$.

The second step is the maximization, or *M*, step and involves re-estimating the component pdfs and mixture weights where the new probability is given as

$$\hat{p}(C_k) = \frac{1}{N} \sum_{n=1}^{N} p\left(C_k \mid \boldsymbol{x}_n\right), \tag{10}$$

with means $\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^{N} p\left(C_k \mid \boldsymbol{x}_n\right) \boldsymbol{x}_n}{\sum_{n=1}^{N} p\left(C_k \mid \boldsymbol{x}_n\right)}$ and covariances $\hat{\Sigma}_k = \frac{\sum_{n=1}^{N} p\left(C_k \mid \boldsymbol{x}_n\right)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^T(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)}{\sum_{n=1}^{N} p\left(C_k \mid \boldsymbol{x}_n\right)}$.

This step is repeated until convergence. Figure 14 shows our clustered result, where the colours represent the "labels" which will tell us how to classify the points. Also, the elliptical contours of the first three standard deviations have been superimposed on the clusters. Notice that all the points have been correctly classified and, visually, the contours on this plot look very close to those shown in Figure 12, for Mahalanobis *K*-means clustering.

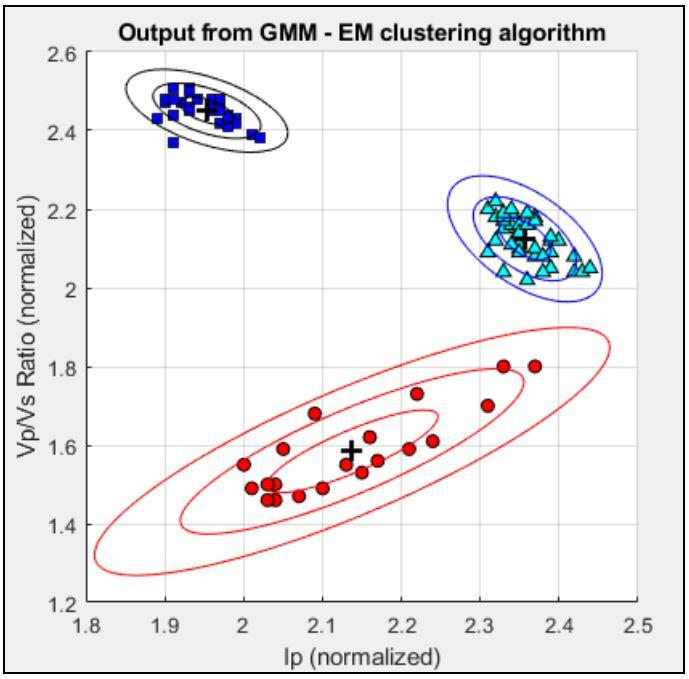


Figure 14: The final estimates of the means and covariances for the GMM algorithm.

To show that the contours in Figure 14 are identical to those in Figure 12, here are the statistics for the three clusters:

$$\mu_1 = \begin{bmatrix} 2.1375 \\ 1.5840 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 0.0119 & 0.0096 \\ 0.0096 & 0.0110 \end{bmatrix},$$
$$\mu_2 = \begin{bmatrix} 1.9529 \\ 2.4503 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.0012 & -0.0006 \\ -0.0006 & 0.0012 \end{bmatrix},$$
$$\text{and } \mu_3 = \begin{bmatrix} 2.3571 \\ 2.1248 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 0.0011 & -0.001 \\ -0.001 & 0.0029 \end{bmatrix}.$$

Comparing these results with those shown earlier for Mahalanobis *K*-means clustering we see they are identical. Thus, despite the differences between the two algorithms, they produce identical results. This is pointed out by Bishop (2006), where he refers to GMM as a "soft" clustering approach and Mahalanobis *K*-means clustering as the "hard" clustering approach. However, in general GMM requires more iterations than *K*-means.

SEISMIC FACIES PREDICTION WITH UNSUPERVISED CLUSTERING

In this section, I will go beyond the two-dimensional examples of clustering given in the previous sections and apply our two clustering techniques to seismic facies prediction.

Coléou et al. (2003) were the first to apply unsupervised clustering techniques to seismic facies classification, using both the Self Organizing Map (SOM) technique and *K*-means clustering.

Wallet and Hardisty (2019) applied the GMM technique to seismic facies classification. I will not apply the SOM technique in this study but will focus on the *K*-means technique as well as GMM, by extending the theory discussed in the earlier sections.

In this example, I will apply the *K*-means and GMM algorithms to a 3D seismic volume recorded over the Blackfoot field in Alberta (Dufour, 2002). On of the seismic lines from this volume is shown in Figure 15. Unsupervised clustering using the two algorithms will be applied to seismic structural slices taken from a 10 msec (5 sample) window taken 10 msec below the Lower Mannville event, which is the deepest event shown on the section. That is, we will create "phantom" horizons starting 10 msec below the picked Lower Mannville event and continuing to 18 msec below this event. Each map will be parallel to the structure picked at the Lower Mannville.

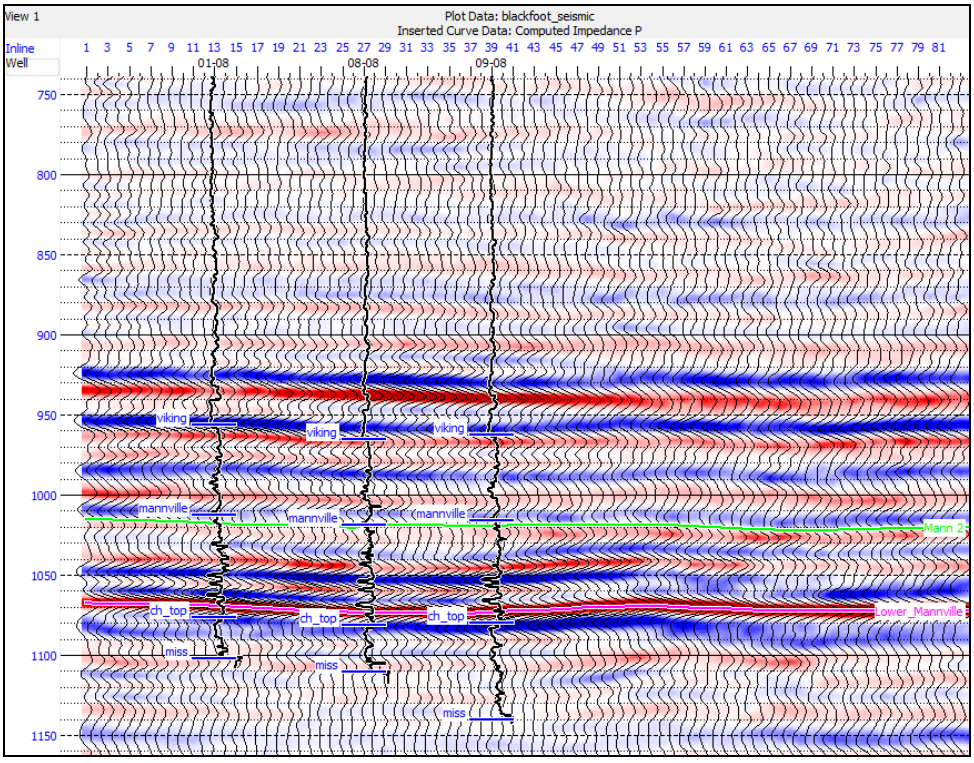


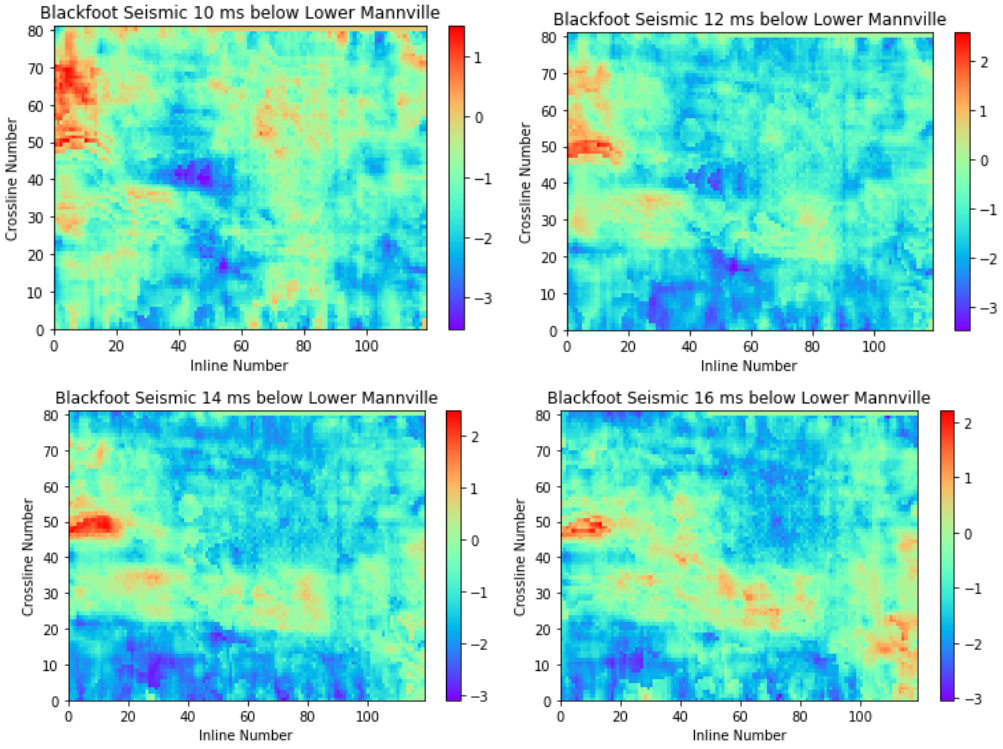
Figure 15: A single seismic line taken from the Blackfoot 3D volume, where we will analyze five seismic structural slices taken below the Lower Mannville picked event.

The structural slices are shown in Figure 16, where the colour bar represents standard deviations of the seismic amplitude away from zero. Note that the seismic volume consists of 119 inlines by 81 crosslines, for a total of 9639 traces. Thus, the clustering algorithms will see the input as 9639 5-dimensional vector points, or

$$\boldsymbol{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \\ x_{i5} \end{bmatrix}, i = 1,\ldots,9639\,,$$

where x_{ij} is the seismic amplitude of the j^{th} slice at the i^{th} seismic grid node.

On these slices we can notice two dominant events. First, these is a NW to SE channel event that becomes more dominant as the slices get deeper, shown in red on the slices, where red represents positive amplitudes. Second, we see a strong positive event on the NW part of the map which gradually diminishes as we move deeper. Third, there is a roughly circular event in the top east part of the map which changes polarity from positive to negative amplitudes as we move to deeper times.



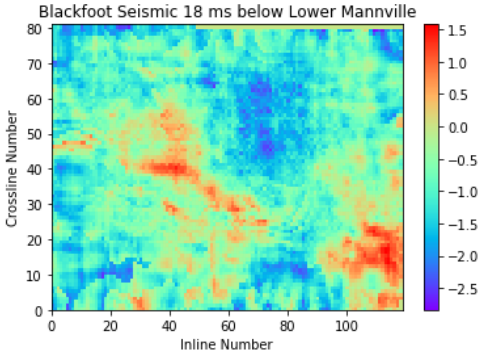


Figure 16: The five structural slices from the Blackfoot dataset used as inputs to the two clustering algorithms.

Now we will apply the *K*-means and GMM algorithms to the five structural slices shown in Figure 16, which will give us an unsupervised estimate of the seismic facies over the map surface. The implementation was done using the scikit-learn package in Python (see Appendix A). The *K*-means implementation in scikit-learn is the standard implementation using Euclidean distance, so the results from the GMM algorithm can also be interpreted as the results we would have obtained from Mahalanobis *K*-means.

In the following figures, I will use three, five and ten clusters, respectively. Note that because GMM assigns the initial cluster means and covariances randomly, the colours of the clusters will not always match between the *K*-means and GMM results.

The results using three clusters is shown in Figure 17. The *K*-means and GMM results are quite similar except that *K*-means sees the NW-SE feature as less extensive than GMM and shows a prominent event (in blue) on both sides of this feature. The facies on the NW part of the map is clearest on the *K*-means result. The large circular facies in the top east side of the map is more obvious in the GMM result.

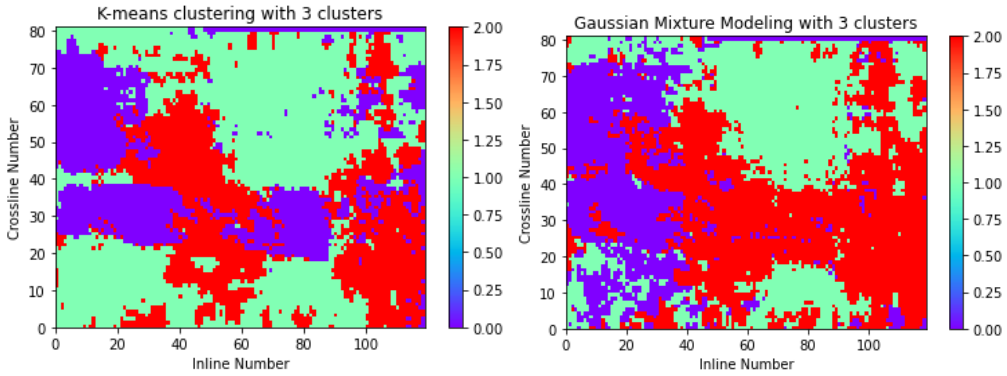


Figure 17: K-means clustering (left) and GMM clustering (right) with three clusters, using the input data shown in Figure 16.

The results using five clusters is shown in Figure 18. Again, the *K*-means and GMM results are even more similar than the three cluster result of Figure 16 if you take into account the different colour schemes for the cluster order. Both the *K*-

means and GMM algorithms show similar detail in the NW-SE feature and also on the event in the NW corner of the map. Also, the large circular facies in the top west part of the map is becoming more distinct in both methods.

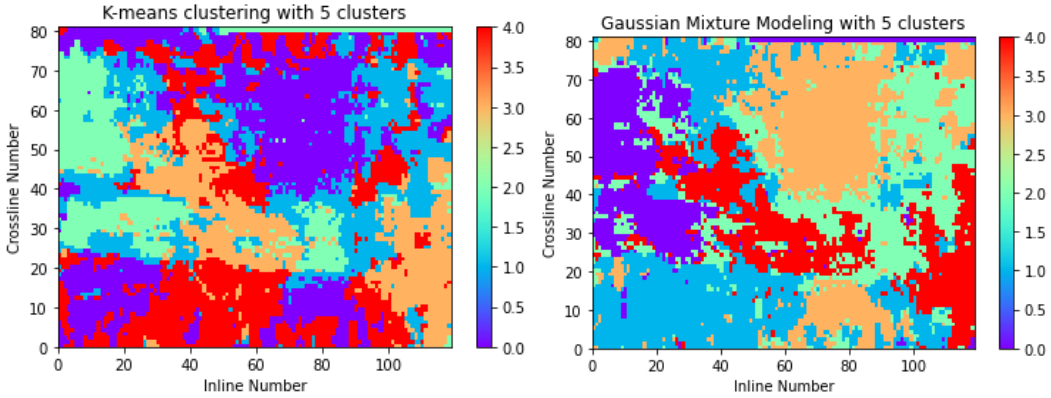


Figure 18: K-means clustering (left) and GMM clustering (right) with five clusters, using the input data shown in Figure 16.

Finally, the results using ten clusters is shown in Figure 19. Now the *K*-means and GMM results are not as similar as in the three and five clusters examples. Also, both results are quite "busy" looking. Both approaches sees the NW-SE facies as much less continuous. The GMM algorithm has shown the large circular feature in the top west of the map as very continuous, but the *K*-means algorithm has broken it up into several pieces. The only feature that is similar on the two results is the facies in the NW corner of the map.

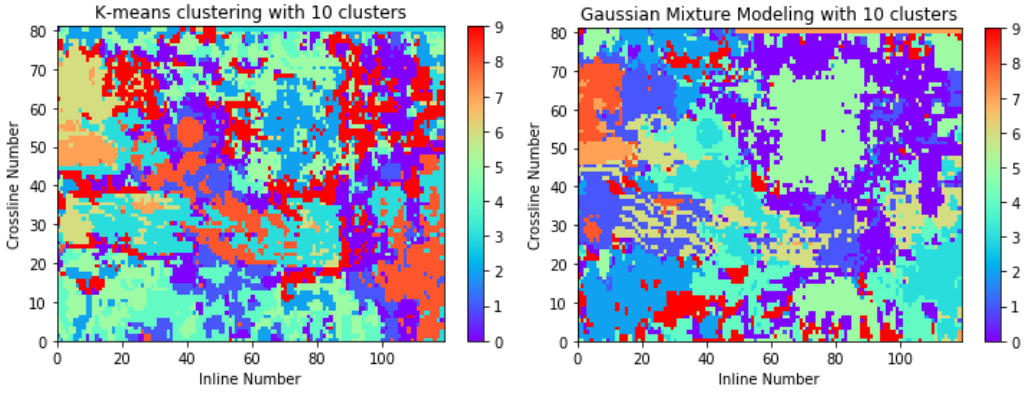


Figure 19: K-means clustering (left) and GMM clustering (right) with ten clusters, using the input data shown in Figure 15.

CONCLUSIONS

In this study, I have applied the *K*-means clustering and Gaussian Mixture Modeling algorithms to several examples, two of which were "toy" examples and two of which were real geoscience examples. In the first section I described the theory of *K*-means clustering using a Euclidean distance metric. When applied to a simple example with clearly defined circular clusters, this algorithm behaved well. However, when applied to a synthetic dataset that mimiced an AVO class 3 anomaly, which has elliptical clusters. I then showed that by using a statistical, or

Mahalanobis, distance metric, *K*-means was able to correctly identify the clusters. We called this method Mahalanobis *K*-means clustering (Russell and Lines, 2003), but it is also referred to as elliptical *K*-means clustering (Sung and Poggio, 1994).

We then described the theory behind Gaussian Mixture Modeling (GMM) and applied both GMM and K-means clustering (both Euclidean and Mahalanobis) to results from the pre-stack inversion of a Gulf Coast dataset, where our two attributes were acoustic impedance (I_P) and V_P/V_S ratio, and we had three clusters that represented a gas sand, a wet sand/shale complex, and a carbonate. Even though the clusters were well separated, the Euclidean *K*-means algorithm miss-classified one of the points. However, the Mahalanobis *K*-means and GMM algorithms classified all the points correctly and gave identical results.

Finally, in the last section I applied Euclidean *K*-means and GMM to seismic facies prediction. The reason I did not apply Mahalanobis *K*-means is that I used scikit-learn in Python for the application, which only has a Euclidean *K*-means option. The two algorithms were applied to a 3D seismic volume recorded over the Blackfoot field in Alberta, in which we extracted five seismic structural slices from below the Lower Mannville event. Thus, unlike our first three examples, we were in 5-dimensional, rather than 2-dimensional space. We applied the algorithm with three, five and ten clusters. From the results it would appear that the five cluster result gives the optimum number of facies, since the results of both algorihms were the most consistent in those displays. However, if you do want to run clustering with a lot more than five facies, the Gaussian Mixture Modeling approach appears to give the most continuous and consistent results.

APPENDIX A

Here is a listing of the Python program that created the clustered facies maps:

```
import numpy as np
import matplotlib.pyplot as plt
# Read the data, consisting of a 9639x5 dimensional array,
# where each column consists of a single 119x81dimensional
# structure slice from the Blackfoot seismic volume
M = np.loadtxt("C:\Text_Files\Blackfoot_data.txt")
# Import and run the Gaussian Mixture Model algorithm from scikit learn
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=5, covariance_type='full').fit(M)
labels_gmm = gmm.predict(M)
g_centers = gmm.means_
g_cov = gmm.covariances_
g_weights = gmm.weights_
c = labels_gmm.reshape(119,81)
# Plot the GMM result
plt.pcolor(np.transpose(c),cmap = 'rainbow')
plt.xlabel('Inline Number')
plt.ylabel('Crossline Number')
plt.title('Gaussian Mixture Modeling with 5 clusters')
```

```
plt.colorbar()
plt.show()
# Import and run the K-means algorithm from scikit learn
from sklearn.cluster import KMeans
kmeans = KMeans(5, random_state=0)
labels_km = kmeans.fit(M).predict(M)
d = labels_km.reshape(119,81)
# Plot the K-means result
plt.pcolor(np.transpose(d),cmap = 'rainbow')
plt.xlabel('Inline Number')
plt.ylabel('Crossline Number')
plt.title('K-means clustering with 5 clusters')
plt.colorbar()
plt.show()
```

Note that the input data was exported as a text file from the Python ecosystem in the HampsonRussell Geoview program. If the script is run through the HampsonRussell Python ecosystem it is possible to expand the number of data slices that are extracted and used as input to the two clustering algorithms.

REFERENCES

Bishop, C.M, 2006, Pattern Recognition and Machine Learning: Springer-Verlag.

Coléou, T., Poupon, M., and Azbel, K., 2003, Unsupervised seismic facies classification: A review and comparison of techniques and implementation: The Leading Edge, 942-953.

Dufour, J., Squires, J, Goodway, W.N., Edmunds, A., and Shook, I., 2002, Integrated geological and geophysical interpretation case study, and Lamé rock parameter extractions using AVO analysis on the Blackfoot 3C-3D seismic data, southern Alberta, Canada: Geophysics, 67, 27-37.

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, Upper Saddle River, New Jersey.

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

Russell, B.H and Lines, L.R., 2003, Mahalanobis clustering, with applications to AVO classification and seismic reservoir parameter estimation: CREWES Research Report **15**.

Sung, K.K. and T. Poggio, 1994, Example-based learning for view-based human face detection: AI Memo 1521, MIT.

Wallet, B.C. and Hardisty, R., 2019, Unsupervised seismic facies using Gaussian mixture models: Interpretation, August 2019, SE93 – SE111.

ACKNOWLEDGEMENTS

We wish to thank our colleagues at the CREWES Project and at HampsonRussell Software, a division of CGG GeoSoftware for their support and ideas, as well as the sponsors of the CREWES Project.