EXTRACTION OF UNDERLYING GEOLOGICAL STRUCTURE FROM SEISMIC DATA USING DATA MINING TECHNIQUES

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Ushasi Ghosh

August 2014

EXTRACTION OF UNDERLYING GEOLOGICAL STRUCTURE FROM SEISMIC DATA USING DATA MINING TECHNIQUES

Ushasi Ghosh

ACKNOWLEDGEMENTS

I would like to begin by acknowledging the guidance and support I received from Dr. Ricardo Vilalta. Dr. Vilalta's knowledge and analyzing skills have impressed me again and again throughout the course of my study. I am thankful for the all the discussions and the advice that I have received and this thesis would not be in its current shape without his help.

I gratefully acknowledge the help I received from Dr. Pablo Guillen for patiently introducing me to the project and the new concept of seismology. My interactions with Dr. Guillen encouraged me to looking at the problems from an industrial point of view which was critical for the success of this project. I would also like to thank Dr. Thomas K. Holley and Dr. Nikoloas V. Tsekos for taking out time from their busy schedule and be a part of my committee. I thank my fellow lab mates for their help and collaboration especially Roberto Valerio. I appreciate his participation in this project and thank him for taking out time from his own dissertation.

I express my appreciation to my parents, my husband and my friends for their continuous love and support. This thesis would not have been possible without them.

EXTRACTION OF UNDERLYING GEOLOGICAL STRUCTURE FROM SEISMIC DATA USING DATA MINING TECHNIQUES

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Ushasi Ghosh

August 2014

ABSTRACT

The development of seismic-imaging technology has substantially improved the exploration of subsurface deposits of crude oil, natural gas and minerals. Recent advances in data capture, processing power and storage capabilities have enabled us to analyze large volumes of seismic data. In this study we report on the implementation of machine learning and data mining techniques for analysis of seismic data to reveal salt deposits underneath the soil. Several seismic attributes have been extracted from these datasets. Using information gain, the best six attributes (homogeneity, contrast, energy, median, peaks and average energy) have been selected for further classification. Finally we compared the results obtained using four different clustering techniques: k-means algorithm, expectation maximization algorithm, min-cut algorithm and Euclidean clustering.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Machine learning is the study of how to develop algorithms that can make the learner (the machine) recognize patterns from complex data and make accurate predictions. When applied to raw data, most learning algorithms are unable to make precise predictions. Instead, the algorithms are applied to features extracted from the raw data. These features are developed by experts in the field and contain important prior knowledge that machine learning would be unable to discover of its own. Here in this thesis, we have applied machine learning algorithms to seismic features, extracted from the data provided by our collaborator Repsol Energy. This chapter focuses on the basics of machine learning as well as seismic data—which is important for a better understanding of the features used, and ultimately to attain more accurate predictions.

1.1 Machine Learning and Data Mining

With the advancement of computers and digital storage, an increasingly large amount of data is collected and stored every day. These large amounts of raw data often have hidden information lying beneath it. Analyzing and extracting information from these large datasets is challenging. Exploring large data in tabular form is nearly impossible. Although some of these data can be plotted graphically, due to the size of the datasets, local patterns hidden in the data are often overlooked. The goal of data mining is to discover patterns in large datasets involving methods at the intersection of machine learning, statistics and database systems.

Machine learning evolved from the broad field of artificial intelligence with the objective of growing intelligent abilities in machines. To quote Arthur Samuel, it is the “field of study that gives computers the ability to learn without being explicitly programmed” [1]. In machine learning, we write a programmable algorithm such that it optimizes some performing criteria with the help of example data or past experience. That is, we try to create a model which has parameters, and we try to optimize these parameters with the help of training data or past experience. Supervised learning requires labeled examples [2]–[5] to predict correct labels for novel inputs; labelling large datasets requires significant amount of resources. In the unsupervised feature learning approach, the goal is to learn a useful feature representation and forming data groups only from unlabeled examples [6].

To analyze the seismic data, provided by our collaborator Repsol Energy, we extracted relevant features. We studied a class of algorithms to analyze the datasets; in order to find the most efficient algorithm it’s important to understand the higher level concepts encoded in the seismic data.

1.2 Seismic Data

Seismology [7] is one of the most powerful methods of investigating subterranean formation of crude oil and mineral deposits. In this technique [8]–[10], shock waves or sound waves are generated by a source (e.g., air guns or vibrators). These waves pass through the Earth and some of their energy is reflected back to surface and captured by detectors or receptors known as geophones (when in ground) or hydrophones (when on water) [Figure 1.1]. From these collected data, a model of the subsurface structure is generated. Once the seismic data are collected, seismic processing is done, where the data is manipulated into an image.

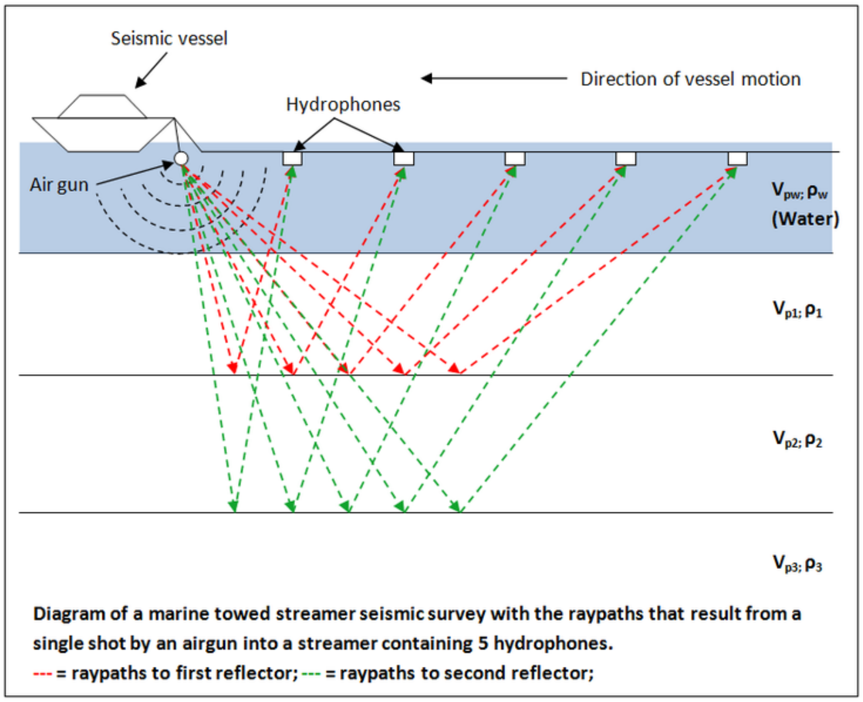

Seismic vessel
Direction of vessel motion
Hydrophones
Air gun
$V_{pw}; \rho_w$
(Water)
$V_{p1}; \rho_1$
$V_{p2}; \rho_2$
$V_{p3}; \rho_3$
Diagram of a marine towed streamer seismic survey with the raypaths that result from a single shot by an airgun into a streamer containing 5 hydrophones.
--- = raypaths to first reflector; --- = raypaths to second reflector;

Figure 1.1 *Seismic data collection by the generation of sound waves and the detection of reflected waves by sensors [adapted fromhttp://blog.cloudera.com/b;og/2012/01/seismic-data-science-hadoop-use-case]*

A set of seismic data can be classified as two-dimensional (2D), three-dimensional (3D) or four-dimensional (4D). A set of 2D seismic data represents a two-dimensional section of a part of the subsurface. 3D seismic data is the volumetric display of the underground structure and 4D datasets are the same collected at at least two different times.

There are some terms commonly used in seismology [11]. Each individual reflection mentioned above is called a trace or seismic line. To reduce the probability of errors, the traces are not formed from one single reflection. Multiple shots are produced from the same position, and an average of their reflections is taken. This average value is then represented as a single trace. There are two more terms related to seismic data— inlines and crosslines. Inlines represent the traces that were collected parallel to the direction of data acquisition, whereas crosslines represents traces which were collected perpendicular to the direction of data acquisition [Figure 1.2].

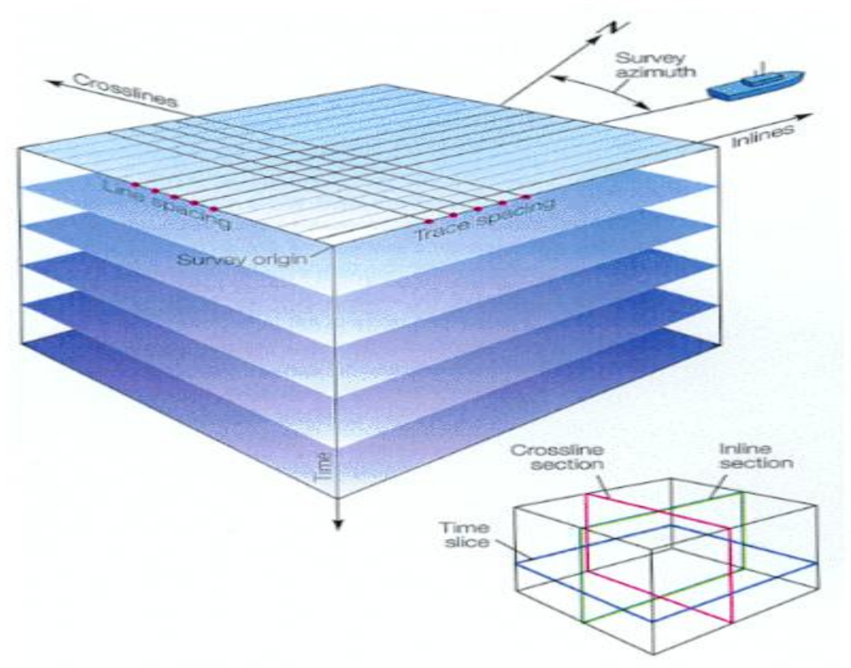


Figure 1.2 *Schematic diagram of a three dimensional seismic data with seismic traces and cross lines.* [adapted from *h*ttp://www.glossary.oilfeild.slb.com/en/Terms.aspx?Look In= term%20name&filter=crossline]

Seismic data is one of the most important forms of data for oil and gas industry for detecting the position of the oil.

1.3 Seismic Data and Machine Learning

Seismic exploration yields large volumes of data and a manual analysis of such large amount of data would require a significant amount of resources. Alternatively, machine learning can be used to analyze a given seismic data to locate and extract the structure and position of a certain seismic feature (e.g., salt structure) in it. Knowing the structure and position of the salt is important to locate the presence of oil and natural gas below it [12]–[14]. Best seismic attributes for categorizing a salt structure were selected. We applied a class of machine learning algorithms to the seismic images with an aim of distinctly classifying the salt from the rest of the image.

1.4 Our Approach

The data were first preprocessed before applying any machine learning techniques to it. Preprocessing of data is necessary to extract the important information from the data and to eliminate the unwanted information that might not help in classification or may hinder the results. Moreover, preprocessing of data is necessary to eliminate various noises in the data.

Data preprocessing is done by first extracting various attributes from the data [15]. Again, selection of the best few attributes was necessary as some of these attributes might not give us important information and hamper the results [16]. In addition, elimination of some of these attributes was necessary to escape the curse of dimensionality. The attributes were selected according to their strength of class separability [17].

After pre-processing, we explored the data with different clustering techniques. Clustering is partitioning a group of entities (data points) into different groups (clusters) such that the entities belonging to the same group are more similar compared to the entities in the other groups. The data points or entities can be a wide range of things depending on the field of study. Data points can be two-dimensional or three-dimensional. By plotting on a graph, the data points are often represented in an image.

The first and most challenging thing in clustering is to find the similarity measures between entities, or simply to know when object1 is more similar to object2 than object3. Finding the similarity measure is relatively easier for data points or entities that can be represented by numerical values and plotted in an n dimensional space. In that case, the simple concept of Euclidian distance can be used to define the degree of similarity. It becomes complex when entities cannot be represented by a numerical value and hence cannot be plotted in an n dimensional space. Now, we need to find a way to represent it in some form of numerical value or find some other way to measure the similarity between two objects.

In this project we experimented with four different clustering algorithms: the k-means algorithm [18], the expectation-maximization algorithm [19], the min-cut algorithm [20] and the Euclidian clustering technique [21]. We observed that the efficiency and the output results are different for different clustering techniques. Some form of similarity was noted between the k-means algorithm and the expectation maximization algorithm.

The k-means algorithm tries to find k different clusters. It initially takes k points as their means or centroids of the k clusters. Then it tries to assign all the points in the dataset to each of these k clusters. Then it updates the centroids of these clusters by finding the average of the data points assigned to that particular cluster. These two steps are followed iteratively until convergence.

The Expectation Maximization (EM) algorithm is similar to k-means algorithm, but instead of doing a crisp assignment of data points to a cluster, it assigns the data points to clusters with a probability. This algorithm considers that the dataset is the result of probability distributions which can be defined with some parameters. It iteratively tries to find the probability that each of the objects belong to these distributions and then tries to find the new parameter estimates of these distributions. Iteration continues until convergence.

The min-cut algorithm finds the *cut* in a graph representative of data points, so that the cut (sum of the weights of the points the cut goes through) is *minimum*. This completely divides the entire dataset into two distinct classes.

The fourth algorithm used is the Euclidian clustering algorithm. This algorithm uses a simple approach. It considers a point and then finds its neighbors (two points are neighbors if their distance is below a mentioned threshold) and assigns all of them to a particular cluster. It then considers the neighbors of the neighbors and assigns them to the previous cluster. This is done until a threshold (number of points that can be in a cluster) is reached. After which a new point which is not assigned to any cluster is considered and the same steps are repeated.

1.5 Outline

This thesis is segmented into 5 chapters. After introduction, Chapter 2 discusses the preprocessing techniques used for calculating various attributes. Chapter 3 reviews the details of the four clustering techniques and the algorithms along with the proximity measures used. Chapter 4 shows the results obtained in our experiments and discusses the results in more details. In the conclusions, Chapter 5 briefs the progress that is made by this thesis and the impact it is going to make in the relevant field of study.

Chapter 2

Data Preprocessing: Feature Extraction and Selection from Seismic Data

In this chapter we discuss the methods for feature extraction of seismic data, and the selection of a subset of these features based on their strength to achieve class separability.

2.1 Feature Extraction

In seismology, advanced techniques are used to send a shockwave beneath into the subsurface, and receive the reflection back to produce seismic data which carries the signatures of underlying seismic features. Once the original data is collected and mapped to the image, we are ready to process this data for classification. However, raw data may not be useful for accurate results. Therefore, there is a need to enhance it further. This can be done by feature analysis [22]–[24].

It is possible to create a new set of attributes from the original attributes which will have the capability of capturing important information from the raw data. This is known as feature analysis. Feature analysis is done in three methods: feature extraction, mapping the data to a new space, and feature construction [25]. Mapping the data into new space to get rid of the noise involves methods such as Fourier Transformation. Feature construction is used when the original data possesses important information, but traditional data mining algorithms are incapable of isolating them. Here, a new set of features are constructed from the original set of attributes [26], [27].

For our experiments we have adopted the method of feature extraction. Feature extraction is done by creating a new set of features from the original raw data. An important issue with feature extraction is that it is highly domain specific. Over time various features and feature extraction techniques developed for a specific field might not be suitable for applications in other domains [26], [27].

The seismic data was provided by our collaborator Repsol Energy. We used these data to implement the machine-learning algorithms in our project. We initially extracted 19 features from the original data set. Most of these were traditional features used for seismic data analysis. We tested three more features extracted from Gray Level Co-occurrence Matrices: homogeneity, entropy and contrast. These features are often used in traditional image processing. The three new features were proved to be superior to the rest in terms of class separability. As we continue, the different extracted features will be discussed in more details.

We investigated 19 attributes: some traditionally useful in seismic data analysis as suggested by our collaborators and the rest were evaluated from Gray Level Co-occurrence Matrices.

2.1.1 Mean

The first feature calculated was the mean of the raw values. Mean is evaluated to smooth out the raw data by eliminating noise. We considered our seismic image as a two-dimensional matrix. Next, we decided on the window size. Suppose we consider a window size of n, then our window will be a (n x n) matrix (which are (n x n) neighboring elements from the seismic data). We then found the mean of these neighboring vales representing the value of the middle most elements.

While choosing the value of n, one must keep in mind that n must be an odd number so that there is no conflict for the center element of the window.

In our example below, we consider n as 3 and give an example how we move the window and how calculations are done.

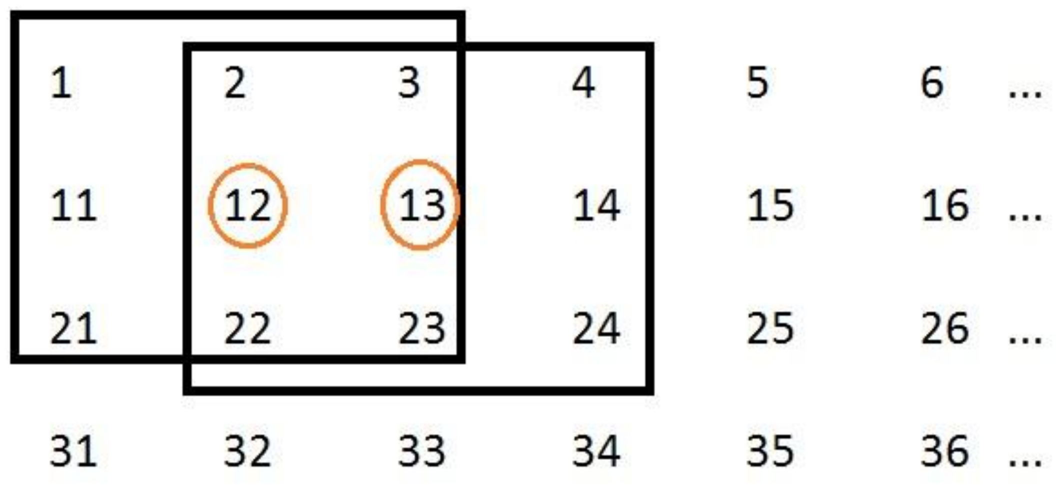


Figure 2.1 *Example of how the window moves.*

Our first window consists of the element numbers 1, 2, 3, 11, 12, 13, 21, 22 and 23. We calculate the mean of the values of these elements and assign it to the most centered element (in our case it is the element 12). Once the value of the 12th element value for the specific attribute is assigned, we then move our window to one step right. Here we calculate the mean of the values of the elements that fall within this window (element no. 2, 3, 4, 12, 13, 14, 22, 23 and 24 in our example) and assign it to the most centered element of the window (i.e., the 13th element). In this way we create a moving window which moves 1 element each time. Once a row is completed, we then start from left hand side of the next row and hence cover the entire two-dimensional image. With this moving window, we lose (n-1)/2 rows or columns of elements from each edge of the original data.

We calculate the mean as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Where x_i are the values of the elements inside the window.

2.1.2 Median

The next attribute is the median. It is also calculated with a moving window as follows:

All the n^2 samples in the window are ordered in ascending order, such that

$$x_1 \leq x_2 \leq x_3 \leq \ldots\ldots \leq\leq \ldots\ldots \leq x_N$$

where $N = n^2$

Then the median is defined by

$$x_{median} = x_k$$

where $k = (N+1)/2$

2.1.3 Variance

Variance is evaluated with the moving window using the formula:

$$\sigma^2 = \frac{\sum_{i=1}^{N}\left(x_i - \bar{X}\right)^2}{N}$$

where $\bar{X}$ is the mean of all the values in the window.

2.1.4 Standard Deviation

Standard Deviation is calculated by finding the square root of the variance calculated in the above case.

Later either the variance or the standard deviation (only one of them) will be considered to avoid redundancy among the features. This is also true in the case of mean and median.

2.1.5 Reflection Intensity

Reflection intensity (RI) is calculated by the following formula with the moving window.

$$RI = \frac{1}{N}\sum_{i=1}^{N}\left|x_i\right|$$

2.1.6 Average Energy

With the moving window, average energy (AE) is calculated as follows:

$$AE = \frac{1}{N}\sum_{i=1}^{N} x_i^{2}$$

2.1.7 Curve Length

Curve length (L) is calculated as follows:

The two-dimensional array is converted into a one-dimensional array and then calculation is done as follows:

$$L = \sum_{i=1}^{N-1} \left| x_{i+1} - x_i \right|$$

The value obtained is assigned to the middle most value of the two-dimensional windows.

2.1.8 Threshold

The threshold (γ) is derived using:

$$\gamma = \frac{3}{N-1} \sqrt{\sum_{i=1}^{N} \left(x_i - \bar{X} \right)^2}$$

2.1.9 Peak

Here the two dimensional window matrices are again converted to a one-dimensional matrix and the following is calculated:

$$\kappa = \frac{1}{2} \sum_{i=1}^{N-2} max \left\{ 0, \left| sgn\left[x_{i+1} - x_i \right] - sgn\left[x_{i+2} - x_{i+1} \right] \right| \right\}$$

where

$$max(a,b) = \begin{cases} a & if \quad a > b \\ b & if \quad a < b \\ a \ o \ b & if \quad a = b \end{cases}$$

$$sgn(x) = \begin{cases} 1, & if \quad x > 0 \\ 0, & if \quad x = 0 \\ -1, & if \quad x < 0 \end{cases}$$

2.1.10 Root Mean Square

Root Mean Square (δ) is calculated from the moving window as follows:

$$\delta = \sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N}}$$

2.1.11 Average Non-linear Energy

The window was once again converted to a one-dimensional array and the following was calculated:

$$\Psi = \frac{1}{N-2} \sum_{i=2}^{N-1} x_i^2 - x_{i-1} x_{i+1}$$

2.2 Derivation of Attributes from Gray Level Co-occurrence Matrices

The next remaining attributes are derived from Gray Level Co-occurrence Matrices. Texture Analysis is also widely used in the field of image processing. Features obtained from the Gray-Level Co-occurrence Matrix extensively used for the texture analysis. A two-dimensional seismic data can be considered as an image. The Gray Level Co-occurrence Matrix is a two dimensional dependence matrix, which captures the spatial dependence of gray-level values of the image that contribute to the understanding and analysis of texture.

2.2.1 Formation of Gray Level Co-occurrence Matrices

To calculate the Gray Level Co-occurrence Matrices (or GLCM) we used the moving window again. However, this time we generally selected a comparatively larger window size for accuracy. For our experiments we used a window size of 11 or greater. Once this window was selected, our first task was to discretize this window into the desired number of gray levels (say k). Now the co-occurrence matrix P[i,j] (of size k x k) can be defined with the help of the displacement vector **d**, where **d** = (dx,dy), tracking the number of pixels separated by **d** having gray levels i and j. The three displacement matrix can be of 3 types (0,1), (1,1), (1,0). This process of formation of co-occurrence matrix is shown below with an example.

The three displacement vectors are represented below:

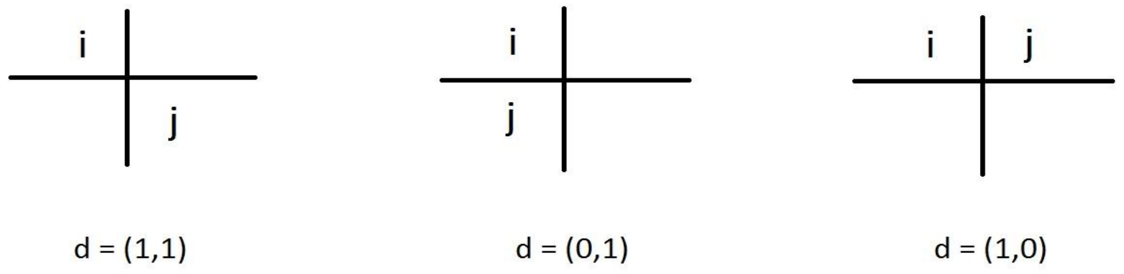


Figure 2.2 *Three different types of displacement vectors*

The three co-occurrence matrices are formed in accordance to each of the displacement vector **d**.

Say we have k gray levels, then the size of the co-occurrence matrix will be (k x k). We start with displacement matrix **d** = (1,1). Now we fill the position P(i,j) of the co-occurrence matrix with the number of pairs who has the value i and the value j in the discretized window exactly in the positions of **d** = (1,1) as shown in the above figure. The values of these matrices are then divided by the total number of comparisons done to fill each cell of the co-occurrence matrix. For example, if **d** = (1,1), then number of comparisons done = (k-1)*(k-1). Where as if **d** = (0,1), number of comparisons = k*(k-1), and if **d** = (1,0), the number of comparisons = (k-1)*k. An example is given below.

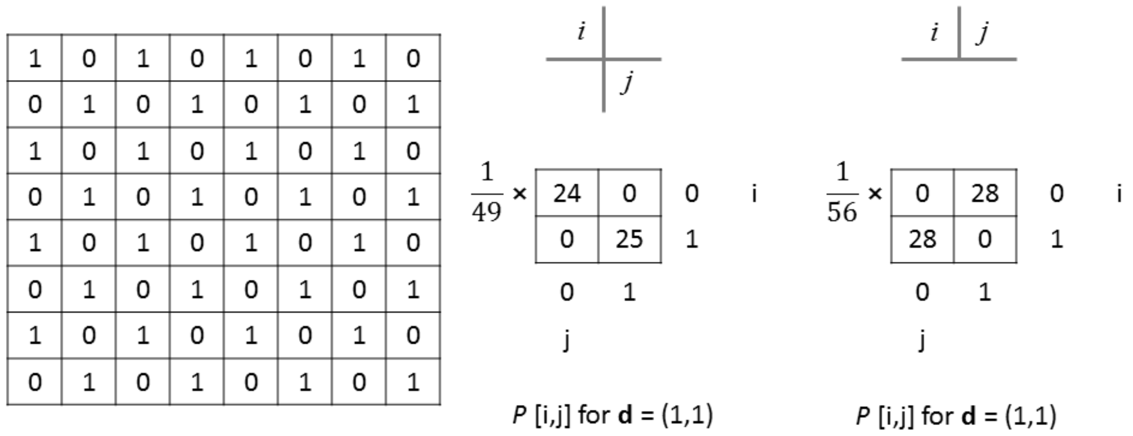


Figure 2.3 *Example of creation of Co-occurrance Matrix from a* 8x8 *window (Reference Machine Vision by Ramesh Jain, Rangachar Kasturi, Brian G. Schunck).*

Once the co-occurrence matrices are calculated, the three variables energy, contrast and homogeneity are calculated. These values are calculated as the following and the best among the three values calculated for each of the attribute is selected [3].

2.2.2 Energy from GLCM

Energy is calculated from each of the Gray Level Co-occurrence Matrices as follows:

$$\sum_{i=1}^{k}\sum_{j=1}^{k} P[i,j]^2$$

2.2.3 Contrast from GLCM

Contrast is calculated using the formula:

$$Contrast = \sum_{i=1}^{k}\sum_{j=1}^{k}(i-j)^2 P[i,j]$$

2.2.4 Homogeneity from GLCM

Homogeneity is calculated as follows:

$$Homogeneity = \sum_{i=1}^{k}\sum_{j=1}^{k}\frac{P[i,j]}{1+|i-j|}$$

2.2.5 Other Attributes

Five other attributes, Chaos Texture, Contrast Texture, Weighted second momentum, Zero Crossing, and Instantaneous phase were provided by our collaborators.

2.3 Selection of Features

Though creating new attributes help us to retrieve more information that might be hidden in the original data, it also has some limitations. Some of these attributes might not contribute at all in the role of classification. Using classification techniques along with these attributes might hinder the results instead of enhancing them. Moreover, too many attributes may lead to the curse of dimensionality. Hence choosing the right set of attributes is very important to attain high accuracy in the classification step.

To solve the above problem, each of the attributes was analyzed in terms of their strength of classification on the training data. There is more than one method to attain this. For our experiments, we used Information Gain to assess these attributes.

We first divided all the numeric attributes into a number of intervals (values where there seemed to be a change of class in the training data). The Information Gain obtained from these values was noted and the best information gain obtained was chosen. Also the value of the attribute where this information gain was obtained was marked.

The Information Gain of an attribute A, at any point was calculated as follows:

$$IG(A) = H(S) - \sum W_i H(S_i)$$

where,

$H(S)$ = the entropy of the class distribution for the entire training set,

$H(S_i)$ = the entropy of each of the subsets of examples included by each value of A.

W_i = The weight which is estimated by the fraction of examples with that value of A_i .

Entropy is defined as follows:

$$H(X) = -\sum P(x_i) \log_2 P(x_i)$$

where, x is a random variable.

Lower values of entropies are preferred as they correspond to less random and more structured distributions. On the other hand, Information Gain measures how much entropy reduction is achieved in class distribution when an attribute is used to separate the classes. Hence, higher values of Information Gain are preferred as it corresponds to high entropy reduction. When each attribute interval or in our case attribute partition comprises of more examples of the same class, Information Gain takes higher value.

According to the Information Gain obtained from the attributes, we ranked them in order. This is represented in Table 2.1:

Table 2.1 *List of the attributes ranked according to the Information Gain obtained.*

Rank	Attribute	Information gain						Standard Deviation
		Run 1	Run2	Run3	Run 4	Run 5	Mean	
1	Homogeneity	0.2135	0.2708	0.2708	0.2708	0.2708	0.25934	0.02292
2	Contrast	0.1901	0.1773	0.24	0.1127	0.2492	0.19386	0.049126
3	Energy	0.1621	0.3352	0.1087	0.1523	0.144	0.18046	0.07944
4	Mean	0.1336	0.1665	0.1501	0.1878	0.1215	0.1519	0.023513
5	Median	0.132	0.1372	0.1402	0.1697	0.1663	0.1498	0.015706
6	Peaks	0.2741	0.1288	0.0876	0	0.1375	0.1256	0.088796
7	Average Energy	0.094	0.0691	0.1171	0.0439	0.0482	0.07446	0.027757
8	Reflection Intensity	0.1121	0.033	0.053	0.096	0.049	0.06862	0.030126
9	Instantaneous Phase	0.0589	0.0755	0.0579	0.0865	0.0516	0.06608	0.012914
10	Root mean square	0.0554	0.065	0.0764	0.0488	0.13	0.05172	0.028974
11	Threshold	0.0673	0.0297	0.049	0.0679	0.0256	0.0479	0.017923
12	Contrast texture	0.0579	0.0361	0.0552	0.0401	0.042	0.04626	0.008657
13	Standard deviation	0.0082	0.0294	0.041	0.0348	0.0906	0.0408	0.027235
14	Curve Length	0.0511	0.0467	0.0158	0.0212	0.0496	0.03688	0.01517
15	Weighted Second Momentum	0.0245	0.0192	0.058	0.0251	0.0358	0.0325	0.013833
16	Variance	0.0435	0.0262	0.0065	0.0614	0.0205	0.03162	0.019037
17	Average non-linear Energy	0.0206	0.0185	0.0346	0.0179	0.0186	0.02204	0.006346
18	Chaos Texture	0	0.0072	0.0176	0.0343	0	0.011852	0.012958
19	Zero Crossings	0	0	0	0	0	0	0

From the above table we can observe a marked difference between the 6th and 7th ranked attributes. The top six attributes stand out in terms of discriminative power. Hence from this result we first decided to use these top six attributes for further classification. However, if we notice the attributes carefully, we observe a redundant attribute among the top six attributes. Mean and median may act as redundant attributes. For this reason, we chose mean median, and discarded median. Instead we select the 7th attribute (i.e. average energy) for further classification instead of the median.

Homogeneity has the highest rank among the rest of the attributes. Homogeneity captures the degree of smoothness in an image. This can help in isolation of certain geological structures from the background elements. High homogeneity partnered with high energy and low contrast generally indicates massive deposits of salt. As a result, the top three attributes seems to be really important for automated classification processes.

The other top attributes like mean, peak, and average energy capture direct signal information and consequently contain high amount of information to help the classifier to distinguish between different geological structures.

The class separation power of the top three attributes can be explained with the following results. Taking into account the value for which we found the best information gain, we plotted our images into two classes positive and negative in the following figures. In the original image, the distinctive triangular area is the salt structure. This is an image, where we already know the structure of the salt body in it. The values corresponding to the red class shown in our results corresponds to the structure of the salt, if a crude estimation was made taking the value of the attribute corresponding to the best information gain as a threshold and separating the original space of data into two classes.

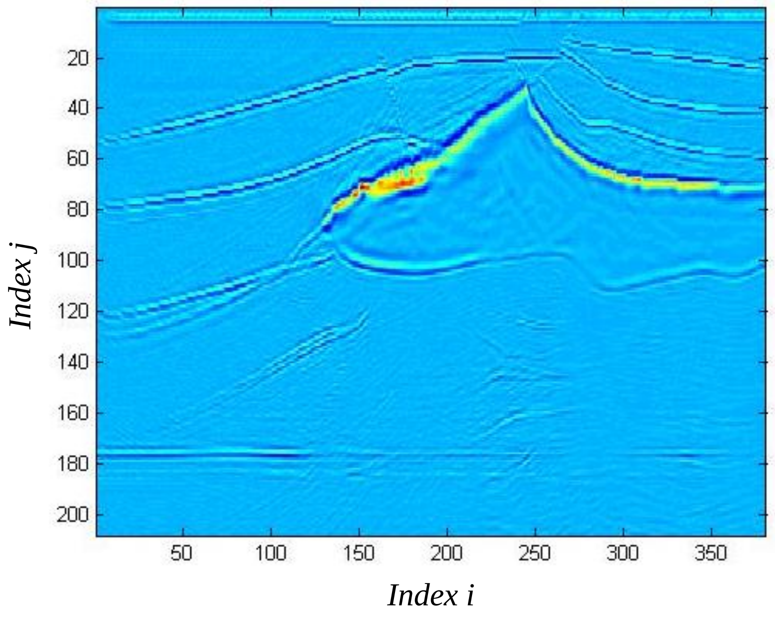


Figure 2.4 *208×381 2D-seismic image with raw data. The distinctive triangular area in the center is salt.*

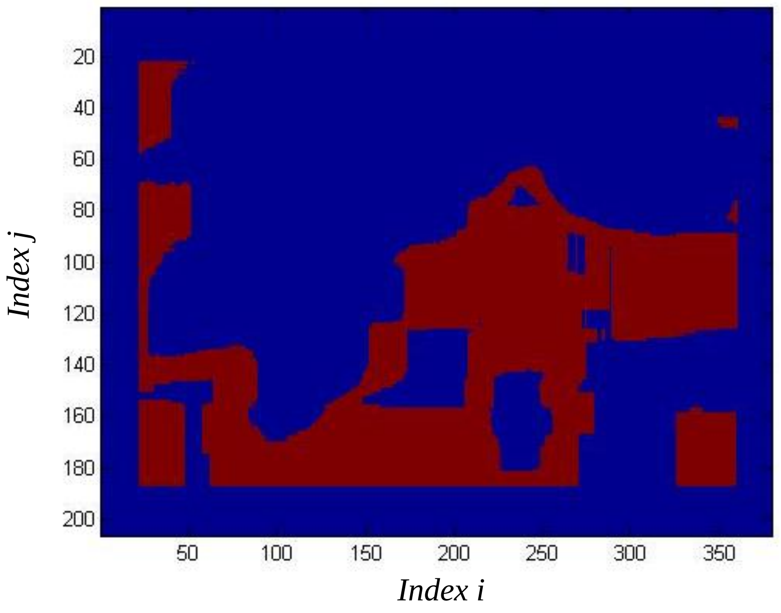


Figure 2.5 *208×381 2D-seismic image after seismic interpretation using the class separation power of homogeneity as the threshold. Two different colors represent two different classes.*

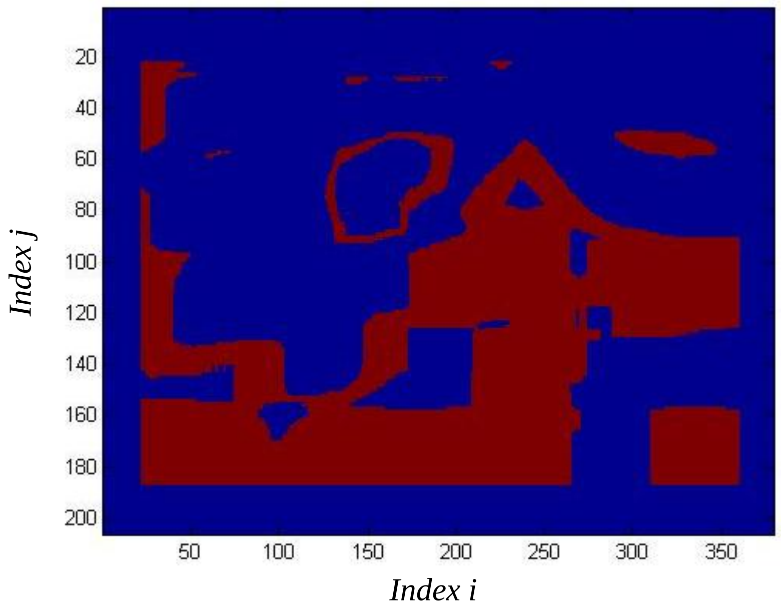


Fig 2.6. *208×381 2D-seismic image after seismic interpretation using the class separation power of contrast as the threshold. Two different colors represent two different classes.*

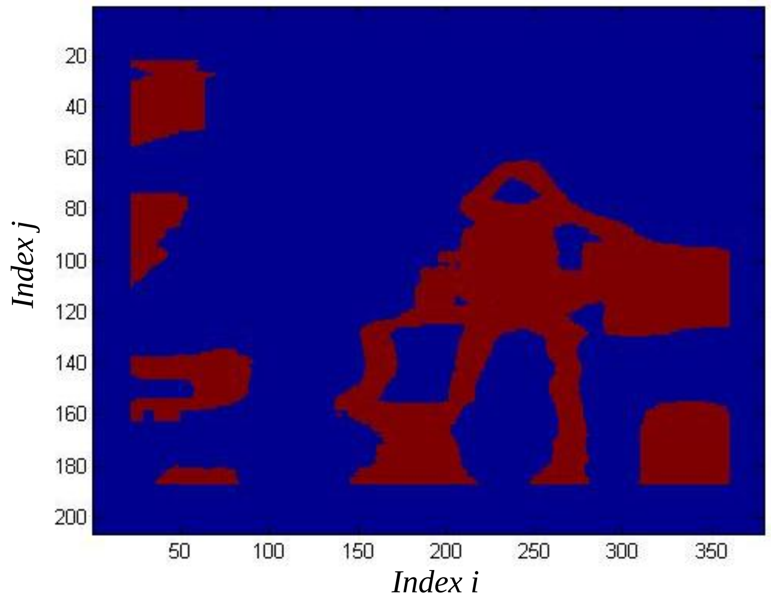


Fig 2.7 *208×381 2D-seismic image after seismic interpretation using the class separation power of energy as the threshold. Two different colors represent two different classes.*

Thus for our further experiments we use the attributes of Homogeneity, Contrast, Energy, Mean, Peaks and Average Energy as our subset of attributes.

Chapter 3

Experiments with Various Clustering Techniques

3.1. Introduction

In the first part of the project, we extracted attributes from our original raw data and then selected a subset of these attributes. These attributes are expected to have more power in terms of class separability compared to the attributes that were discarded.

Once this process of selection of attributes was over, the next challenge was to select a technique that would efficiently segregate the data into two distinct classes (salt and no-salt).This could have been done with classification techniques, clustering techniques or a combination of both. In our project, after the selection of the attributes or features, we started exploiting the data with different clustering techniques so as to divide the data into different clusters.

Clustering is a method of grouping data into subsets, such that the similar instances of the data are grouped together, while different instances belong to different groups [28]. Clustering is an unsupervised learning technique that can be utilized to extract patterns hidden in the data. Clustering algorithms can be categorized based on their cluster models. A detailed discussion of different clustering techniques is outside the scope of this thesis. Here we focus on the various clustering algorithms that we have used in this study.

In order to allocate data points to different clusters a measure of similarity/ proximity is required. This can be achieved by using a measure of distance between a pair of data points. Similarity can be computed by measuring the distance between two elements, i.e. how close they are from each other. The lesser the distance the more similar the objects are.

3.1.1 Distance or Proximity Measure

There are various methods of measuring distances between two different objects. Distance measurement depends on the type of data available (e.g., numeric, Boolean or string). Cosine Similarity and Jaccard Measure are often used for measuring proximity among documents. In this thesis, we are only concerned with numerical types of data.

For numeric data, different distance measures are available. Some of the commonly used distance measures are Euclidean Distance and Manhattan Distance. Other distances comprises of Squared Euclidean Distance, Normalized Squared Euclidean Distance, Chess Board Distance, Bray Curtis Distance, Canberra Distance, Cosine Distance and Correlation Distance.

For our experiments, we have used the most common measure of distance — the Euclidean Distance. The Euclidean Distance ($d_{Euclidean}$) between two points $(x_1, x_2, \ldots\ldots x_n)$ and $(y_1, y_2, \ldots\ldots y_n)$ in an n dimensional space is measured as follows.

$$d_{Euclidean} = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + \ldots\ldots + (y_n - x_n)^2} \quad \ldots\ldots\ldots\ldots\ldots\ [3.1.1]$$

3.2. K-Means Clustering

The k-means algorithm is one of the most primitive and fundamental clustering techniques of data mining. This technique is also widely used in image processing. In this section we will initially introduce the k-means algorithm and then discuss the techniques of evaluating the clusters formed from this technique. Subsequently we will proceed with the discussion of space and time complexity of this algorithm. We will conclude with the advantages and disadvantages together with the results obtained.

3.2.1 The Algorithm

To execute the k-means algorithm, the user first needs to decide on the number of clusters they want from the data. Once the number of clusters are decided (say k), we start by choosing k 'means' or 'centroids', by randomly selecting k points from the data and initialize them as the centroids of the k clusters. After the initialization of the k means the algorithm first assigns each point in the data set to a cluster and then it tries to update the clusters and their centroids at every step of the iteration. The assignment of the points to a cluster is done as follows. We measure the distance of each point from the k centroids of the clusters (which were assigned in the previous step). Then we assign the point to the cluster, whose centroid is closest to the point. Once the assignment is done we update the centroids of the clusters by calculating the average value of all the data points assigned to that cluster. These two steps of assigning points to a cluster and updating the centroids of the clusters each time is done iteratively until some stopping criteria are met. The stopping criteria are like the distance among the new and previous centroids are below a threshold or the maximum number of iterations was met. The pseudo code for the algorithm is given bellow:

```
K-meansAlgorithm
    ➢ Select k points from the data set randomly as initial centroids.
    ➢ Repeat until any one of the stopping criteria are met:
        o For each point in the dataset:
            Assign it to the cluster whose centroid is the closest to the point
        o For each cluster:
            Calculate the new centroid and update it.
```

3.2.2 Measuring the Quality of the Clusters Formed

Measuring the quality of the clusters obtained from a method is also vital to determine the efficiency of the clustering technique for particular type of data. These measuring techniques sometimes depend on the proximity measure used in the calculation. Sum of standard squared error (also known as the 'scatter') is one of the well-known methods used to measure the quality of the clusters. This is computed by calculating the error of each point from the centroid, i.e., the distance of a point in the cluster from its centroid and finally adding the square of these distances to find the sum of the squared error. Given below is the equation for calculation of the sum of squared error (SSE) for a particular cluster j.

$$SSE_j = \sum_{i=1}^{m} dist(x_i, c_j)^2 \quad \text{.. [3.2.1]}$$

where, $X_j = \{x_1, x_2, \ldots\ldots x_n\}$, $be\ the\ set\ of\ all\ the\ points\ that\ belong\ to\ the\ cluster\ j$

$$and\ c_j\ is\ the\ centroid\ of\ the\ cluster$$

Once the sum of the squared error for each cluster is found, the next step is to find the quality of the result obtained from the clustering technique. This can be done by finding the average sum of the squared errors found from all the clusters obtained.

$$SSE(C_1, C_2, \ldots\ldots C_k) = \frac{1}{k}\sum_{j=1}^{K}\sum_{i=1}^{m} dist(x_i\ , c_j\ )^2 \quad \ldots\ldots\ldots\ldots. [3.2.2]$$

In step 2, k-means algorithm tries to optimize the sum of squared errors in the iteration loop. Moreover, as it tries to minimize the sum of squared error for specific collection of centroids and clusters, it assures a local minimum. On the other hand, this quality of k-means algorithm might lead it to suboptimal clusters. This can be fixed by using larger value of k, i.e. a greater number of clusters. Quality of a cluster can also be determined from the standard deviation.

3.2.3 Space and Time Complexity of k-means Algorithm

Time Complexity:

Time complexity is linearly proportional to the number of data points, number of iterations and the number of clusters. Time required for a k means algorithm to run is *O(i*K*m*n),* where *m* is the number of data points, *n* is the number of attributes, *K* is the number of desired clusters and *i* is the number of iterations. As most of the changes usually occur in the first few iterations, hence *i* is usually not a big number. Thus we can conclude k means algorithm has a linear time complexity given the number of cluster is significantly lower than the number of data points [28].

Space Complexity:

K-means algorithm does not take up plenty of space as the only things needed to be stored are the data points and the centroids. The space required for k means algorithm is *O((m+K) n)*.

Hence the time complexity and the space complexity of the k-means algorithm seems to be clearly moderate which puts it to be much ahead of the other algorithms available.

3.2.4 Advantages and Disadvantages of k-means Algorithm

The k-means algorithm has the advantage over other commonly used clustering algorithms due to its linear time and space complexity. It also showed promising results for our experiments on seismic data. But on the other hand, k-means algorithm also possesses some drawbacks. One of its biggest weaknesses is its sensitivity to outliers and noise in the data. In addition, k-means does a crisp assignment of data points to the clusters. This hard assignment of points is not reasonable for points near the decision boundaries.

3.3. Expectation Maximization Algorithm

3.3.1 The Algorithm

It is often assumed that data is generated due to a result of statistical process and the data is described with the help of these statistical models that best fits the data. The statistical models are defined in terms of a distribution and a set of parameters of the distribution. Expecation Maximization (EM) Algorithm deals with a particular kind of statistical model called the mixture model. This algorithm was explained and its name was given in 1997 by Arthur Dempster, Nan Laird and Donald Rubin. [4]. It is widely used in genetics, clinical and social studies.

In mixture model, it is assumed that there are a number of probability distributions with definite parameters, and the data is a set of observations from a mixture of these different distributions. These probability distributions can be anything but most of the time is assumed to be a multivariate normal distribution. The reason behind assuming it to be multivariate normal is it is well inferred and also produces good results for most data. Here, each probability distribution corresponds to a cluster, whereas the parameters of the distribution actually describe the corresponding cluster, especially the area and centers of the clusters. Maximum Likelihood Estimation (MLE) is a procedure used to estimate parameters of these statistical models. Expectation Maximization (EM) algorithm uses the concept of maximum likelihood estimation for estimating the parameters of the mixture models. Here each distribution corresponds to different groups or better called clusters.

At the beginning of computation, we do not know the estimates of the probability distributions. However, the underlying knowledge of the probably distributions ($prob(x|\Theta)$) enabled us to find out the parameters of the distributions. The EM algorithm runs iteratively by first estimating the probability distribution (called the Expectation Step) and then use the above estimation to update the parameters of the probability distributions (called the Maximization Step). Iteration stops on convergence. Here, a crisp assignment of clusters is not implemented on the objects. Instead, the objects are assigned to clusters with a definite probability density of it belonging to the cluster, and the sum of all such probability distribution for an objects sum up to 1.

The EM Algorithm is discussed in more details after a brief explanation of the mixture model and the maximum likelihood estimation. Several distributions are given assigned with similar but different parameters. Then, one of these distributions was randomly selected to produce an object from them. In mixture model, each distribution represents a different group (a different cluster).

Now we state the problem statement:

Let there be K distributions, and N number of objects that need to be assigned to clusters. Let X be the set of all the objects: $X = \{x_1 , x_2 , ... , x_N\}$. θ_j be parameters of jth distribution and Θ be a set of all parameters, such that $\Theta = \{\theta_1 ..... \theta_K\}$. Let w_j be the mixture weight, i.e. the probability density that for any object the distribution j is chosen. These weights have a constraint of their sum being adding up to 1, i.e. $\sum^{K}_{j=1} w_j=1$. Then the probability distribution of an object x_i, parameterized on θ_j i.e. the probability density that an object x_i is from the jth distribution is:

$$P(x_i \mid \theta_j) \quad \text{...} [3.3.1]$$

Thus, the probability density of an object is:

$$P(x_i|\Theta) = \sum_{j=1}^{K} w_j \, p_j \, (x_i|\theta_j) \quad \text{................................} [3.3.2]$$

If we consider the objects to be generated in an independent manner, then we can consider the probability density of the entire set of objects as:

$$P(X|\Theta) = \prod_{i=1}^{N} P(x_i|\Theta) = \prod_{i=1}^{N} \sum_{j=1}^{K} w_j \, p_j \, (x_i \, |\theta_j) \text{ .... [3.3.3]}$$

For Gaussian univariate mixture model, the probability density function of a one dimensional Gaussian distribution for any point *xi* (Gaussian Density Function) is given by:

$$P\left(x_i\middle|\theta_j\right) = \frac{1}{\sqrt{2\pi}\ \sigma}\ e^{-\frac{(x-\mu)^2}{2\,\mu^2}} \text{ .... [3.3.4]}$$

Once it is assumed that the data is a mixture of Gaussian distributions, the next task is to estimate the parameters of these distributions using maximum likelihood estimation. We consider that m points from the data set is generated from a one dimensional Gaussian Distribution, and assuming that they are generated independently, the probability density of all the m points are the product of their individual probability densities. Hence from equation 3.3.4, we get:

$$P\left(X\middle|\theta_j\right) = \prod_{i=}^{m} \frac{1}{\sqrt{2\pi}\ \sigma}\ e^{-\frac{(x_i-\ \mu)^2}{2\,\sigma^2}} \text{ .... [3.3.5]}$$

Since probability densities are very small numbers, the equation is accounted in logarithmic scale:

$$\log P\left(X\middle|\theta_j\right) = -\sum_{i=}^{m} \frac{(\,x_i-\ \mu\,)^2}{2\,\sigma^2} - \frac{1}{2} m \log 2\pi\ - m \log \sigma \text{ .... [3.3.6]}$$

Now the values of the parameters are to be assigned carefully so that, the set of points suites it the best or the selected data is most likely, i.e. it maximizes equation 3.3.5. This is the maximum likelihood approach and the method of estimating the parameters are known as maximum likelihood estimation. For a specific set of data, the probability density is a function of the parameters of the distribution. The probability density equation in 3.3.5 is called the likelihood function. The log-likelihood function is also used very often due to small values of the probability density and the parameter values maximizing the likelihood function will also maximize the log likelihood function. The likelihood function is plotted by the probable values of the parameters to find out the values of the parameters that produces maximum value for likelihood. However, this approach is not feasible for big datasets. So statistically the problem can be solved by taking derivative of the equation of the likelihood function with respect to that parameter and setting the function to 0 and then solving for the parameters.

An issue with this alternative approach is that it is not possible to accurately predict which point comes from which distribution and calculating the probability density of each point is not viable. That is when the Expectation Maximization algorithm comes into play. Estimation maximization algorithm iteratively tries to estimate the parameters of the model.

The EM Algorithm basically consists of two major steps called the Expectation Step and the Maximization Step. These two steps are computed iteratively until convergence. In the expectation step, the EM algorithm tries to calculate the probability that each point belongs to each distribution. On the other hand, in the maximization step, these probabilities are used to find the parameters of the distributions.

The Expectation Maximization Algorithm is defined as follows and the two steps of Expectation and maximization as explained:

Expectation Maximization Algorithm:

- *Initialize the parameters of the distributions randomly*
- *Repeat till change in parameters are above a threshold*
    - *Expectation Step: For each object x_i, calculate P(j | x_i , Θ). i.e. the probability that each object belongs to each distribution.*
    - *Maximization Step: Find new estimates of the parameters of the distributions that maximize the expected likelihood.*

Expectation Step:

In the expectation step, we try to compute the probability that each object belongs to each distribution or more specifically a point came from a particular distribution, i.e., P(j | x_i , Θ). Hence, we can further rewrite the formulae as:

$$P(j\,|x_i\,,\Theta) = \frac{w_j\,P(\,x_i\,|\theta_j\,)}{\sum_{j=1}^{K} w_j\,P(x_i\,|\theta_j)} \quad \text{.................................. [3.3.7]}$$

Maximization Step:

The maximization step is already discussed in the section of maximum likelihood estimation. In some special cases of the expectation maximization algorithm, this maximization step is much more simplified. This will be explained in the next section, when the similarity of the expectation maximization algorithm with k-means algorithm is discussed.

3.3.2 Similarities with the k-means Algorithm

The k-means algorithm is a special case of estimation maximization algorithm. For Euclidian data, k-means algorithm is a special case with spherical Gaussian distribution having equal covariance matrices but different mean.

In k-means algorithm, the first step assigns each object to a cluster. In the expectation step of the EM algorithm, the each object is assigned to each distribution (in this case we can call it a cluster) only with certain probability density. In the second step of k-means algorithm, we compute the centroids of the clusters. On the other hand, in the maximization step of the expectation maximization algorithm, we try to compute the parameters of the distributions (or clusters) that maximized the likelihood function.

3.3.3 Advantages and Disadvantages of EM Algorithm

Like every other algorithm, the estimation maximization algorithm has advantages as well as some disadvantages. The EM algorithm is applicable to a wide range of data and is also helpful for clustering data sets which consists of missing data. It's fast and guaranteed to converge. The complexity at each iteration is always linear. Moreover, unlike some other algorithms, EM algorithm does not have the hassle of choosing a step size. Even with the above mentioned advantages, EM algorithm has its limitations, which constrains good results to all data types with any properties. EM algorithm is locally optimal, which means it may converge to local optima along with its limitation of the convergence speed. EM algorithm has slow convergence. It is also sensitive to initialization of parameters. These disadvantages create limitation to the application of EM algorithm to any kind of datasets.

3.4 Min-Cut Algorithm

3.4.1 Introduction

The min-cut algorithm is a graph-based algorithm [29]–[31]. The effort is given in bisecting the graph into two disjoint segments with an intention of disconnecting the most loosely connected nodes, at the same time keeping the nodes that are tightly connected to each other in the same partition.

A cut in a graph is a set of edges such that the removal of these edges disconnects the graph. All graphs considered here are weighted graphs. A min cut or minimum cut is a cut on the graph which has minimum cost. In other words, a min cut is also a set of edges such that these set have minimum weight and removal of these edges will disconnect the graph, i.e., a cut with minimum cost.

3.4.2 Problem Formulation for the Min-cut Algorithm

Suppose we have a graph with n vertices (which represents the data points in the data set) and m weighted undirected vertices. In our problem of seismic data, the weights edges can represent the distance measure between the two connecting vertices. The closer two points the lesser the distance between them. But according to our graph, edges that connect points closer to each other must have more weights than edges connecting points farther to each other. Therefore, the weights are assigned by inverting the distances between the points and normalizing these weights. It must also be noticed that in our problem with the seismic data, we can measure the distance between any points in the dataset with any other data point. As a consequence, the graph formed from this dataset will have all the vertices connected to each other. This would make the graph really big and computationally challenging.

Once we consider the graph with n vertices and m edges, our purpose is to separate the set of vertices into two non-empty sets such that it minimizes the total weight of the edges connecting them, i.e., to find the minimum cut.

There are certain assumptions we make before proceeding. First, the graph is always assumed to be connected, else the problem will be irrelevant. Secondly, we assume that the edges are non-negative, else the problem will be NP-complete by a nominal transformation from maximum cut problem.

This problem has two variants: the normal min-cut graph algorithm and the s-t min cut algorithm. The problem statement of the normal min-cut algorithm is mentioned above. In the s-t min-cut algorithm, we select two vertices s and t which are required to be on the two opposite sides of the cuts. The normal min-cut problem has no such restriction. If we consider all pair of vertices as s and t and implement the s-t min cut algorithm for each of these pairs, then the min-cut algorithm is the minimum taken over all the s-t min-cut among all the pairs. Besides, the min-cut problem for unweighted graph is similar to finding the connectivity of the graph. This means that to find the minimum number of edges that need to be removed to disconnect the graph.

The min-cut algorithm has applications in various fields. One of the most common applications of min cut algorithm is in the field of network design. Other areas of application include study of project networks, partitioning database, traveling salesman problem.

3.4.3 The Algorithm

Before we begin with the explanation of the algorithm, let us describe a few important terms used.

Most-Tightly-Connected Vertex

Let V be a set of all the vertices in the original graph and A be a subset of these vertices. The most-tightly-connected vertex to A is the vertex which does not belong to A and whose sum of edge weight to A is the maximum. In other words, if we consider each of the vertices that are not in A and evaluate the sum of the weights of the edges connecting them to the vertices in A, the vertex having the maximum sum is the most tightly connected vertex.

M*erging of Two Nodes*

When two nodes are merged, the weight of the new edges connecting the new nodes would be the sum of the edges between the nodes before the merge.

An example can be given as follows. Suppose we have a graph with 5 nodes or vertices: {A, B, C, D, E} and all the nodes are connected to each other. If we merge D and E, then the new set of nodes will be {A, B, C, DE}. The new weight of the edges will be as follows edge A-DE will have the weight of the sum of the edges A-D and A-E. Similarly edges B-DE will have the weight of (B-D + B-E) and C-DE as (C-D + C-E).

The pseudo code of the min-cut algorithm is given below.

Let V be the set of all the vertices, w be the weight of the min cut (calculated at each iteration).

```
Min-Cut Algorithm:

➢ $Set\ w = \infty$.
➢ $While\ |V| > 1$
        o $s - t\_Phase\_Cut = Min\_Cut\_Phase\ (G, w)$
        o if $weight\_of\_s - t\_Cut < w$
              $Min\_Cut = s - t\_Phase\_Cut$

        o $Merge(G, s, t)$
➢ $Return\ Min\_Cut$
```

```
Min-Cut Phase Function

➢ $a = Any\ arbitrary\ vertex\ of\ G$
➢ $A = \{a\}$
➢ $While\ A \neq V$
        $v_1 = Most\ tightly\ connected\ vertex\ to\ A$

        $A = A \cup (v_1)$

➢ $(s, t) are\ the\ last\ and\ second - last\ vertices\ added\ to\ A\ respectively$
➢ Return (A-t,t)
```

3.5. Euclidean Clustering

3.5.1 Introduction

Euclidean Clustering is one of the newer concepts in clustering techniques. This algorithm was not only conceptually transparent, it was simpler to implement relative to the previous algorithms and also gave us very interesting results. This algorithm needed a few user input parameters. Finding the right technique to find a good value for these parameters was the challenging part. We start with explaining the algorithm and then go to the techniques of finding the right value for the parameters.

3.5.2 The Algorithm

Let X be a set of all the data points where $X = \{x_1, x_2, \dots \dots x_n\}$ and d_{th} be the neighboring distance. The neighboring distance is the distance such that any two points whose distance is less than or equal to neighboring distance is considered as neighbors.

Euclidean Clustering Algorithm:

```
➢ Create an empty queue Q
➢ For every point $x_i \epsilon X$ and $x_i$ is already not assigned to any cluster:
    o Add $x_i$ to Q if $x_i$ is already not assigned to a cluster.
    o For every point $x_i \epsilon Q$
    o Search a set of points ($X_i^m$), such that if $x_i^m \epsilon X_i^m$ then,$x_i^m$ is a
      neighbor of $x_i$, i.e. $x_i^m$ lies in the sphere with $x_i$ as center and r
      as radius, where $r<d_{th}$.
    o For every neighbor $x_i^m \epsilon x_i^m$ if Q has not reached the threshold
      ( i.e. the maximum number of points a cluster can hold), check
      if the point has already been processed or assigned to a cluster.
      If not, add it to Q.
➢ Once all the points in Q have been processed, or Q has reached the threshold
  ( i.e. the maximum number of points a cluster can hold) assign all the points in
  Q to a new cluster.
➢ Empty Q.
```

First let us consider the neighboring distance d_{th}. Keeping a moderate value for m we can test of suitable values for d_{th}. Say we limit the value of m to be 5 or 10. Then we start with experimenting for the value of d_{th}. For a small value of d_{th}, the algorithm will start considering all the points testing as neighbors and the clusters will be filled up very fast. This can be perceived by keeping a track of the time taken for the clusters to fill up. This time will definitely depend on the processing power of the computer. Now if we gradually keep on changing the neighboring distance at a particular point of time, we will notice that that the time taken to fill up each of these clusters will increase significantly. This is the point when the algorithm stops considering all the points it checks as its neighbor. We can securely choose this value of the neighboring distance to run the algorithm for the final results.

The next challenge was deciding on the maximum number of points the clusters can hold. This method can be best explained with the help of the results obtained as shown in the figures in the results section. We experimented with starting with a smaller value of *m*. As we kept on increasing the value of *m*, we noticed a very interesting phenomenon. Initially, with a smaller value of *m*, the algorithm would group the data which would produce different clusters towards the edge of the structure of the salt. As we kept on increasing the value of *m*, the algorithm started to place all the points near the boundary of the salt structure into one single cluster and rest into the other clusters. But if we keep on increasing the value of *m*, the algorithm would distinctly produce two clusters: One with the points at the boundary of the salt structure and the other with the rest of the points. Slowly as we kept on increasing *m*, we could see the more precise boundary of this salt structure. Gradually with further increase of the value of *m*, the structure vanishes as the algorithm starts considering all the points as not its neighbor. This concept will be more transparent with the results shown in the next chapter. The reason behind this phenomenon is also explained more precisely in the section in our results section.

Chapter 4

Results

4.1 Introduction:

This project can be grossly categorized into two parts. The first part was the preprocessing of the seismic data and extraction of relevant attributes. The second part was the experimentation with a class of clustering techniques. The algorithms were coded in Matlab. Here in this chapter, we will discuss the results obtained and compare all the algorithms in the context of seismic data analysis.

4.2. Pre-processing: Feature Extraction and Selection

We plotted the original data in figure 4.1

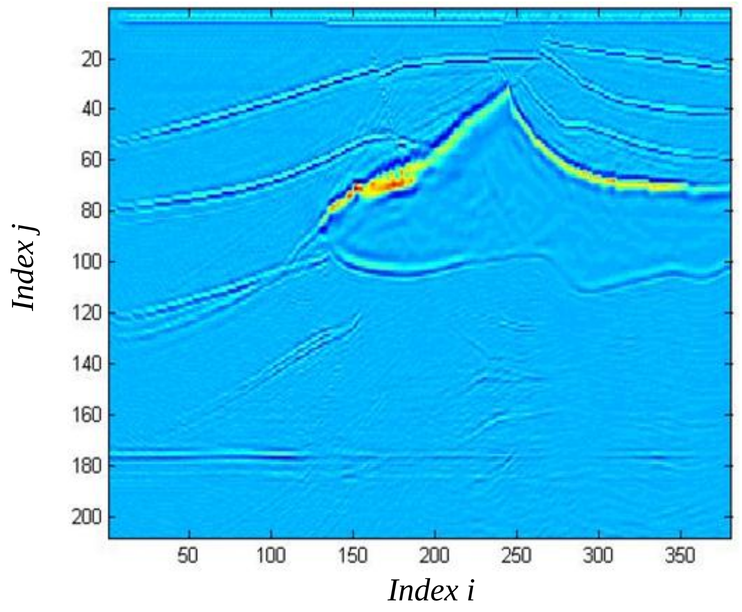
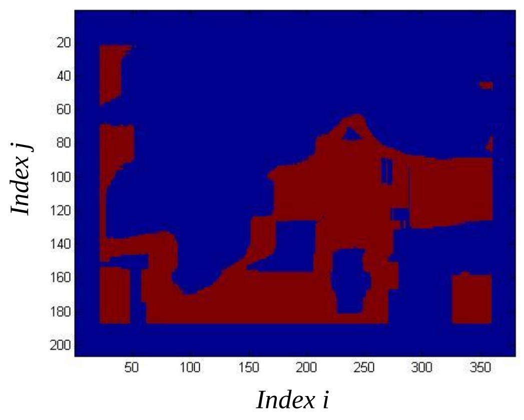
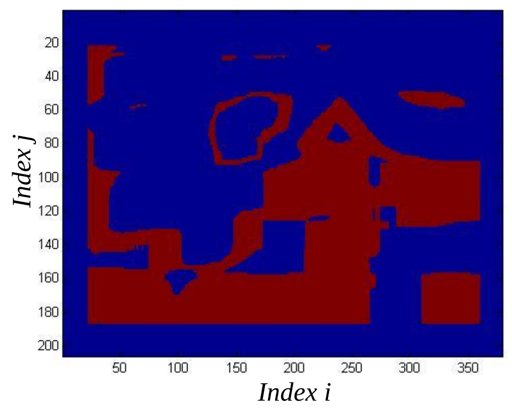


Figure 4.1 *208×381 2D-seismic image with raw data*

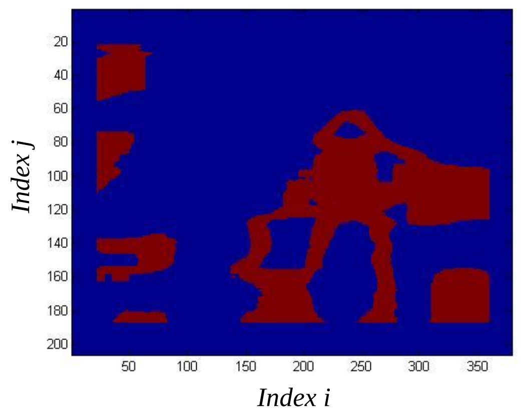
Data preprocessing was performed by creating 19 attributes and then finding out the information gains produced by the attributes. There were various thresholds or decision points (the value at the decision nodes when we considered it as a tree) which were used to divide the entire dataset into two parts. Among those we selected the best threshold, i.e., the threshold that gave the best Information Gain. Figure 4.2 displays the outcome if we consider these results.



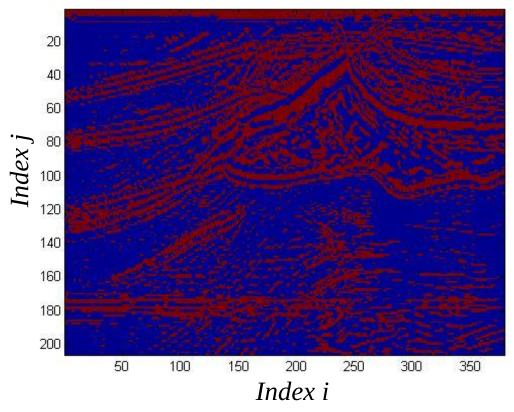
(a) Homogeneity



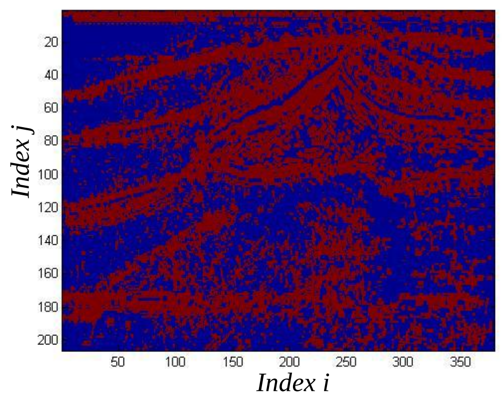
(b) Contrast



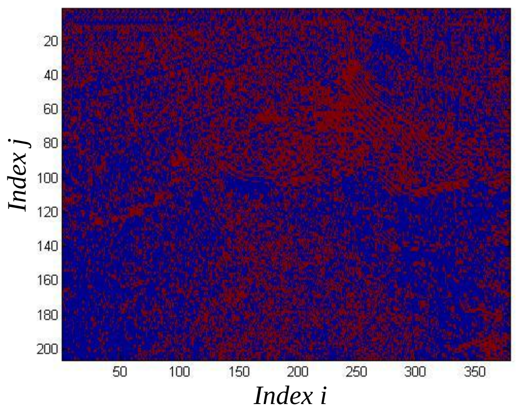
(c) Energy



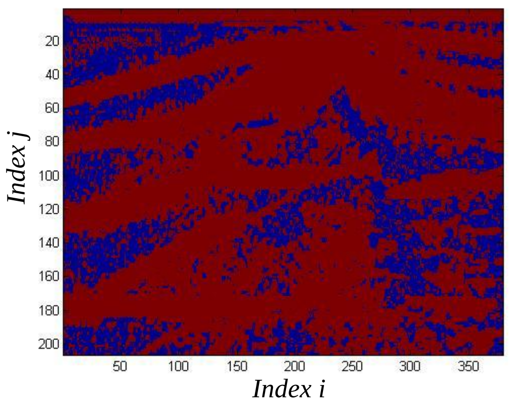
(d) Mean



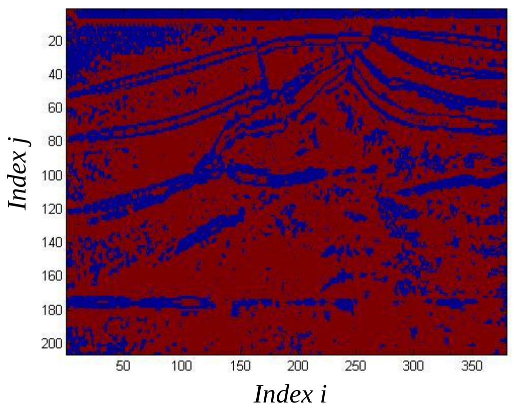
(e) Median



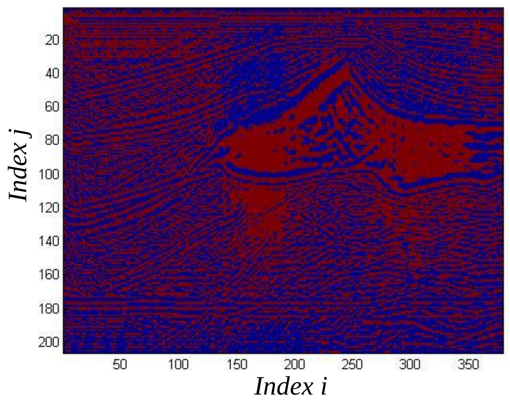
(f) Peaks



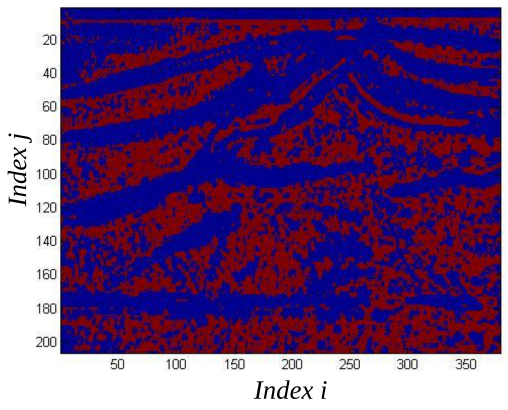
(g) Average Energy



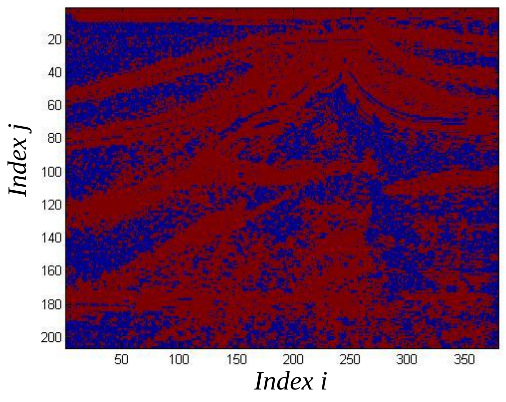
(h) Reflection Intensity



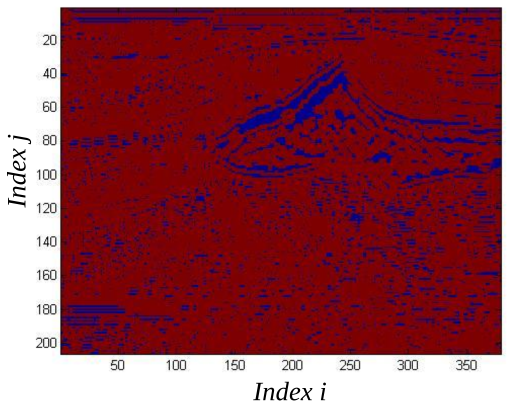
(i) Instantaneous Phase



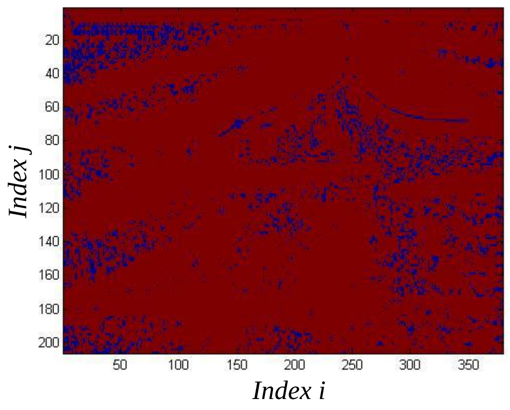
(j) Root Mean Square



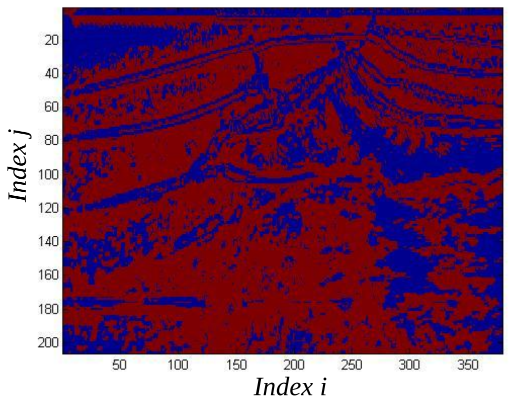
(k) Threshold



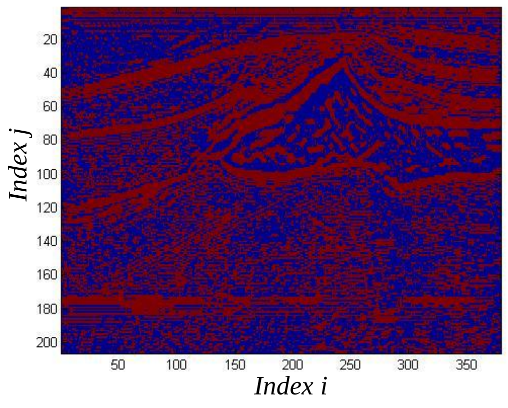
(l) Contrast Texture



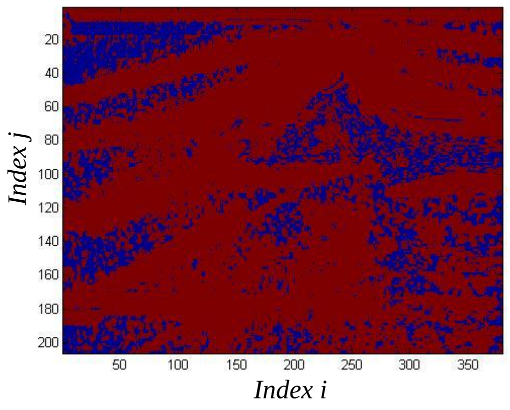
(m) Standard Deviation



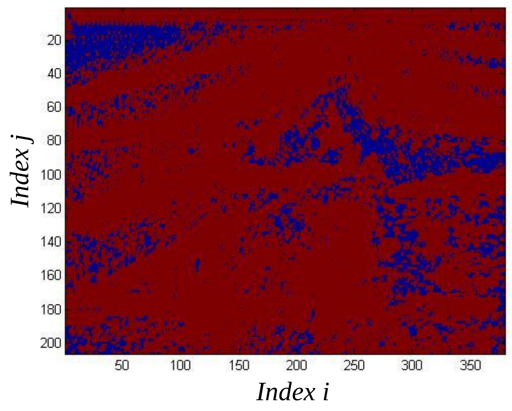
(n) Curve length



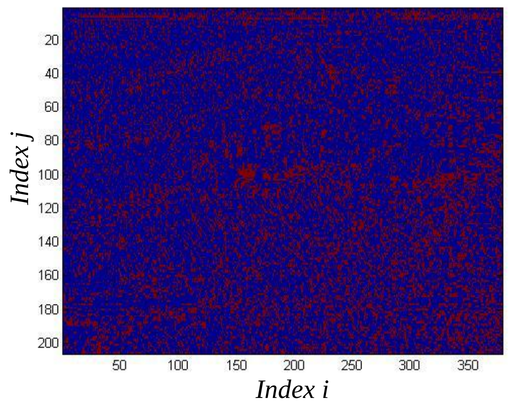
(o) Weighted Second Momentum



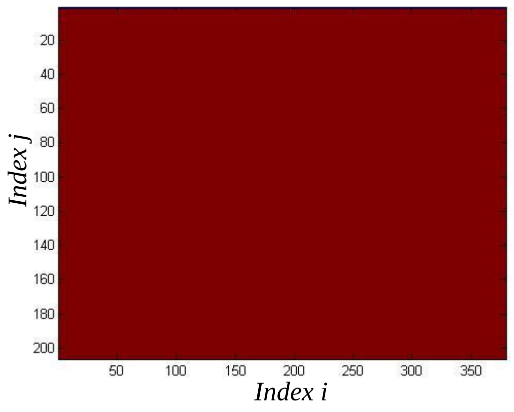
(p) Variance



(q) Average Non-linear Energy



(r) Chaos Texture



(s) Zero crossing

Figure 4.2 *208×381 2D-seismic image after seismic interpretation using the class separation power of the attributes arranged in accordance to the information gain obtained. Two different colors represent two different classes.*

For each of these attributes, we run them for 5 times to find the information gain and we consider the mean of the information gains observed. In the table below, we provide the information gains obtained from these attributes and also the standard deviation of the above. These attributed are ranked accordingly.

Table 4.1 *List of attributes ranked according to the Information Gain obtained.*

Rank	Attribute	Information gain						Standard Deviation
		Run 1	Run2	Run3	Run 4	Run 5	Mean	
1	Homogeneity	0.2135	0.2708	0.2708	0.2708	0.2708	0.25934	0.02292
2	Contrast	0.1901	0.1773	0.24	0.1127	0.2492	0.19386	0.049126
3	Energy	0.1621	0.3352	0.1087	0.1523	0.144	0.18046	0.07944
4	Mean	0.1336	0.1665	0.1501	0.1878	0.1215	0.1519	0.023513
5	Median	0.132	0.1372	0.1402	0.1697	0.1663	0.1498	0.015706
6	Peaks	0.2741	0.1288	0.0876	0	0.1375	0.1256	0.088796
7	Average Energy	0.094	0.0691	0.1171	0.0439	0.0482	0.07446	0.027757
8	Reflection Intensity	0.1121	0.033	0.053	0.096	0.049	0.06862	0.030126
9	Instantaneous Phase	0.0589	0.0755	0.0579	0.0865	0.0516	0.06608	0.012914
10	Root mean square	0.0554	0.065	0.0764	0.0488	0.13	0.05172	0.028974
11	Threshold	0.0673	0.0297	0.049	0.0679	0.0256	0.0479	0.017923
12	Contrast texture	0.0579	0.0361	0.0552	0.0401	0.042	0.04626	0.008657
13	Standard deviation	0.0082	0.0294	0.041	0.0348	0.0906	0.0408	0.027235
14	Curve Length	0.0511	0.0467	0.0158	0.0212	0.0496	0.03688	0.01517
15	Weighted Second Momentum	0.0245	0.0192	0.058	0.0251	0.0358	0.0325	0.013833
16	Variance	0.0435	0.0262	0.0065	0.0614	0.0205	0.03162	0.019037
17	Average non-linear Energy	0.0206	0.0185	0.0346	0.0179	0.0186	0.02204	0.006346
18	Chaos Texture	0	0.0072	0.0176	0.0343	0	0.011852	0.012958
19	Zero Crossings	0	0	0	0	0	0	0

The next step was to select the appropriate attributes from a pool of these 19 attributes. We selected the top six attributes called the Homogeneity, Contrast, Energy, Mean, Median and Peaks for further experimentation. We took a notice that there is a drastic change in the information gain obtained between the sixth and the seventh variable. Therefore, only these six variables were considered for further experimentations. Nevertheless, two of the selected variables, mean and median, were considered together as redundant attributes. We neglected mean and used the next attribute peaks for the rest of the experimentation.

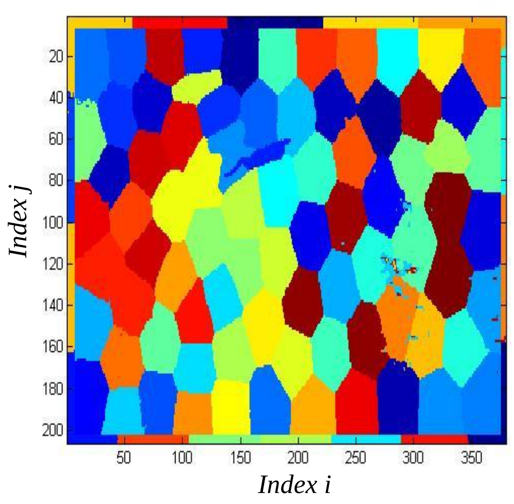
The spatial co-ordinates played an important role in the clustering techniques as we assigned the data points adjacent to each other in the same clusters instead of having clusters that have data points scattered all over. Taking that into account, we used the x and the y co-ordinates as the seventh and the eighth attribute.

4.3. Clustering the Data

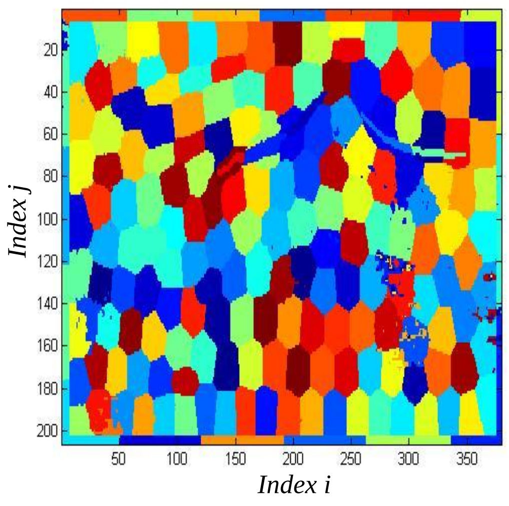
We implemented four different clustering algorithms, the results are discussed below.

4.3.1. K-means algorithm

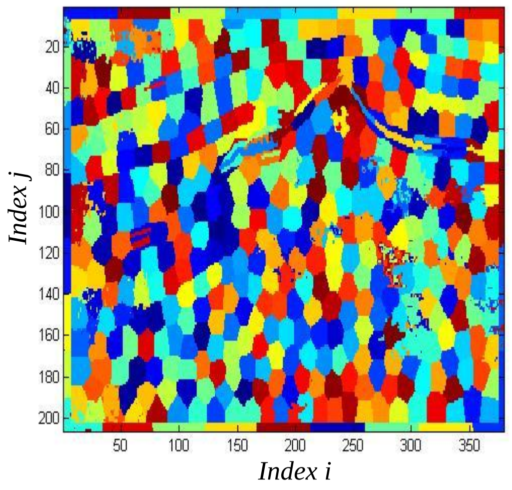
Matlab provides us with an inbuilt function for k-means. We have used both the inbuilt function and written our own code for the implementation of this algorithm. We have experimented with different values of 'k' i.e. the number of clusters. Selected results of both versions are shown below.



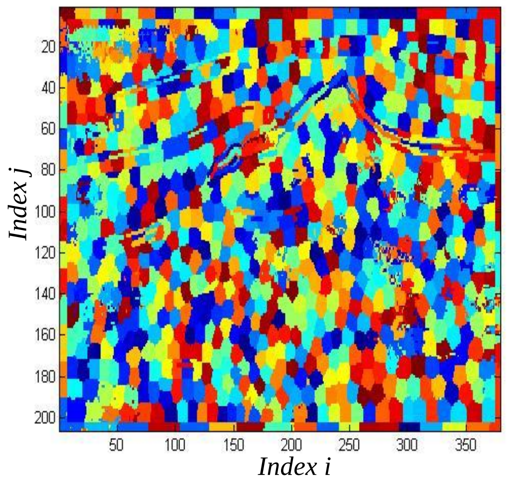
(a) *Number of clusters = 100*



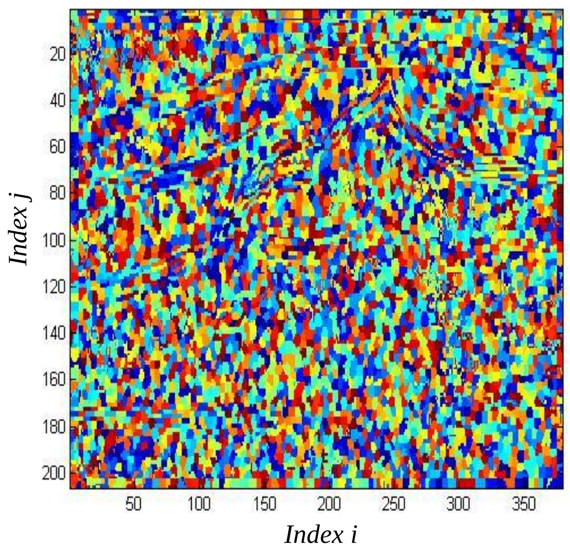
(b) *Number of clusters = 200*



(c) *Number of clusters = 500*

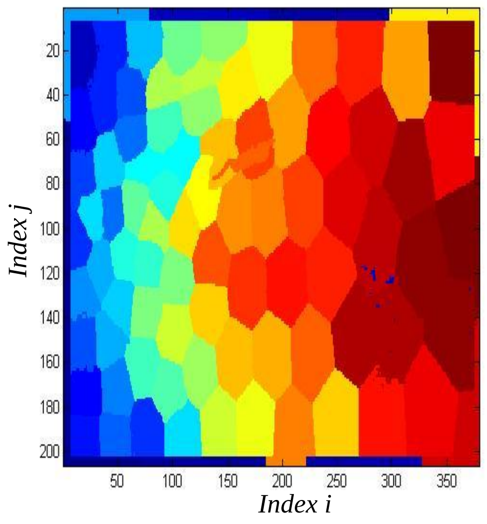


(d) *Number of clusters = 1000*

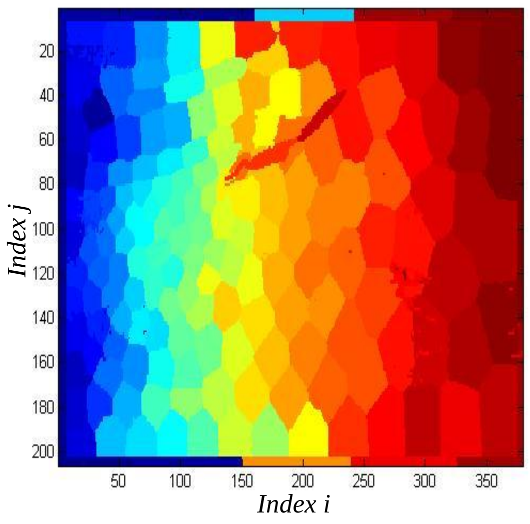


(e) *Number of clusters = 5000*

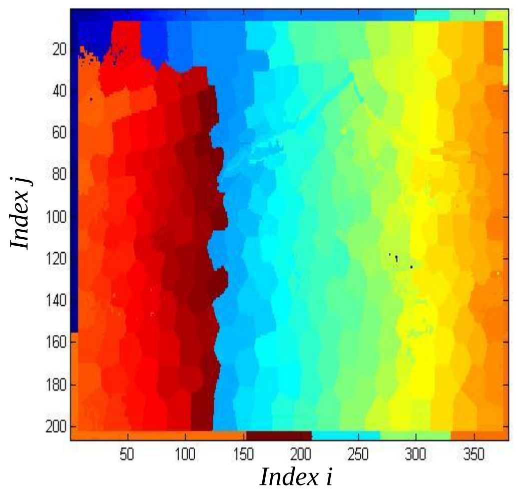
Figure 4.3 *208×381 2D-seismic image after segmentation. Results obtained by code provided by Matlab for k-means figures representing (a) 100, (b) 200, (c) 500, (d) 1000 and (e) 5000 clusters. The segments represent variations on a metric computed by the algorithm and colors are used to visualize such variations.*

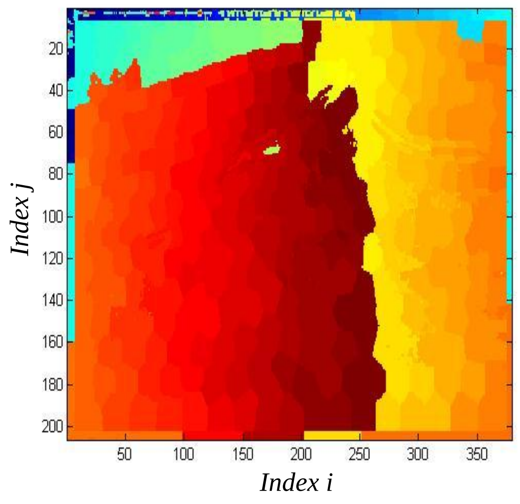


(a) *Number of clusters = 100*



(b) *Number of clusters=200*





(c) Number of clusters = 500   *(d) Number of clusters = 1000*

Figure 4.4 *208×381 2D-seismic image after segmentation. Results obtained by our code for k-means figures representing (a) 100, (b) 200, (c) 500 and (d) 1000 clusters respectively. The segments represent variations on a metric computed by the algorithm and colors are used to visualize such variations.*

4.3.2. Expectation Maximization algorithm:

Both the techniques produced the clusters which clearly indicate the boundary of the salt structure.

For the seismic data in our experiments, k- means algorithm produced better results than the EM algorithms. So, we did not do any further experiment with EM algorithm. The results of the EM Algorithm are shown below.

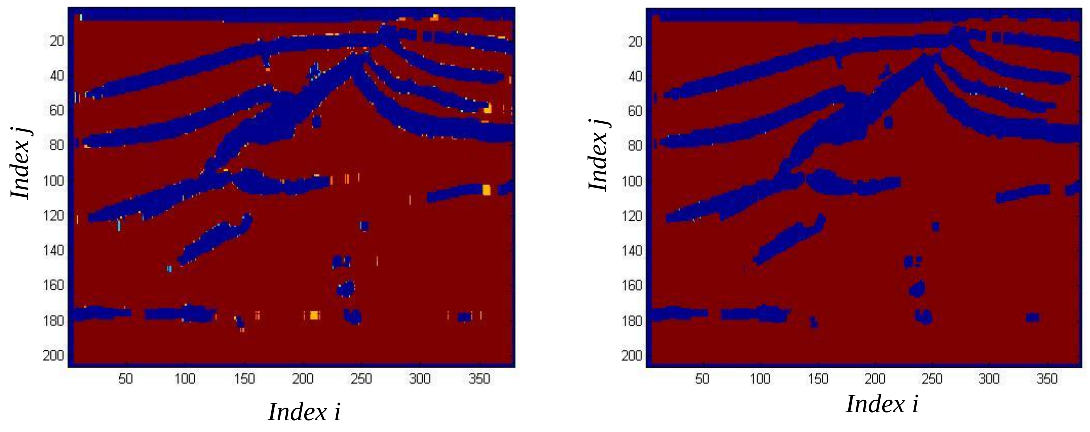


Figure 4.5 *208×381 2D-seismic image after segmentation. Results of EM algorithm with different number of clusters. The segments represent variations on a metric computed by the algorithm and colors are used to visualize such variations.*

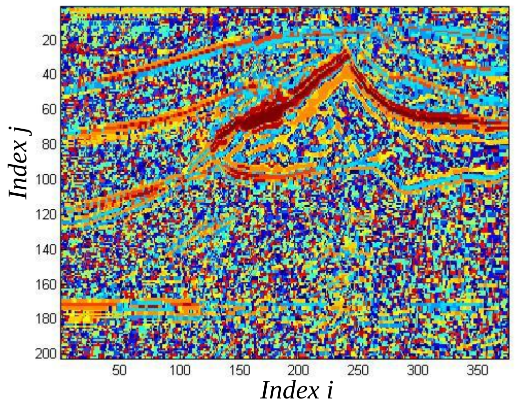
4.3.3. Min-Cut algorithm

While implementing the min-cut algorithm, we tried to represent the entire data set in the form of a graph. This was done by considering each data point as a node and connecting each of these data points to each other with vertices. These vertices were to be given some weight. Since the closer the points are to each other, the more the weight their vertices must have. As a consequence, the weights were an inverse of the distance from one point to another and normalized. Due to the huge amount of data (which is 208 X 381) of 79,248 data points, it seemed unrealistic to connect all the data points to each other. Therefore, to make the algorithm more feasible and efficient, we connected each of the nodes with its closest five neighbors.

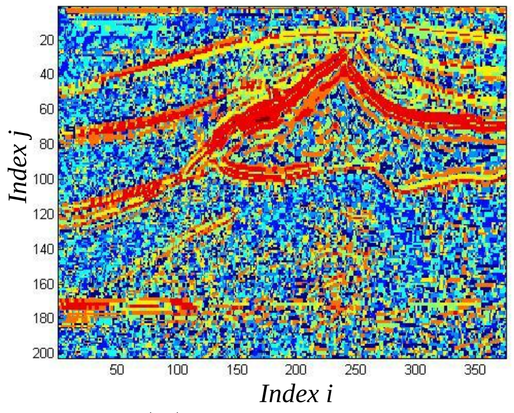
Even though the algorithm seemed overwhelming due to the capability to separate the data into two different classes and extracting a set of points which might be the salt structure, it did not seem to be feasible for large amount of data like seismic data. We tried implementing this algorithm, but the time taken to run these algorithms were not feasible. Moreover we were more concerned of the future implementation of the algorithm in three dimensional data. Hence we did not proceed with this algorithm any further.

4.3.4. Euclidian Clustering algorithm:

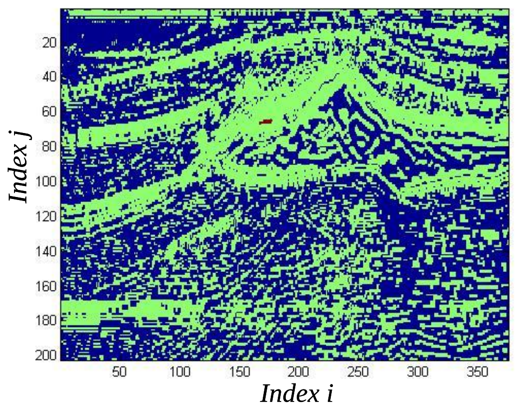
As we mentioned in the previous chapter, we played with the value of 'k' i.e., the number of points that can be in a cluster. Below are the results obtained as we gradually increased the value of k from 5000 to 70,000.



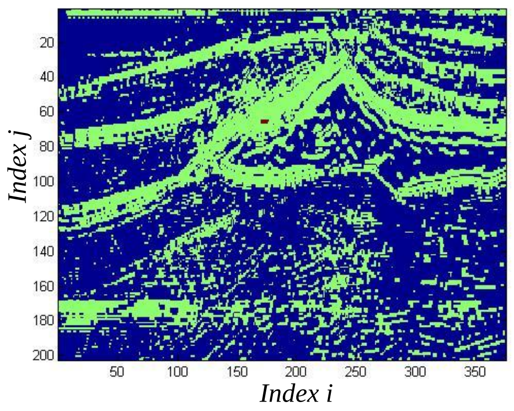
(1) m = 5,000



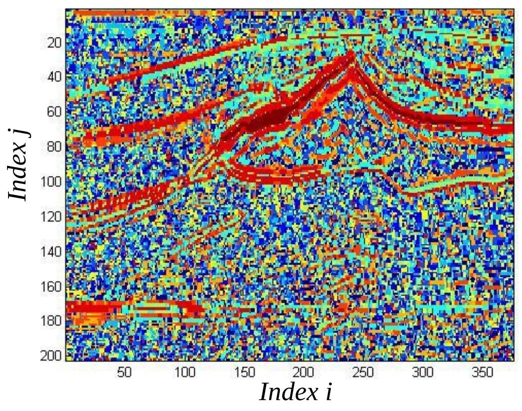
(3) m = 10,000



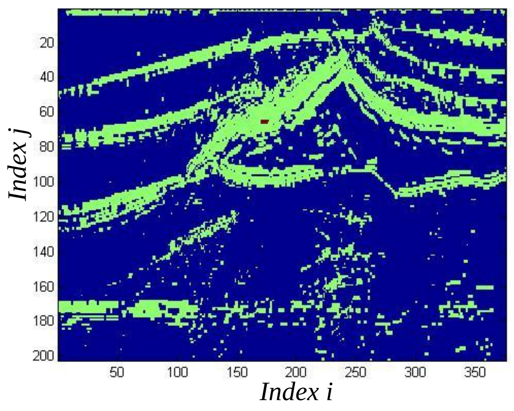
(5) m = 40,000



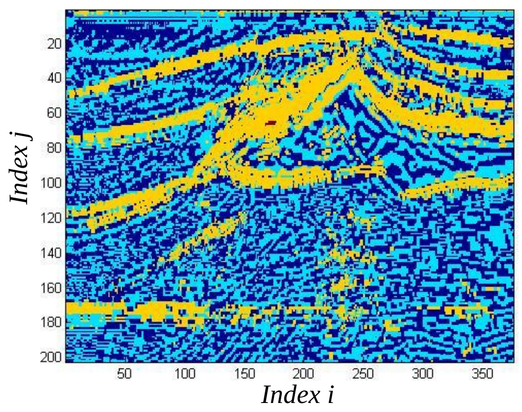
(6) m = 50,000



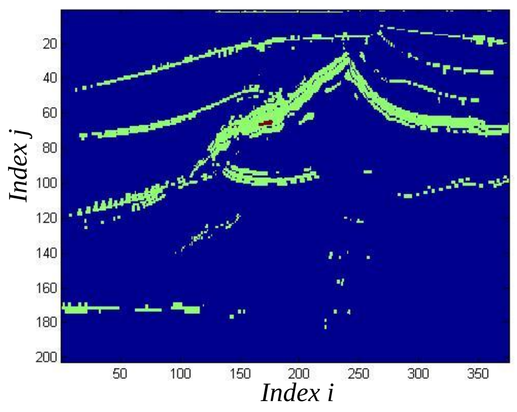
(2) m = 7,500



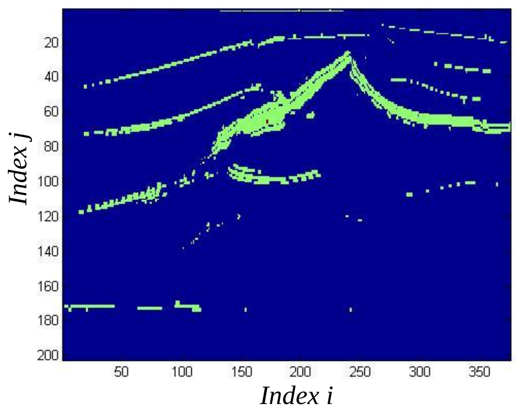
(7) *m* = 60,000



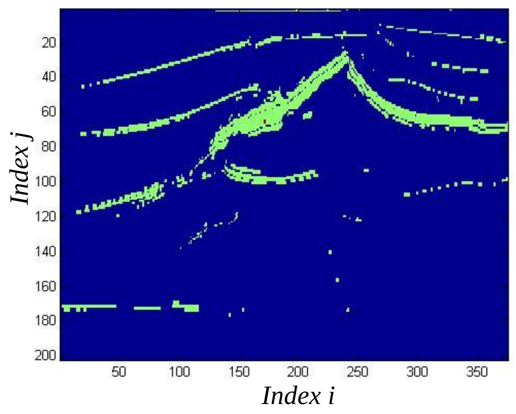
(4) m = 30,000



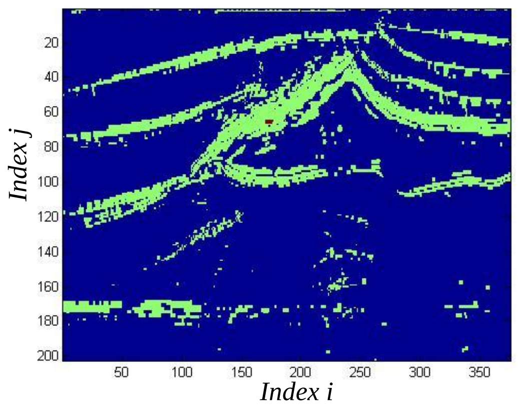
(*9*) *m* = 70,000



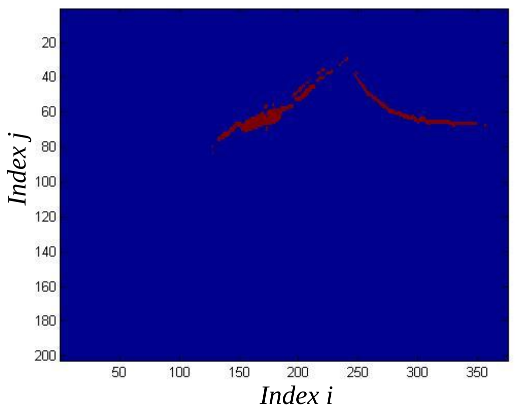
(*11*) *m* = 71, 500



(*10*) *m* = 71,000



(*8*) *m* = 65,000



(12) m = 75,000

Figure 4.6 *208×381 2D-seismic image after segmentation. Results obtained with the gradual increase of the value of m, i.e., the threshold of the number of points a cluster can hold. The segments represent variations on a metric computed by the algorithm and colors are used to visualize such variations.*

We can clearly see from the results above that as we started with a lower value of *m*, where *m* is the number of points the cluster can hold, the algorithm produced many more clusters. Even at that stage we could recognize a cluster that encompassed the surroundings of the salt structure as in the original image. As we gradually increased the value of *m*, the algorithm started dividing the entire dataset into two distinct clusters. With an increase of the value of *m*, we observed the boundary of a mountain like structure. This is the boundary of the salt structure as observed in the original data. With further increase of the value of m, this structure gradually disappeared. This is due to the fact that the algorithm selected a point and puts the closest points into the cluster. Then its puts the points which are closer to these new points added to the cluster but away from the original data point into the cluster. Gradually it puts points further away into the cluster. Hence with a wise decision for the selection of the value of *m*, we can get a clear boundary of the salt structure. Increasing the value of m further led the structure to gradually disappear.

4.4. Conclusion:

From the above results, we can clearly see that the Euclidian clustering algorithm is in the winning position with the k-means algorithm next in rank. Comparing these two, we can conclude that the clusters obtained from the k-means algorithm further need to be classified with the help of a base classifier, whereas the Euclidian clustering technique has the capability to cluster the entire data set into two distinct clusters, one of which is a definite boundary of the salt structure in the data.

Chapter 5

Conclusions

In this dissertation our focus was on extracting the structure of the salt from the rest of the surrounding and to locate any deposit of the salt from seismic images.

Before we started exploring the data, extracting attributes of the raw image was essential to increase the accuracy of our results. The method of feature extraction played an important role in extracting the important information from the data. Not all of the features extracted were very useful as they failed to provide useful information. Eliminating these incompatible and redundant attributes was proved to be critical for achieving a better segregation of the two classes salt and no-salt. Moreover, not only eliminating the weaker attributes, but also ranking and selecting the best attributes was key to a more feasible solution to reduce the curse of dimensionality.

Once the construction of these attributes was established, the next task was selecting the best among these attributes in terms of class separability. This was done with the help of Information Gain. Information Gain makes use of entropy. We select an attribute and choose a suitable point for dividing the data and note the entropy reduced by dividing the data into the two parts than the original entropy of the data. The attributes were ranked according to its power of class separability, i.e. its strength to separate the data into two distinct classes: salt and no-salt.

Most of these attributes are widely used by various seismologists. We tried a few new attributes, including Homogeneity, Contrast and Energy (extracted from Grey-Level Co-occurrence Matrices) which proved to be much better that the conventional attributes used in seismology. The 19 attributes used are as follows mentioned according to their rank performance wise respectively: Homogeneity, Contrast, Energy, Mean, Median, Peaks, Average Energy, Reflection Intensity, Instantaneous Phase, Root Mean Square, Threshold, Chaos Texture, Standard Deviation, Curve Length, Weighted Second Momentum, Variance, Average Non-Linear Energy, Contrast Texture and Zero Crossings.

Once the pre-processing of data was done, the focus was on selection of an efficient technique to segregate the data into two distinct classes: salt and no-salt. We explored the data with four different clustering techniques.

The first among them was the popular k-means algorithm. The k-means algorithm, as the name suggests, tries to find k different clusters with means or centroids of these clusters as the foundation. It takes k, the number of clusters that is to be formed from the user and then randomly selects k points from the dataset. The algorithm then iteratively performs the two steps of assigning all the points in the dataset to the closest centroid and modifying the new centroids by finding the average of the data points assigned to that particular cluster until convergence. This algorithm is both time efficient and space efficient.

The results obtained from this clustering technique were quite impressive as we could observe distinct clusters formed around the structure of the salt body in our test data image. These clusters needs to be further classified into two classes with the help of any base classifier.

The next clustering technique was the Expectation-Maximization Algorithm. The most important difference of expectation-maximization algorithm with the k-means algorithm is that instead of a crisp assignment of the points to any cluster, the expectation maximization algorithm assigns each data point to a cluster with certain probability. The expectation-maximization algorithm assumes that data is a result of probability distributions, which have some parameters and the data can be defined with the help of these probability distributions, and each of these data in the entire dataset comes from these distributions with a certain probability. Each of these distributions actually represents clusters and hence the problem stands out to be like each of the data points in the entire dataset belongs to a cluster with a certain probability. The expectation maximization tries to find out the parameters of these distributions i.e. the clusters and then tries to find the probability with which each of these points belongs to these clusters. Hence a crisp assignment of objects to clusters does not happen in the algorithm.

The expectation maximization algorithm has two major steps. It iteratively first tries to estimate the probability distributions (known as the expectation step) and then uses the above estimates to update the parameters of the probability distributions (known as the maximization step). Iteration stops at convergence.

The results obtained from the k-means algorithm showed better performance than the expectation maximization algorithm. That being the case, k-means algorithm was to be preferred over expectation maximization algorithm for analyzing seismic data using clustering techniques.

The next clustering algorithm we implemented was the min-cut algorithm, which is a graph based algorithm. We considered the entire dataset into a graph, where the data points act as the nodes and the vertices are the similarity measure between these nodes. Once the representation was realized, we cut the graph into two different parts, where the cut will have the minimum cost, which is the most tightly connected vertices will remain in the same cluster.

Initially this algorithm seemed to be really promising as it had the capability of dividing the entire data set into two distinct clusters, but in our project the mincut algorithm did not seem to be a feasible solution to the problem. As the data set was big, it did not seem to be a suitable algorithm for huge data like seismic images. Moreover, the application of the same method in three dimensions would be even more challenging.

The last algorithm applied was the Euclidean clustering algorithm. In this algorithm, the challenge was to set the maximum number of points that can exist in one single cluster. The underlying concept of this algorithm is relatively simple. We selected a point randomly. Then we selected the neighbors of this point and put them in an empty queue and the point itself is put into a specific cluster. Then we processed each of these points in the queue one by one and once the points from the queue were processed, they were assigned to the cluster the initial point was assigned to. The points from the queue were processed by finding the neighbors of these points and putting them into the queue. This was done until the sum of the number of points in the queue and the number of points in the cluster crossed the threshold, after which all the remaining points in the queue were assigned to the cluster.

The Euclidean Clustering produced the best results when compared to all other clustering algorithms used. In our analysis, we gradually increased the threshold of the number of points that can belong to a cluster. By doing this, we observed an interesting phenomenon, where after a certain point the algorithm started dividing the entire dataset into two distinct clusters and then as we increased the threshold the algorithm started extracting a distinct boundary of the entire salt structure.

Future Work:

After the experimentation with the different clustering techniques, the next step of the project is to implement the best pre-processing techniques and the best or most suitable algorithms found for the two dimensional data, in a three dimensional space. Since it is easier to work with two dimensional data which requires less computational power, we first implemented the algorithms in two dimensions with the goal in mind to implement them in three dimensions.

Moreover, we would still want to experiment with more classification techniques like clustering them with k-means and then the clusters obtained could be classified into two distinct classes with the help of a base classifier. The goal is to try out different techniques which would classify the entire structure of the salt into a distinct class instead of just the boundary.

Our future aim is to build a classification technique which would have the best applicability to seismic data and automatically analyze the entire salt structure from a three dimensional data. With this advancement, we would be able to more accurately extract the sub-surface salt structure and thereby precisely locate the oil and natural gas reservoir underneath the surface of the earth.

Bibliography

[1] P. Simon, *Too Big to Ignore: The Business Case for Big Data*. Wiley, 2013, p. 89.

[2] M. Banko and E. Brill, “Scaling to Very Very Large Corpora for Natural Language Disambiguation,” *39th Annu. Meet. Assoc. Comput. Linguist.*, pp. 26–33, 2001.

[3] D. C. Cire, U. Meier, J. Masci, and L. M. Gambardella, “High-Performance Neural Networks for Visual Object Classification,” *Comput. Res. Repos.*, 2011.

[4] S. Jose, “Concept Decompositions for Large Sparse Text Data Using Clustering,” *Mach. Learn.*, vol. 42, pp. 143–175, 2001.

[5] A. Coates, P. Baumstarck, Q. Le, and A. Y. Ng, “Scalable learning for object detection with GPU hardware,” *2009 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 4287–4293, Oct. 2009.

[6] R. Raina and A. Y. Ng, “Self-taught Learning : Transfer Learning from Unlabeled Data,” *Proc. 24 th Int. Conf. Mach. Learn.*, pp. 759–766, 2007.

[7] A. Ben-Menahem, “Review A Concise History of Mainstream Seismology : Origins , Legacy , and Perspectives,” *Bull. Seismol. Soc. Am.*, vol. 85, no. 4, pp. 1202–1225, 1995.

[8] M. Barazangi and L. Brown, *Reflection Seismology : A Global Perspective*, vol. 13. American Geophysical Union, 1986.

[9] W. D. Mooney and T. M. Brocher, “Coincident Seismic Reflection/Refraction Studies of the Continental Lithosphere: A Global Review,” *Geophys. J. R. astr. SOC*, vol. 89, pp. 1–6, 1987.

[10] J. F. Claerbout, *Imaging the Earth’s Interior*. Blackwell Science Inc, 1985.

[11] O. Yilmaz, *Series : Investigations in Geophysics , Volume I Michael R . , Series Editor*. Society of Exploration Geophysicists, 2001.

[12] C. H. Bruce, “Smectite Dehydration--Its Relation to Structural Development and Hydrocarbon Accumulation in Northern Gulf of Mexico Basin,” *Am. Assoc. Pet. Geol. Bull.*, vol. 68, no. 1984, pp. 673–683, 1984.

[13] H. O. Woodbury, I. B. Murray Jr., and R. E. Osborne, “Database Diapirs and Their Relation to Hydrocarbon Accumulation,” *Facts Princ. World Pet. Occur.*, pp. 119–142, 1980.

[14] Y. Changqing, “Salt Structure and Its Relationship with Hydrocarbon Accumulation in Jiangling Sag,” *Fault-block Oil Gas F.*, 2004.

[15] M. R. Chmielewski and J. W. Grzymala-Busse, “Global discretization of continuous attributes as preprocessing for machine learning,” *Int. J. Approx. Reason.*, vol. 15, no. 4, pp. 319–331, Nov. 1996.

[16] A. L. Bluma and P. Langley, “Artificial Intelligence Selection of relevant features and examples in machine,” *Artif. Intell.*, vol. 97, pp. 245–271, 1997.

[17] H. Vafaie and K. De Jong, “Genetic algorithms as a tool for feature selection in machine learning,” *Proc. Fourth Int. Conf. Tools with Artif. Intell. TAI ’92*, pp. 200–203, 1992.

[18] J. Macqueen, “Some Methods for Classification and Analysis OF Multivariate Observations,” *Proc. fifth Berkeley Symp. Math. Stat. Probaility*, vol. 233, no. 233, pp. 281–297, 1967.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *J. R. Stat. Soc.*, vol. 39, no. 1, pp. 1–38, 1977.

[20] A. Blum and S. Chawla, “Learning from Labeled and Unlabeled Data using Graph Mincuts,” *Proc. Eighteenth Int. Conf. Mach. Learn.*, pp. 19–26, 2001.

[21] K. Q. Weinberger, J. Blitzer, and L. K. Saul, “Distance Metric Learning for Large Margin Nearest Neighbor Classification,” *Adv. Neural Inf. Process. Syst.*, vol. 18, 2005.

[22] S. Chopra and K. Marfurt, “Seismic Attributes – A Promising Aid for Geologic Prediction,” *CSEG Rec. 2006 Spec. Ed.*, pp. 110–121, 2006.

[23] S. Chopra and K. J. Marfurt, “75th Anniversary Seismic Attributes — A Historical Perspective,” *Lead. Edge*, vol. 70, no. 5, 2005.

[24] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*. McGraw-Hill, Inc, 1995.

[25] O. due Trier, A. K. Jain, and T. Taxt, “Feature Extraction Methods for Character Recognition -- A Survey,” *Pattern Recognit.*, vol. 29, no. 4, pp. 641–662, 1996.

[26] A. H. S. Solberg, E. Morisbak, and L. Gelius, “3D Segmentation of Salt Using Texture Attributes,” *SEG Tech. Progr. Expand. Abstr.*, pp. 1–5, 2012.

[27] S. Chopra, V. Alexeev, and A. Corporation, “Application of Texture Attribute Analysis to 3D Seismic Data,” *Lead. Edge*, no. September, pp. 29–33, 2005.

[28] L. Rokach and O. Maimon, *Data Mining and Knowledge Discovery Handbook*. Springer, 2010.

[29] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, “Graph Clustering and Minimum Cut Trees,” *Internet Math.*, vol. 1, no. 4, pp. 385–408, Jan. 2004.

[30] M. Stoer and F. Wagner, “A simple min-cut algorithm,” *J. ACM*, vol. 44, no. 4, pp. 585–591, Jul. 1997.

[31] S. N. Sinha, “Graph Cut Algorithms in Vision , Graphics and Machine Learning An Integrative Paper,” *UNC Chapel Hill*, 2009.