A Hybrid Data Mining Approach for Hyperspectral Images Analysis

GRADUATE PROJECT PROJECT

Submitted to the Faculty of
the Computer Science Program
Texas A&M University - Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Xinyi Wang
Summer 2014

Committee Members

Dr. Longzhuang Li ________________________
Committee Chairperson

Dr. Scott A. King ________________________
Committee Member

ABSTRACT

Hyperspectral images are a series of image frames in different wavelength bands which form a cube of data with both spatial and spectral information. Data mining is to mine the useful data from the huge datasets. Since the pixels of hyperspectral images are all in very high dimensional space, data mining for hyperspectral images, or also called hyperspectral image classification, is quite challenging. A different approach with more process steps is introduced for better accuracy. The system contains hyperspectral data understanding, data preparation, vegetation detection algorithm, the k-means algorithm, and support vector machine. Data understanding is the first step to let people know about the datasets. Data preparation is the basic step for input of data mining algorithms because some of the algorithms need to have a specified format. Vegetation detection algorithm is to mine the useful data for the k-means algorithms. The k-means algorithm can generate some feature for support vector machine (SVM). SVM is to apply the features from the previous step and to mine the data, and finally some accuracies are generated according to those processes of hyperspectral imaging dataset, which helps us test the model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Information age is a popular term to describe the world we are living in. However, what is information made up of? Actually it is a wide range of data. Data can be huge, but not all of them are useful for us. How do we abstract the useful data from it? A correct data mining technique is therefore important for numerous practical purposes, especially for the huge dataset. Data mining is currently being exploited in many applications, such as games, businesses and sciences.

In this study, a hyperspectral imaging system is used to obtain the hyperspectral images of some mud from the Ocean. To mine the significant objects (such as algae) from these mud samples is the main purpose here. Finding the algae is good for environment protection, especially for coral protection. The general steps in this project are hyperspectral data understanding, pre-processing, unsupervised learning, feature selection, supervised learning (post-processing), and rules or decision. Data understanding is the first step to let people know about the datasets. Pre-processing is the basic step for data cleaning because the experimental datasets contain lots of noise or even missing data. Unsupervised learning is the next step to mine the data from the cleaned data by using K-means. Feature selection is the step right after unsupervised learning which can generate some feature after data mining algorithm. Supervised learning, also known as post-processing is to apply the features from the previous step and to mine the data, and finally some rules or decision is generated according to those processes of hyperspectral imaging dataset. These data mining algorithms are presented for hybrid techniques. The final goal of this paper is to test more data mining models for hyperspectral image data, including more processing in order to improve the accuracy, and finally provides some conclusions and recommendations regarding these methods.

1.1 Statement of the Problem

It is important for researchers to abstract the useful information from the huge dataset in any field. The question can be summarized as follows:

Given a dataset that contains information in the form of hyperspectral variables that are believed to significantly influence the objects, what are the most appropriate data mining techniques to determine useful information with a reasonable margin of error?

To answer this question, this study explores several data mining methods that might be used to determine the significant objects by using data of hyperspectral images on mud samples. The main goal of this work focuses on studying the currently available applicable data mining tools (such as the k-means algorithm and support vector machine) for finding out the significant objects.

Hyperspectral image classification is the method used to abstract information from images. Finding the optimal approach for hyperspectral image classification for a given application becomes a popular research problem for computer scientists.

Hyperspectral images are a series of image frames in different wavelength bands which form a cube of data with both spatial and spectral information. In each pixel of the hyperspectral image, wavelength bands versus raw intensity counts (arbitrary units) can be obtained and graphed. Different objects have different spectral information in certain wavelength bands. Therefore, the goal is to understand the microdistribution of organisms, by analyzing the plots. Figure 1.1 gives you the idea of what a hyperspectral image look like.

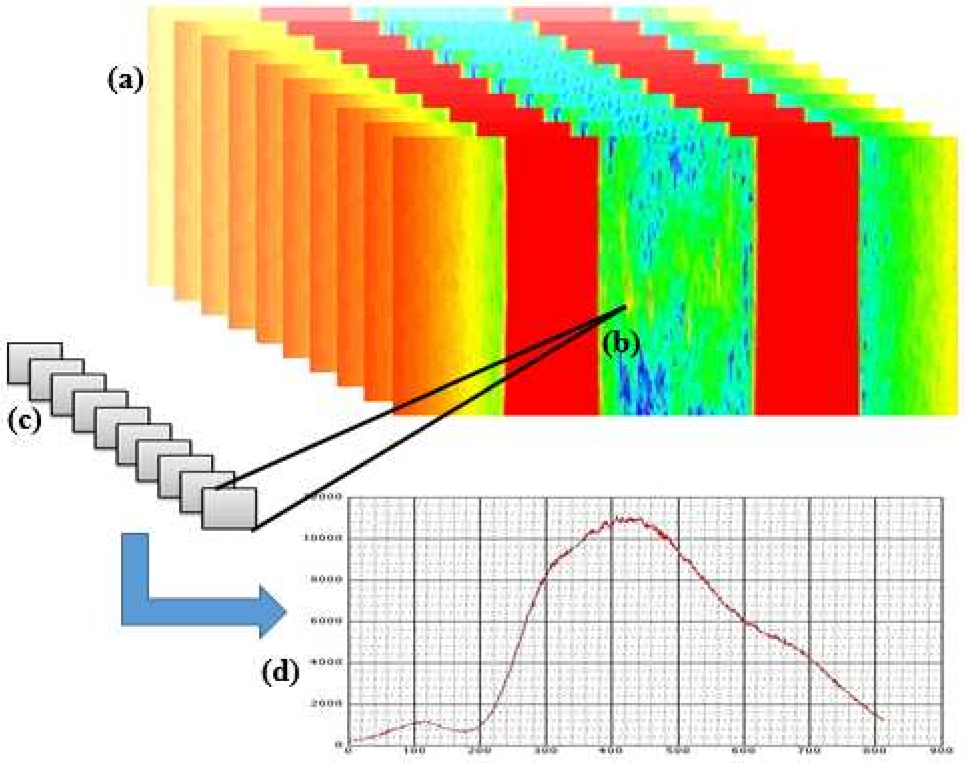


Fig. 1.1. (a) A hyperspectral image of gold nanoparticles. (b) A single pixel location from the hyperspectral image. (c) The pixel at different frames with the same location. (d) The spectrum of the single pixel location, corresponding to a series of image frames at different wavelength.

The essence of hyperspectral image classification using spectral analysis in this project involves processing data through data mining. Data mining refers to finding patterns in large data sets, such as the hyperspetral images, which can be as large as 20 Giga Bytes (GB) each. Several algorithms have been defined in the data mining domain such as decision trees, artificial neural networks, K-means, and support vector machine (SVM) algorithm. Determining and building an efficient and accurate classifier based on the optimal data mining algorithm is one of the objectives of this mud hyperspectral image classification project.

In this study, the hybrid of pre-processing, unsupervised learning and supervised learning is presented for hyperspectral image analysis and surface classification from mud samples. In this system, the classifier has three layers: the first layer is for preprocessing. The medial layer is unsupervised learning, the k-means algorithm. The last layer represents supervised learning, support vector machine.

1.2 Review of the Literature

Since entering in the space age in 1957, people keep documenting the image data of Earth from outer space [1]. The image information can be analyzed for different purposes, such as military uses, agriculture, and urban planning [2]. Hyperspectral images classification is more challenging because of the very high dimensionality of the pixels and the small number of labeled examples typically available for learning [3].

1.2.1 Hyperspectral Image

Because human eyes only can see the light under bands from 390nm to 750 nm, there are requirements for detecting visions, whose bands are below 390 nm or beyond 750nm [4]. For example, the gold nanoparticles are actually spherical organic coated particles, and dissolved in toluene. Then the gold nanoparticles dissolved in toluene were further diluted in different concentration of 14%, 28% and 50% in toluene. Human eyes are mostly sensitive at around band of 555 nm.

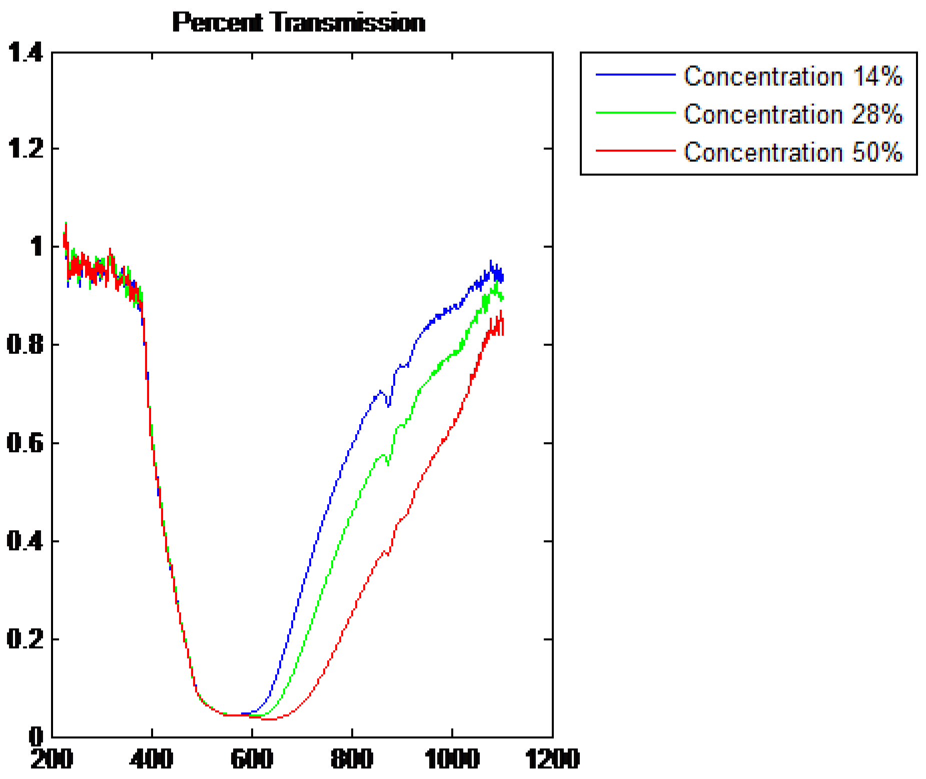


Fig. 1.2. Percent Transmission of three different gold nanoparticle concentrations.

From Figure 1.2 mentioned, it shows that the figures of three samples overlap each other at around band of 555 nm but split up at band of 600 nm or above. It means that it is hard to identify the concentration of three samples by human eyes. If the research is extended beyond the visible, there are some significant differences between the samples.

1.2.2 The Existing Approaches

There are many existing approaches available for hyperspectral classification. The k-means algorithm and support vector machine are getting more attention because of the complexity of high dimensionality of hyperspectral images. Most of approaches are based on them with more process involved. A neighborhood-constrained k-means (NC-k-means) algorithm [13] is developed to incorporate the pure neighborhood index into the traditional k-means algorithm. A parameter-free k-means algorithm [14] is proposed in 2012, which is based on co-occurrence matrix scheme. Support vector machines for hyperspectral image classification with spectral-based kernels [6] are introduced in 2003. A support vector classification method for hyperspectral data based on spectral and morphological information is proposed in 2011 [2]. The semi-supervised classifier named PSO-SVM, which is combined by support vector machine (SVM) and particle swarm optimization (PSO), was proposed in 2010 [16]. They focus their results on either the accuracy or the running time, which means that they either get a better accuracy with a time-consuming system or reduce the running time to get a reasonable accuracy. Therefore, here comes our hybrid system with a reasonable accuracy and running time.

1.3 Methodology

Going back to the original research question: What data mining techniques can be used to find out the significant objects? The literature review in the previous section indicated different alternatives to answer this question. This study selects the techniques that were most successfully used in the literature. The real hyperspectral images of mud sample are used for solve the problem by using these different techniques and their results are compared.

A general data mining is used to find out the significant objects for mud samples. Chapter Two shows the technique of identifying how differences between the visible red and near-infrared bands of the mud sample image, which we can use this technique to identify the areas having the significant objects; k-means, a data mining technique, is an unsupervised learning, which means we can find out the classes or groups, and label them for the final data mining process, SVM; and the SVM technique that takes into account the classes or groups in training the unknown dataset, providing a prediction of each groups. Chapter Three shows how the approaches work and the system architecture. Chapter Four focuses on experimental results from testing a hyperspectral image. Finally, Chapter Five discusses these data mining techniques and provides conclusions and recommendations regarding these methods.

1.4 The sample

The hyperspectral image of the mud sample is taken from Hyperspectral Optical Property Instrumentation (HOPI) Lab at Texas A&M University —Corpus Christi. Figure 1.3 shows the photo of the mud sample.

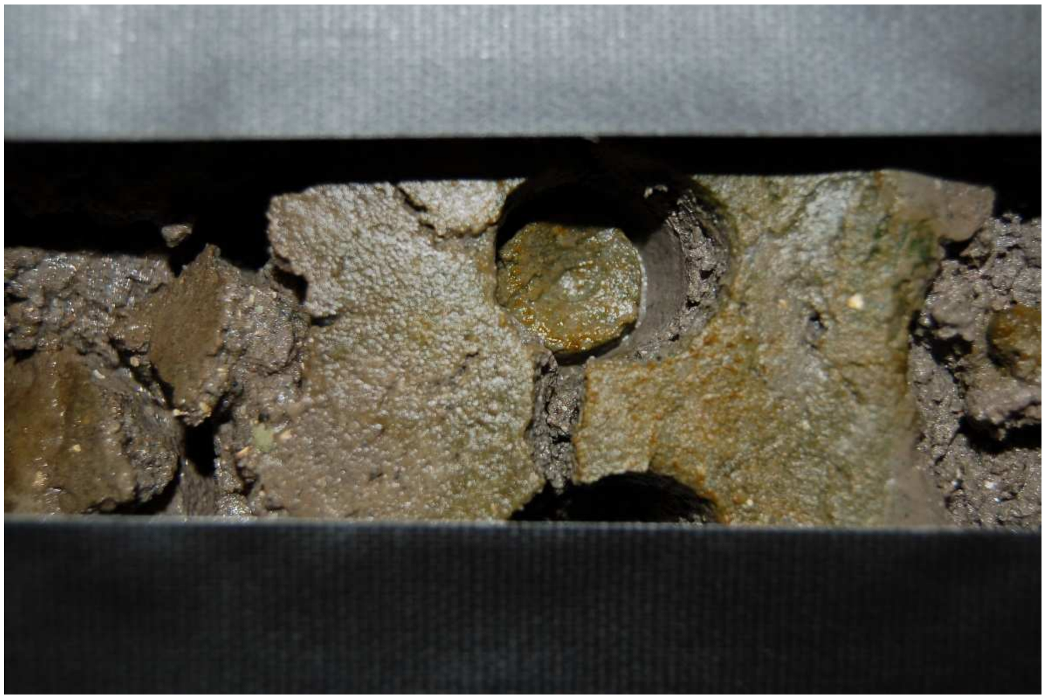


Fig. 1.3. The photo of the mud sample.

Figure 1.4 shows the hyperspectral image of the mud sample in MATLAB with gray scale and original axis.

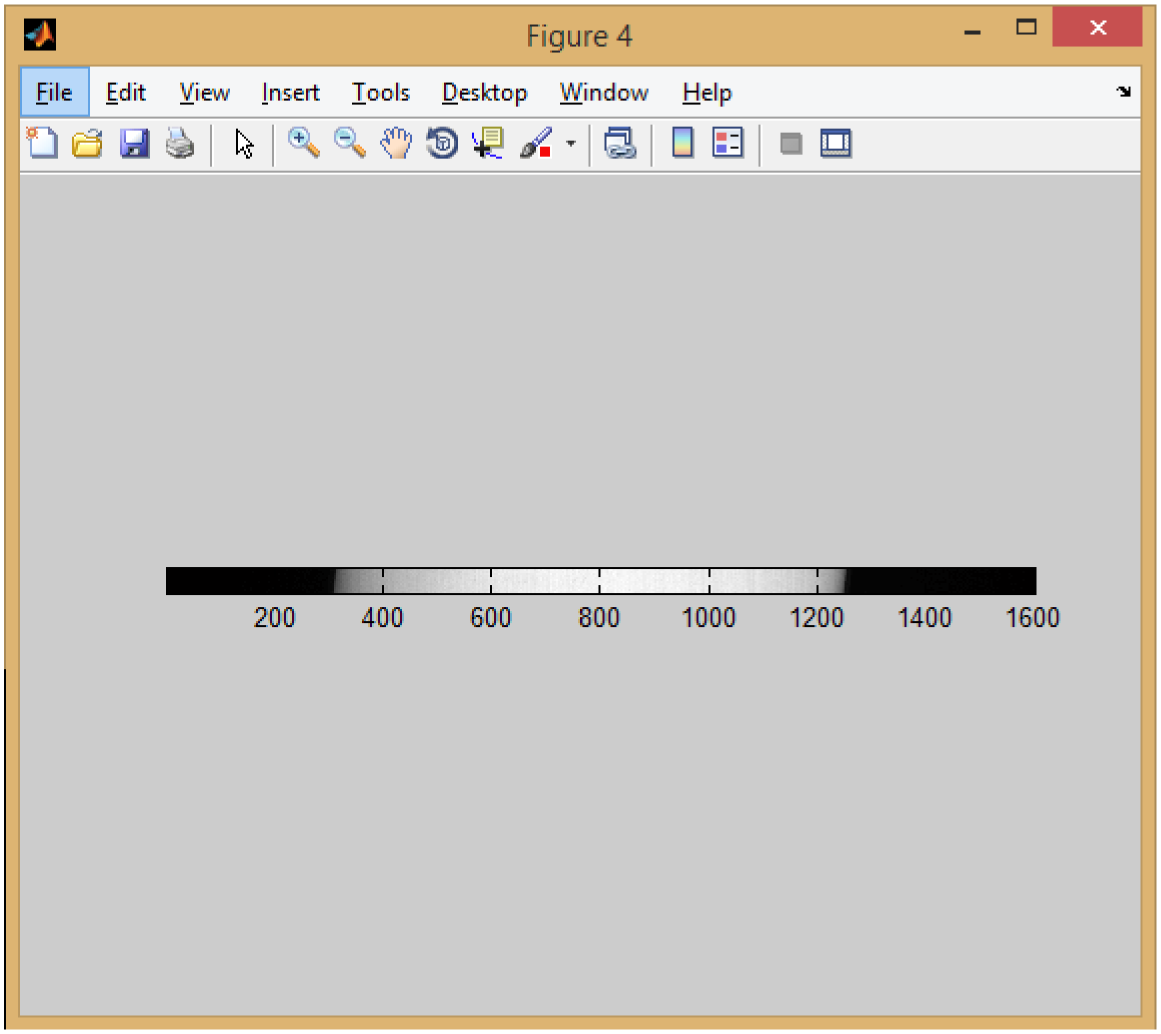
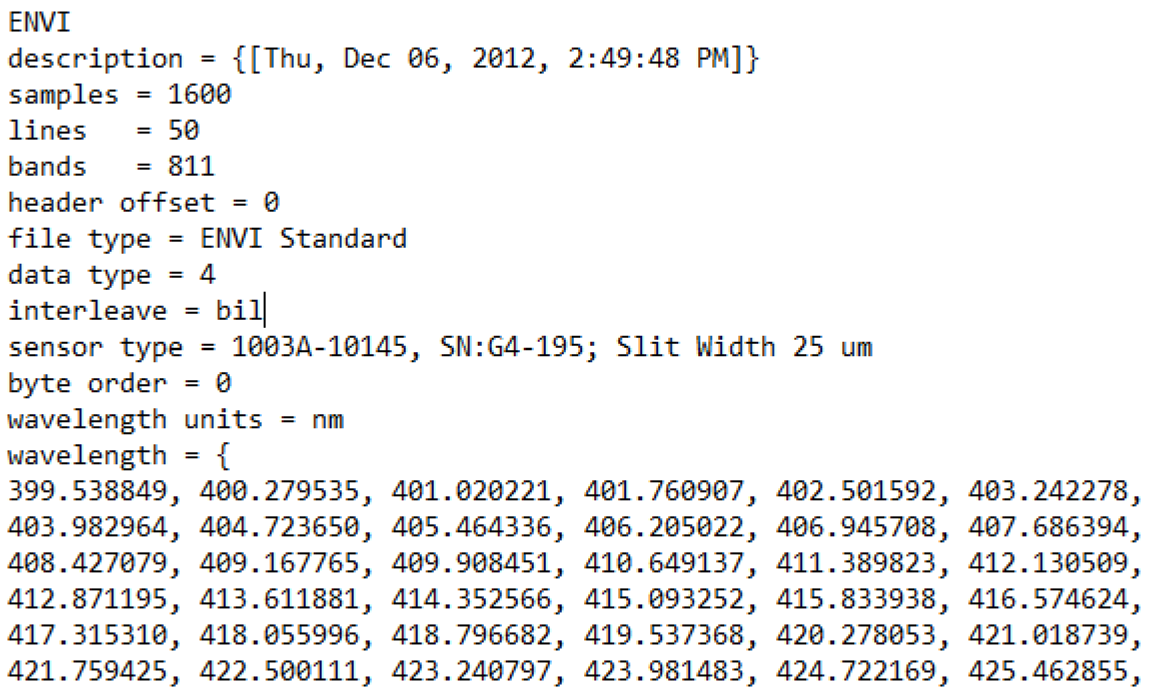


Fig. 1.4. A hyperspectral image of a mud sample.

1.5 Exploring the Dataset

Each hyperspectral image has its own header file, which records the date of taking the image, samples, lines and bands. The important information is saved here for setting up the parameter when the image is processed. Figure 1.5 shows what the header file includes. The wavelength is continuously increased from 399.538849 nm to 999.494408 nm by 0.740686 nm.



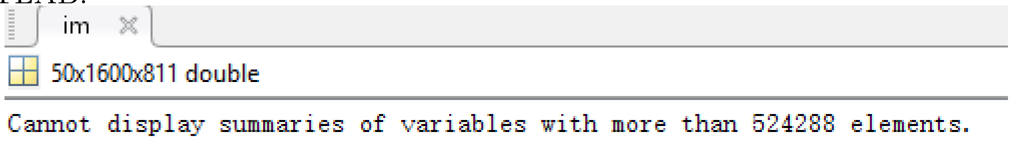
```
ENVI
description = {[Thu, Dec 06, 2012, 2:49:48 PM]}
samples = 1600
lines   = 50
bands   = 811
header offset = 0
file type = ENVI Standard
data type = 4
interleave = bil
sensor type = 1003A-10145, SN:G4-195; Slit Width 25 um
byte order = 0
wavelength units = nm
wavelength = {
399.538849, 400.279535, 401.020221, 401.760907, 402.501592, 403.242278,
403.982964, 404.723650, 405.464336, 406.205022, 406.945708, 407.686394,
408.427079, 409.167765, 409.908451, 410.649137, 411.389823, 412.130509,
412.871195, 413.611881, 414.352566, 415.093252, 415.833938, 416.574624,
417.315310, 418.055996, 418.796682, 419.537368, 420.278053, 421.018739,
421.759425, 422.500111, 423.240797, 423.981483, 424.722169, 425.462855,
```

Fig. 1.5. The header file of a hyperspectral image of a mud sample.

From Figure 1.4, it shows that a hyperspectral image can be imported into MATLAB through the code below:

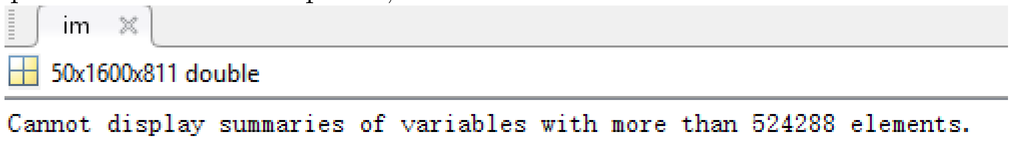
```
im = multibandread(fn, [lines,samples,bands],'single',0,'bil','ieee-le');
```

Once processed, the saved hyperspectral image is shown in the workspace of MATLAB:



```
im
50x1600x811 double
Cannot display summaries of variables with more than 524288 elements.
```

The size of the hyperspectral image of mud sample is 50-by-1600 with 811 bands in depth. Once the im is opened, it is said that:



```
im
50x1600x811 double
Cannot display summaries of variables with more than 524288 elements.
```

To process this hyperspectral image, data preparation in each chapter is always the first and important step to deal with. The Figure 1.6 is the hyerpsectral image of mud sample in MATLAB with color and square axis. Next, all the related algorithm or techniques is shown in Chapter Two.

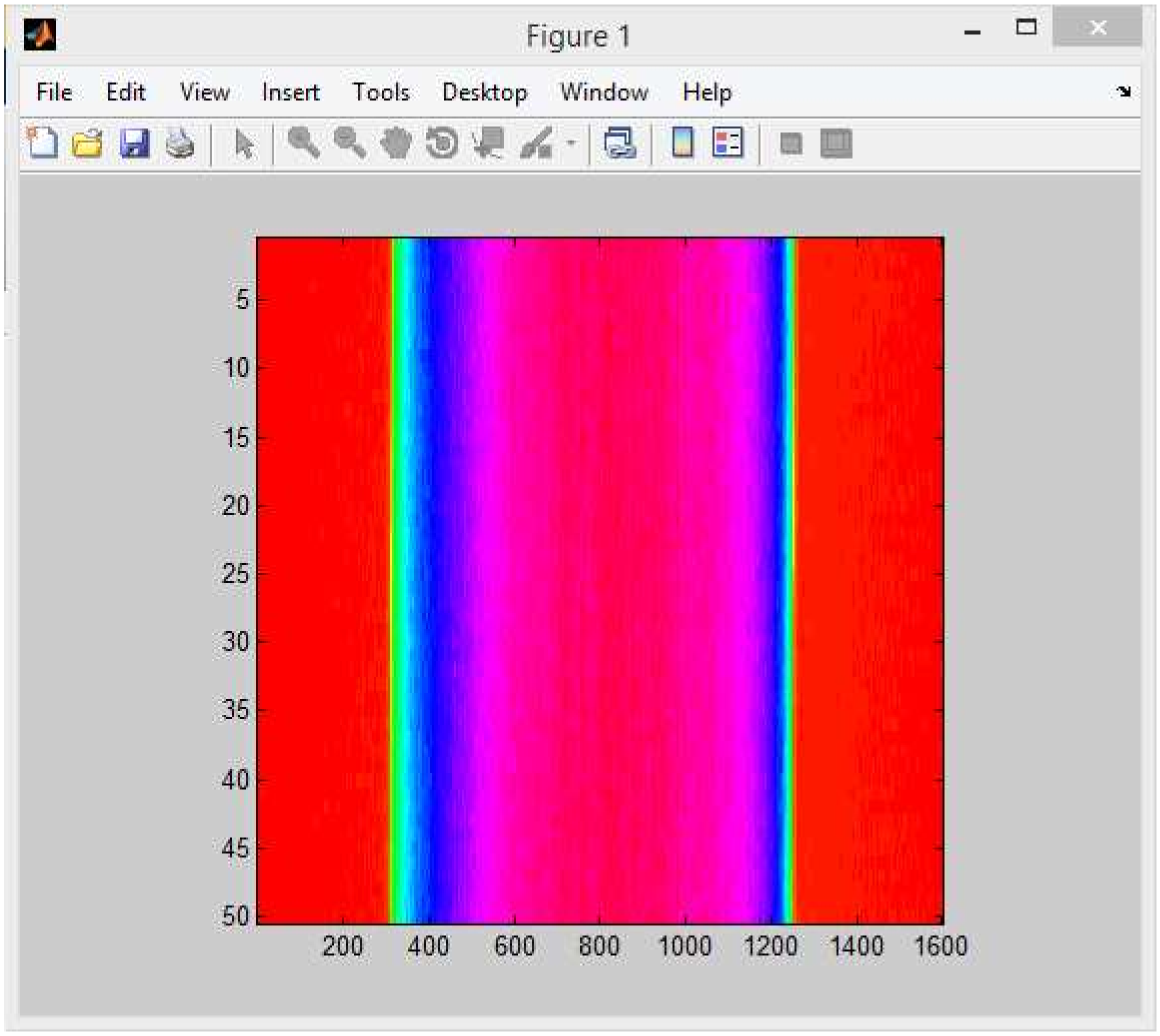


Fig. 1.6. The hyperspectral image of mud sample in MATLAB with color and square axis.

2. RELATED WORK

The related techniques or algorithms are vegetation detection algorithm, the k-means algorithm and support vector machine. Algorithm for vegetation detection can detect the hyperspectral image for the purpose of predicting the percentage of significant objects. The k-means algorithm, a centroid-based technique, will help us find out the clusters. The k-means algorithm is a cluster analysis algorithm, which mainly calculates data aggregation by constantly computing the average distances between each points and mean point. Machine learning is essentially an approximation of the true model of problems, which means the best approximate mode that is used for the true model is called a hypothesis. So there is no doubt that the true model is certainly unknown. Otherwise, the true model can just be used and there is no point to have machine learning. Since the true model is unknown, how much differences between the hypothesis and the real solution are also unknown. However, if some attributes or instances are known, they can be used to approach to the real model. Support Vector Machine is used to find the errors, approach the true model and predict the unknown model.

2.1 The k-means algorithm for Hyperspectral Images

The k-means algorithm is one of the major techniques to be used to classify the hyperspectral images. Several approaches related to the k-means algorithm are introduced for pursuing a better classification or a better accuracy.

A neighborhood-constrained k-means (NC-k-means) algorithm [13] is developed to incorporate the pure neighborhood index into the traditional k-means algorithm. A neighbourhood is defined as a set of contiguous pixels in a certain window of an image. The size of a window is usually 2-by-2 pixels, 3-by-3 pixels, or 4-by-4 pixels. According to the definition of neighbourhood, a binary function called pure neighbourhood index

(PNI) is defined by whether a neighbourhood is a pure neighbourhood of class k or a mixed neighbourhood of class k. The very first step is to classify the images into k clusters and initialize a set of centroids like the normal k-means algorithm does. Once the partition matrix is computed to classify all pixels to initial cluster k, the neighbourhood window scale can be set up. The next step is to calculate the spectral vector of each neighbourhood and determine the value of the neighbourhood label. Then PNI is used to calculate all neighbourhoods, and determine the total number of pixels in pure neighbourhoods. The final step is iteratively to calculate the k-means objective function and adjust the centroid of each cluster k, cluster type of each pixel and cluster type of each neighbourhood. This approach definitely reduce the interference of noise and the heterogeneity to have a better performance compared to the k-means algorithm. The NC-k-means algorithm with neighbourhood window size of 2-by-2 do not have the best results in any cases. The window size must be 3-by-3 or 4-by-4 to have a better performance in some cases. Therefore, the results of the NC-k-means algorithm is not always better than the k-means algorithm.

A parameter-free k-means algorithm [14] is proposed in 2012, which is based on co-occurrence matrix scheme. This method can be used to output the number of desired clusters. The input of this method is a dataset containing n objects. The process of the method is to transform the dataset images to gray scale images, and then transform them to co-occurrence matrix. The next step is to find out a diagonal matrix from the results of co-occurrence matrix. The final step is to determine the k value by finding out the number of local maximums. Since the number of clusters of this parameter-free method always has less than one of other algorithms, the algorithm is not sure to have enough precision. The running time of this method is not mentioned in the article, so it is hard to compare it with other methods.

2.2 Support vector machine for Hyperspectral Images

Due to the high dimensionality of the hyperspectral images, many data mining techniques can not have the reasonable performance. However, support vector machines (SVM) can still maintain a reasonable classification accuracy.

Support vector machines for hyperspectral image classification with spectral-based kernels [6] are introduced. The difference between spectral kernels and usual kernels is that the spectral kernels are based on a quadratic distance evaluation between two samples. Spectral angle (SA) is defined to measure the spectral difference, and the spectral information divergence (SID) is also defined to measure the spectral difference with a different equation. Mixture between spectral-based and quadratic-based kernels is also developed to classify the images. However, the results are not given to us as an accuracy but a figure. It is really hard to confirm which kernel perform the best without the accuracies.

A support vector classification method for hyperspectral data based on spectral and morphological information is proposed in 2011 [2]. The very first step is to apply SVM-recursive feature elimination (RFE) to the original data to obtain the most discriminative spectral subset S_i. The second step is to construct the extended multivariate morphological profile (EMMP) with the subset S_i to obtain $EMMP_i$. The third step is to apply nonparametric weighted feature extraction (NWFE) to the $EMMP_i$ to extract feature $MORP_i$. The fourth step is to apply NWFE to the orightnal hyperspectral data to extract feature $SPEC_i$. The next step is to fuse the $MORP_i$ and the $SPEC_i$ to obtain MS_i. The final step is to use SVM classification based on the MS_i. A feature selection algorithm can construct the EMMP more effectively. The classification accuracy is improved by the feature of the EMMP, but to construct the EMMP is very time-consuming.

A comparison of support vector machine, import vector machines and relevance vector machines for hyperspectral classification is proposed in 2011 [15]. There are two of the proposed enhancements, import vector machines (IVM) and relevance vector machines (RVM), compared to the support vector machine. IVM is similar with SVM in the curve shape and the negative logarithmic likelihood of the binomial distribution. RVM is introduced as the Bayesian principle to the SVM concept. The dataset in this research is a small training data set and a large number of classes. Even though IVM and RVM outperform the SVM, the authors still say that more datasets should be involved to confirm the results.

The semi-supervised classifier named PSO-SVM, which is combined by support vector machine (SVM) and particle swarm optimization (PSO), was proposed in 2010 [16]. This approach inflates the unlabeled samples of hyperspectral images to obtain a result with higher accuracy. The particle estimates the parameter values and the candidates of the labels of the randomly selected unlabeled samples. The fitness function generated by PSO, is designed based on two concepts: the classification accuracy and the ratio calculated by the number of supported vectors divided by the number of samples. Most of the particles have a high accuracy classification. However, a small number of them with a low fitness value influence the result the most. First, find the candidate labels for the unlabeled samples trough a SVM trained by the labeled samples. Second, generate the particles from the unlabeled samples. Third, train SVM classifiers for each particles to inflate the samples. Fourth, apply the fitness function to particles. Fifth, classify the result with the test data. This approach has a better performance compared with other techniques. The RBF kernel function is only adopted in the SVM model, it is better to test with other kernel functions

2.3 Discussion

There are also other approaches to either achieve the goal of faster support vector classifications [11] or achieve the goal of improving the accuracy of support vector machine [12]. The first approach to build up a SVM-tree and decompose the original problem into many linear subproblems with non-linear extensions. Every subproblem is treated as a linear case of SVM at first. If they think they have a good result of that subproblem in the linear case, they will not process anymore in non-linear case. Therefore, they do reduce the running time of processing the large dataset, but this definitely affects the accuracy. In the article, they also claim that they achieve just reasonable classification results. The second approach is about increasing the accuracy by tuning the parameters of support vector machine manually. Considering the average running time of SVM and k-means, finding out the appropriate parameters for SVM is more costly than finding out the appropriate clusters in k-means. Therefore, even though they improve the accuracy of SVM, the running time of SVM is more concerned if a large dataset is processed. Therefore, a hybrid system is formed here to pursue a better accuracy with a reasonable running time. Algorithm for vegetation detection is the first step to detect percentage of significant objects of the hyperspectral image. The k-means algorithm, a centroid-based technique, is considered as the second step to help us identify the location of clusters. Support Vector Machine is used to be the last step to find the errors, approach the true model and provide the accuracy of this model.

3. THE HYBRID SYSTEM DESIGN

Data preparation is the very first step of the algorithm. Figure 3.7 shows a diagram of the system architecture. The hyperspectral dataset is trained by using algorithm for vegetation detection, the k-means algorithm and support vector machine, and an unknown dataset is tested to get the accuracy by using the test mode, 10-fold cross-validation.

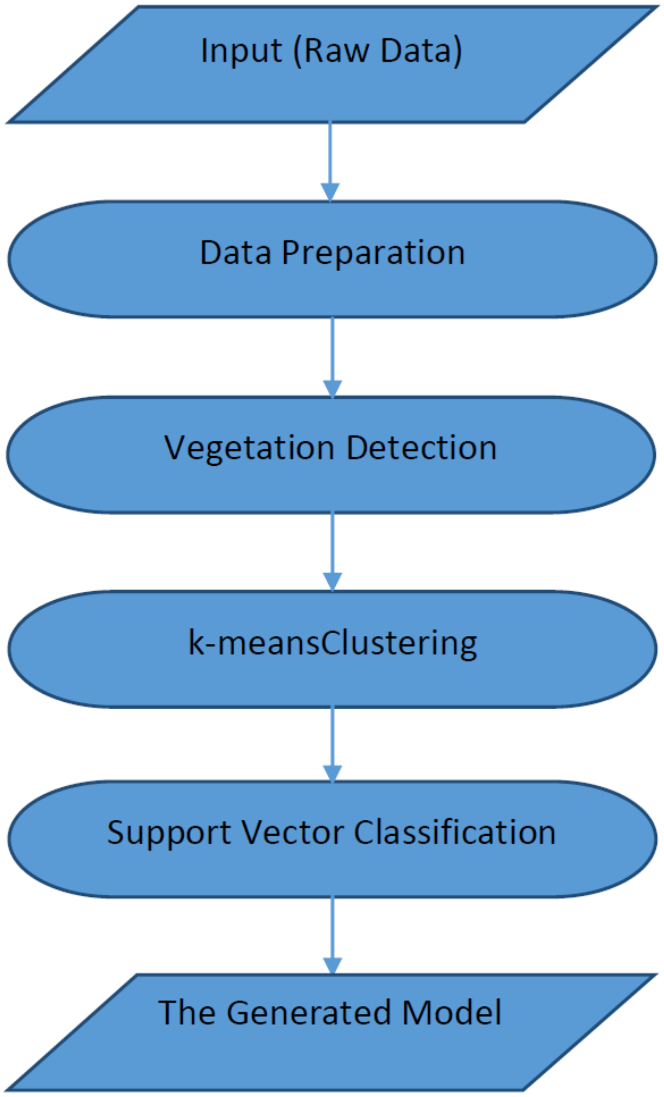


Fig. 3.7. The system architecture.

In this hybrid system, the number of clusters is not required as input because it can be predicted by the k-means algorithm. Using the k-means algorithm is relatively to reduce the running time of the entire system. The procedure of this hybrid system is data preparation for each algorithm; determine the percentage of the significant objects by using algorithm for vegetation detection; find out the clusters or groups of the hyperspectral image by using the k-means algorithm; and finally generate the training model and find out the accuracy by using support vector machine. Some experimental results are shown in the next chapter according to this system architecture.

3.1 Data Preparation

The dataset is an 811-band 50-by-1600 hyperspectral image of a mud sample, whose pixel values are band interleaved by line (BIL) in order to increase band number. BIL is one of three primary methods for encoding multiband image data, which has the advantage of accessing the spatial and spectral data easily.

Since the hyperspectral image is a three dimensional dataset, it is not mathematically convenient to operate for the further data mining. Therefore, the three dimensional dataset is reformatted into two dimensional dataset by having each column display the entire pixel values of each layer or band, so we make a new 80000-by-811 dataset.

3.2 Vegetation Detection Algorithm

This algorithm tells us how to find out the differences between the visible red and near-infrared bands of a Landsat image and the percentage of the Vegetation. To predict the percentage, we need to find out the differences between the visible red and near-infrared bands of the hyperspectral image. This technique identifies the pixels or area that contains the significant objects, such as algae.

To achieve this result, there are six steps in this original algorithm [7], shown in Figure 3.8:

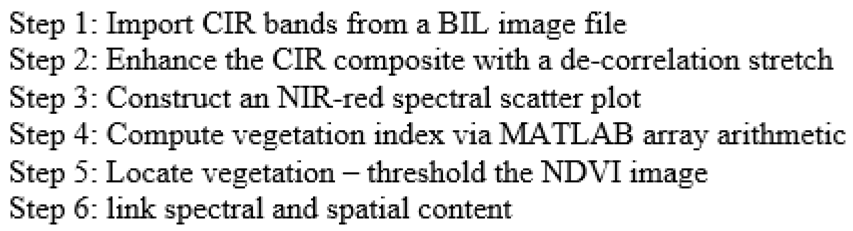

Step 1: Import CIR bands from a BIL image file
Step 2: Enhance the CIR composite with a de-correlation stretch
Step 3: Construct an NIR-red spectral scatter plot
Step 4: Compute vegetation index via MATLAB array arithmetic
Step 5: Locate vegetation – threshold the NDVI image
Step 6: link spectral and spatial content

Fig. 3.8. Vegetation detection algorithm.

The very first step is to select the three most informative bands from the mud sample file. To determine these three bands, we need to find out how many 0s are in each band. Therefore, by figuring out the three least 0s bands, we can know the three bands we want. With the three bands, we can continue to run our algorithm to find out the percentage of the significant objects.

The color-infrared (CIR) composite is the result when the near infrared (NIR), the visible red and the visible green are mapped into the red, green and blue planes of an RGB image. To read the hyperspectral image, the MATLAB function multibandread can help us handle it in a special form that we can recognize. Therefore, when we use the MATLAB function multibandread from the three determined bands, we have an RGB image, which red means the NIR band, green means the visible red band and blue means the visible green band. However, still the contrast is not as good as we want.

Next, we will construct an NIR-Red scatter plot, which helps us compare the NIR band (displayed as red) and the visible red band (displayed as green). As we said above, we can easily extract the NIR and red bands from the CIR composite. So we can see the difference between these two bands in grayscale images.

To find out the significant objects, we have to compute the index of each pixel:

$$\text{Index} = (\text{NIR - red})\ ./\ (\text{NIR + red}) \tag{3.1}$$

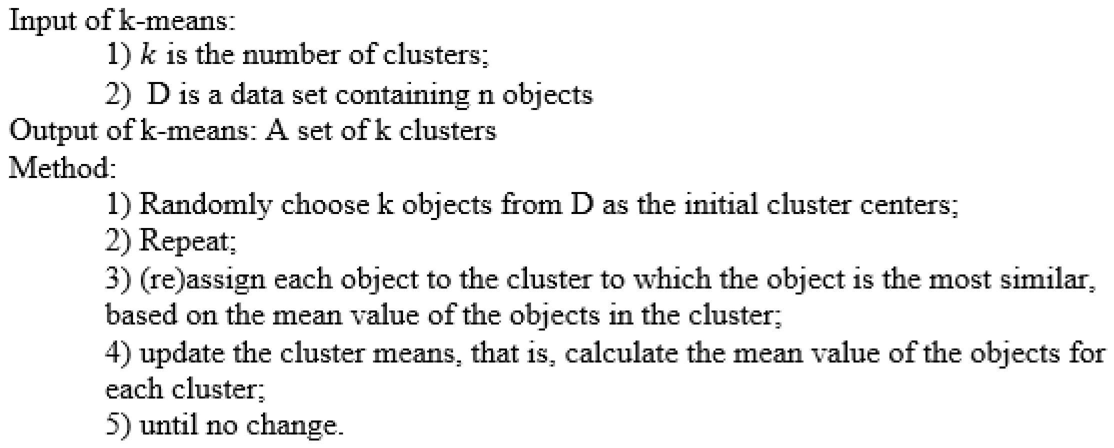
In order to identify pixels most likely to contain significant objects, we apply a simple threshold to the image. We just need to create the significant objects image that have index greater than threshold. Percentage of significant objects is calculated by ratio of number of while pixels in binary image to total number of pixels in NIR image.

3.3 The k-means Algorithm

3.3.1 Introduction

A centroid-based partitioning technique uses the centroid of a cluster, C_i, to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean or medoid of the objects or points assigned to the cluster. The k-means algorithm is for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster [5].

The k-means algorithm is a typical clustering algorithm based on distances, using distance as the similarity index for evaluation, which also considers the closer the objects, the greater the similarity. The clusters in this algorithm are determined by nearby objects, so the ultimate goal of this algorithm is to get compact and independent clusters. Figure 3.9 shows the k-means algorithm [5].



```
Input of k-means:
        1) $k$ is the number of clusters;
        2) D is a data set containing n objects
Output of k-means: A set of k clusters
Method:
        1) Randomly choose k objects from D as the initial cluster centers;
        2) Repeat;
        3) (re)assign each object to the cluster to which the object is the most similar,
        based on the mean value of the objects in the cluster;
        4) update the cluster means, that is, calculate the mean value of the objects for
        each cluster;
        5) until no change.
```

Fig. 3.9. The k-means algorithm.

3.3.2 Find the Right Input

The input of the k-means algorithm, k, is very important, but the question is how the value of k is determined. In the k-means algorithm, k cluster centers are chosen at first, where k is a user-specified parameter, which is the desired number of clusters. Therefore, the number of clusters contained in the dataset is known at first. However in many cases, the distribution of the dataset is unknown. In fact, clustering is one way to find the distribution, which leads us into the contradiction of which came first, the chicken or the egg. Some methods of how to effectively determine the value of k will be introduced.

The first method is to find the stabilization of a dataset. A dataset can be resampled twice to generate two subsets, and then the two subsets can be clustered by using the same algorithm, which generates two sets of k clusters. Next, the similarity of these two sets are calculated. If the degree of similarity is high, it means this k-cluster is stable and can be used for the entire dataset [8].

The second method is starting the k-means algorithm with k=1, and going through the process of splitting and merging. The structure will evolve its stable equilibrium state k_i. It is very useful for a slightly overlapping dataset [8].

The next question is how to choose the appropriate initial cluster centers. One method is to choose the initial centers randomly, but this may cause poor quality of the clusters. However, the initial cluster centers can be chosen randomly multiple times, and then run each set with different random centers. Finally, the cluster with the least sum of squared errors (SSE) is chosen [5].

The second effective method is to take a sample and use hierarchical clustering techniques. k clusters can be extracted from the hierarchical clustering, and used as the initial cluster centers. This method is effective, but only for the following two situations: first, the sample is relatively small; second, the value of k is relatively smaller than the size of the sample [5].

Another method is to choose the first cluster center randomly or take the first points as their cluster centers. For each subsequent initial center, the point that is farthest away from the initial center is chosen. It is certain that the initial cluster center is not only random but also spread out by using this method, but beware of the outliers. In addition, the cost of finding out points farthest from the current set of initial centers is also very large. One way to overcome this problem is to use the samples, which can also reduce the amount of computation [5].

Which distance measure is better becomes the next question. Euclidean distance and cosine similarity are different in mathematical logic. Euclidean distance measure is affected by different scale indicators. Therefore, generally it need to be standardized, and the greater the distance is, the greater the differences between individuals are. Cosine similarity measure in vector space is not affected by the scale indicators. The value of cosine lies in the interval [-1,1], and the larger the value is, the smaller the differences are. However, for specific applications, under what circumstances are better for using Euclidean distance or under what circumstances are better for the use of cosine similarity?

In the geometric meaning, a triangle is formed by using a segment in an n-dimensional vector space as a base and the origin as a apex, but the angle of the apex is unknown. It means that even though the distance between two vectors are certain, the value of cosine of the angle between them is still uncertain. If two vectors have the same trend, the cosine similarity tends to have better performance than Euclidean distance [8]. For example, two vectors are (9,9) and (6,6). Obviously, these two vectors have the same trend, but the solution of Euclidean distance is way worse than cosine similarity.

If there are no points assigned to some cluster, it can form an empty cluster. A strategy is required to choose a substitute cluster center. Otherwise, the sum of squared errors will be large. One way is to choose a point farthest away from the current cluster center, which can eliminate the most impact on sum of squared errors. Another method is to choose a substitute from the cluster center with the largest sum of squared errors, which can split the clusters and reduce the sum of squared errors. If there is more than one empty cluster, the process should be repeated several times. In addition, the empty cluster may cause bugs during the programming.

3.3.3 The k-means Algorithm Process

To compute the distance on numeric data, distance measures are commonly used, which includes Minkowski distance, Euclidean distance and Manhattan distance.

The Minkowski distance between sample point i and sample point j is defined as

$$d(i,j) = \sqrt[h]{|x_{i1} - y_{j1}|^h + |x_{i2} - y_{j2}|^h + \cdots + |x_{ip} - y_{jp}|^h} \tag{3.2}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $i = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and h is the order (the distance so defined is also called $L - h$ norm). Here, h can be positive, negative, or even infinite. Therefore, here comes some special case of Minkowski distance when h is specified.

While $h = 1$, here comes Manhattan distance which also called as city block distance ($L_1 norm$). It is defined as

$$d(i, j) = \sqrt{|x_{i1} - y_{j1}|^2 + |x_{i2} - y_{j2}|^2 + \cdots + |x_{ip} - y_{jp}|^2} \tag{3.3}$$

What is more, the supremum distance is generated from the Minkowsi distance for $h \to \infty$. It is defined as

$$d(i, j) = \lim_{h \to \infty} (\sum_{f=1}^{p} |x_{if} - x_{jf}|)^{\frac{1}{h}} = \max_{f} |x_{if} - x_{jf}| \tag{3.4}$$

There are also some important properties for these distance measures as follows:

$$d(i, j) > 0 if i \neq j, and d(i, i) = 0 (Positive definiteness) \tag{3.5}$$

$$d(i, j) = d(j, i)(Symmetry) \tag{3.6}$$

$$d(i, j) \leq d(i, k) + d(k, j)(Triangel Inequality) \tag{3.7}$$

Last but not the least, cosine similarity can be also used for distance measure. It is defined as

$$\cos(d_1, d_2) = \frac{(d_1 \cdot d_2)}{\|d_1\| \cdot \|d_2\|} \tag{3.8}$$

where d_1, d_2 are two vectors, $\cdot$ indicates vector dot product and $\|d\|$ is the length of vector d.

The purpose of using k-means is to divide the sample into k clusters. In cluster analysis, a training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ is given, where each $x^{(i)} \in \mathbb{R}^n$. First of all, k cluster centers will be chosen randomly as $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$.

Secondly, for every sample i, each cluster that the sample belongs to should be calculated as:

$$c^{(i)} := arg\min_{j} \|x^{(i)} - \mu_j\|^2 \tag{3.9}$$

And for every cluster j, the cluster centers (or mean points) of each cluster will be recalculated as:

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}} \tag{3.10}$$

Here, k is the given number of clusters, and $c^{(i)}$ represents the cluster that has the shortest distance between sample i and k clusters, where the value of $c^{(i)}$ is ranged from 1 to k. The cluster centers (or mean points) μ_j represent the guess of the center of the samples that belongs to the same cluster.

For example, in the universe, all the stars can be divided into k constellations. To do that, first of all, k points (or k stars) in the universe are randomly chosen as cluster centers. Then the first step is to calculate each distance between each stars and k cluster centers, and choose the nearest constellations as $c^{(i)}$. Therefore, after the first step, all the stars will have their own constellations. The next step is, for each constellations, to re-calculate their new cluster centers μ_j, which means find out the average of all the stars in each constellations. Then, repeat step one and step two until there is no change for cluster centers or the change can be ignored. Figure 3.10 shows how k-means works for n sample points, here $k = 3$.

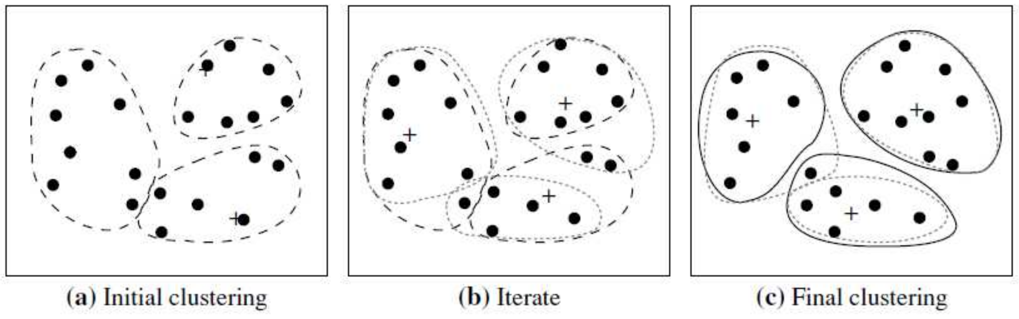


Fig. 3.10. Clustering the sample points by using the k-means algorithm.

The iteration steps of the k-means algorithm will not end until there is no change for cluster centers or the change can be ignored. In mathematical logic, it means that the algorithm will converge. As it was mentioned above, the end condition of the algorithm in the previous section is convergence. Therefore, it can prove that the k-means algorithm can ensure the convergence. Next, for figuring out the convergence, distortion function will be defined as follows:

$$J(c, \mu) = \sum_{i=1}^{m} \|x(i) - \mu_{c^{(i)}}\|^2 \tag{3.11}$$

The function J means the sum of squares of the distances between each sample points and their cluster centers, so the purpose of the k-means algorithm is to minimize the function J. Assume that before the function J reaches its minimum, the cluster center μ_j of each cluster can not be changed and the cluster $c^{(i)}$ that the sample points belong to can adjusted for the reduction of the function J. In the same manner, the cluster $c^{(i)}$ that the sample points belong to can be fixed and the cluster center μ_j of each cluster can be adjusted for the reduction of the function J. These two processes can make sure that the function J will monotonically decrease within the loop. When the function J achieve to its minimum, μ and c also converge. Theoretically, there are many different pairs of μ and c that make the function J reach its minimum, but they are rare practically.

Since the distortion function J is a non-convex function, it means that it is not guaranteed that the obtained minimum value is the global minimum. Therefore, it is really picky and tricky how the initial cluster centers of the k-means algorithm are chosen, but generally the local minimum, achieved by the k-means algorithm, is enough to meet the requirement. If the local minimum does not meet the requirement, the different initial value of the k-means algorithm can be chosen and find out the corresponding μ and c of the minimum J among the different cases.

3.4 Support Vector Machine (SVM)

3.4.1 Introduction

Data classification is one of the most important subjects in data mining. Data classification is based on some existing training data, and then generates a trained classifier according to some theory. Next, the trained classifier is used to test the unknown data to find out a new classification.

One of these theories is Support Vector Machine (SVM). SVM is first proposed by Cortes and Vapnik in 1995. SVM has its advantage of solving the problem of nonlinear and high dimensional pattern recognition, and be able to extend the use to other machine learning problems, such as the function fitting [5]. SVM is a method for the classification of both linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using support vectors and margins. The data mining algorithm SVM is capable of finding the maximum margin for image pattern recognition with high classification accuracy and limited architecture design [6].

Support Vector Machine is also a method that is based on the boundary classification. The basic principle of SVM can be defined as follows. For the better understanding, two-dimensional data is taken as an example. If all the points of a training data is distributed on a two-dimensional plane, they are gathered in different areas according to its own classification. The goal of those classification algorithm based on the boundary classification is to find the boundary between the classifications through training. If the boundary is a straight line, this case is linear; if the boundary is a curve, this case is considered as non-linear. For the multi-dimensional data, such as N-dimensional, they can be regarded as the points in the N-dimensional space, and the boundary in such space is (N-1)-dimensional plane. Thus, linear classifier in multi-dimensional data is a hyper-plane, and non-linear classifier in multi-dimensional data is a hyper-curve.

Linear classifier is shown in the following Figure 3.11. According to the position of the new data point against the boundary, its classification can be determined. Here is the example of only two classifications, but it can be easily extended to multi-classification.

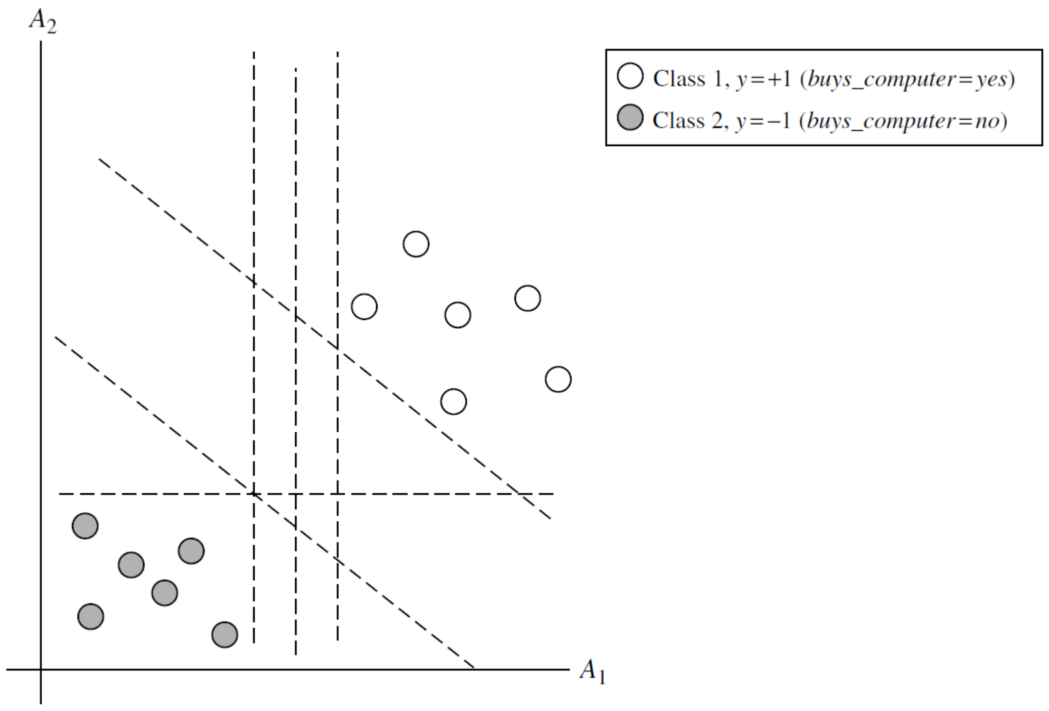

A_2
A_1
Class 1, $y=+1$ (*buys_computer=yes*)
Class 2, $y=-1$ (*buys_computer=no*)

Fig. 3.11. Linearly separable data.

3.4.2 The SVM Process

All the support vector machine are based on a linear classifier, but not all the data can be linearly separated. For example, in a two-dimensional plane, all the points that belongs to two classifications can be separated by a curve. Therefore, all the points in a low-dimensional space can be mapped into high-dimensional space by using SVM, and in the high-dimensional space, all the points are now linearly separated. The boundary of two classification can be determined by a linear classifier now. In a high-dimensional space, it is a linear classifier; in the original dimensional space, it is a non-linear classifier.

The algorithm of mapping from a low-dimensional space to a high-dimensional space is not the key to SVM, and these algorithms are given the kernel functions, which includes Polynomial kernel of degree h, Gaussian radial basis function kernel and Sigmold kernel. SVM is considered as an optimization problem, which means to find out the optimal solution.

Assume that the problem is considered as a function $f(x)$. To find out the optimal solution is to find out the minimum value of the function $f(x)$. Since $f(x)$ is not always continuously differentiable, the solution can not be found by finding the points where derivative equals to zero. The optimization problems can be divided into two categories: unconstrained optimization and constrained optimization. Unconstrained optimization can be expressed as

$$\min_x f(x) \tag{3.12}$$

, and constrained optimization can be expressed as

$$\begin{cases} min_x f(x) & x \in E^n \\ s.t.\ \varphi_i(x) \geq 0 & i \in \{1, 2, \ldots, m\} \end{cases} \tag{3.13}$$

3.4.3 Linearly Separable

From Figure 3.11, the white points are class 1, the black points are class 2, and there are several straight lines that separate these two classifications are represented as $f(x) = k \cdot x + b$, where both k and b are vectors. Therefore, they can also be written in the form of $f(x) = k_1 \cdot x_1 + k_2 \cdot x_2 + \cdots + k_n \cdot x_n + b$. When the dimension of vector x is 2, $f(x)$ is a straight line in a two-dimensional plane; when the dimension of vector x is 3, $f(x)$ is a plane in a two-dimensional space; when the dimension of vector x is greater than 3, $f(x)$ is an (n-1)-dimensional hyper-plane in an n-dimensional space. Here, sign function will used to determine which classification the points belong to. When $f(x)$ is greater than zero, $sign(f(x))$ is +1; when $f(x)$ is less than zero, $sign(f(x))$ is -1. As shown in the Figure 3.11, the y value of the white points is +1, and the y value of the black points is -1.

However, there are several lines that can separate these two classifications, so how to determine the optimal one is the key to SVM. In Figure 3.12, there are two possible separating hyper-planes and their margins.

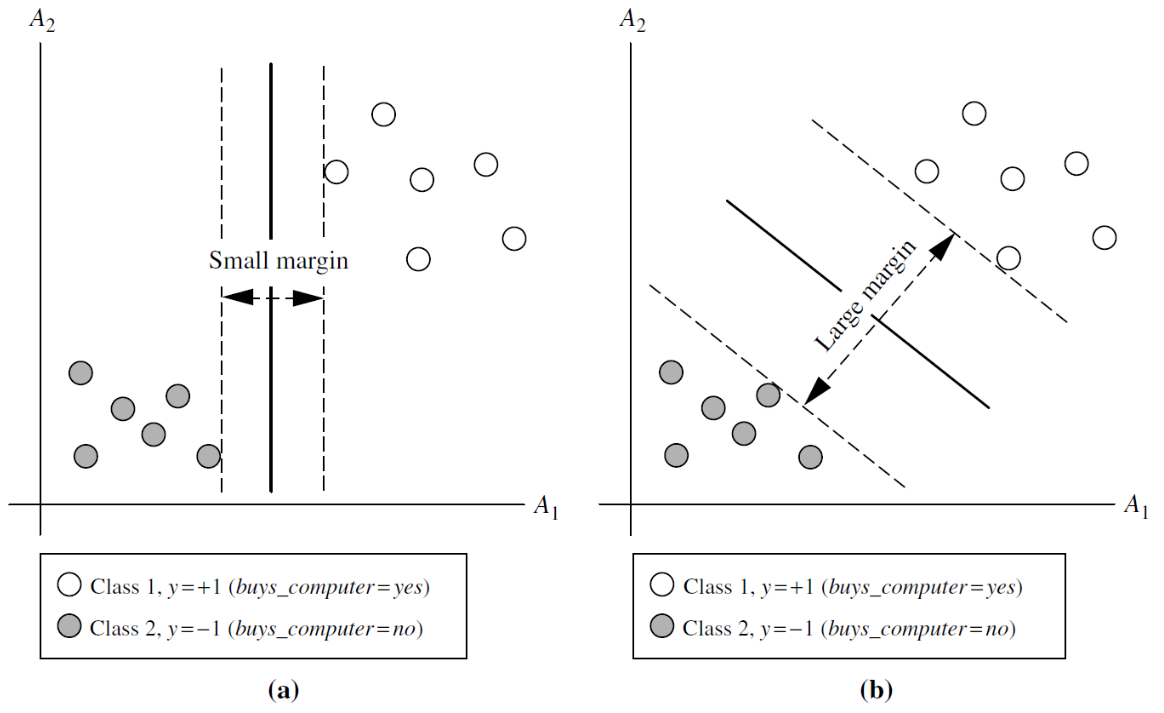

A_2
Small margin
A_1
○ Class 1, $y=+1$ (*buys_computer=yes*)
● Class 2, $y=-1$ (*buys_computer=no*)
(a)
A_2
Large margin
A_1
○ Class 1, $y=+1$ (*buys_computer=yes*)
● Class 2, $y=-1$ (*buys_computer=no*)
(b)

Fig. 3.12. Two possible linearly separable cases.

Intuitively speaking, the larger the margin is, the better the case is. So, maximum marginal hyper-plane is one of the most important conditions in the SVM theory. It makes more sense that the linear classifier with the largest margin is chosen. From a probabilistic point of view, it makes the points with the least confidence have the most confidence. The points on the dash lines are called support vectors, and the straight line in the middle is called classifier boundary $sign(f(x))$. If M is the margin width, M can be defined as

$$M = \frac{2}{\sqrt{k \cdot k}} \tag{3.14}$$

Therefore, the problem is transferred into finding out the optimization:

$$\max \frac{1}{\|k\|} \rightarrow \min \frac{1}{2}\|k\|^2 \tag{3.15}$$

where $\|k\|$ is the Euclidean norm of k, which also equal to $\sqrt{k \cdot k}$. Therefore, the original SVM problem is also the optimization problem:

$$\begin{cases} \min \frac{1}{2}\|k\|^2 \\ s.t.\ y_i(k^T x_i + b) \geq 1,\ i = 1, 2, \ldots, n \end{cases} \tag{3.16}$$

3.4.4 Linearly Inseparable

Since there are too many limitations of linearly separable cases in reality, linearly inseparable cases are mentioned here in Figure 3.13.

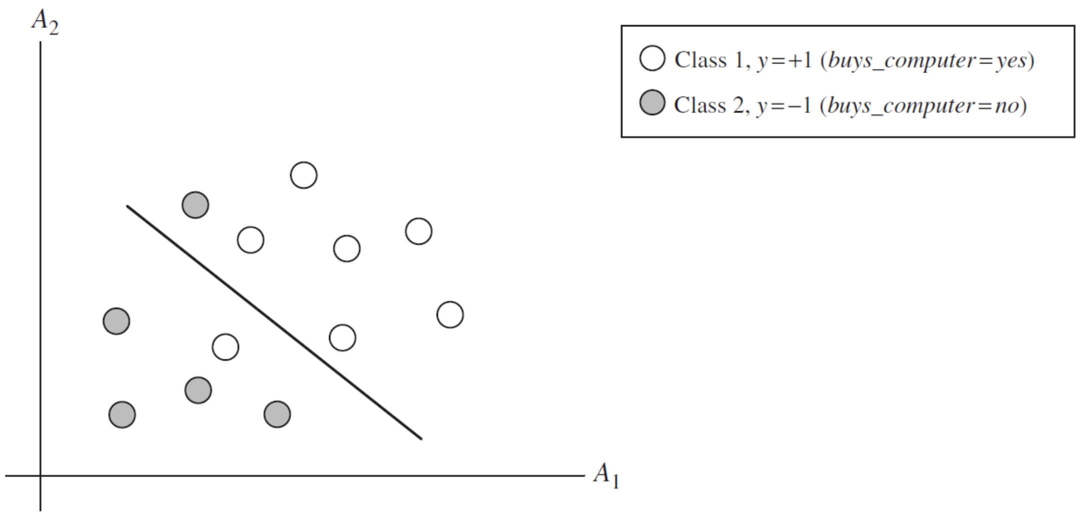

A_2
A_1
Class 1, $y=+1$ (*buys_computer=yes*)
Class 2, $y=-1$ (*buys_computer=no*)

Fig. 3.13. Linearly inseparable data.

In order to do the classification in this case, there are two ways: one is to use the straight line or hyper-plane, but it is not guaranteed that all the points should be separated according to the classification. However, the error is not guaranteed also; the other is to use the curve to completely separate all the points, and it is associated with the kernel function.

Since the error of the first method is not guaranteed, here the second method is detailed. In the Figure 3.13, it is not possible to separate all the points by using a straight line, so the original input data has to be transferred into a higher dimensional space for linearly separation. However, the nonlinear mapping to a higher dimensional space is uncertain, and the computation involved may be costly. There are three admissible kernel functions (shown in Figure 3.14) with which can be replaced anywhere $\phi(X_i) \cdot \phi(X_j)$shows in the algorithm.

Polynomial kernel of degree *h*: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = (\boldsymbol{X}_i \cdot \boldsymbol{X}_j + 1)^h$

Gaussian radial basis function kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|\boldsymbol{X}_i - \boldsymbol{X}_j\|^2/2\sigma^2}$

Sigmoid kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = \tanh(\kappa \boldsymbol{X}_i \cdot \boldsymbol{X}_j - \delta)$

Fig. 3.14. Three kernel functions.

3.4.5 Multiple classification (N-classification)

In the case of multiple classification, there are two basic methods: one is 1 vs (N-1) method; the other is 1 vs 1 method.

In the first method, N classifiers are trained, and whether i-th classifier belongs to classification i or other classifications is checked. To identify an x, a classification with the largest $g^j(x)$ is where i belongs to. $g^j(x)$ is defined as

$$g^j(x) = \sum_{i=1}^{j} a_i^j y_i K(x, x_i) + b^j \tag{3.17}$$

In the second method, $N * (N-1)/2$ classifiers are trained. To identify an x, a vote method is used. The classification with the most votes finally gets x.

4. EXPERIMENTAL RESULTS

The hyperspectral image of the mud sample mentioned in Chapter 1 is used as the input data (raw data) of the hybrid system. The dataset is an 811-band 50-by-1600 hyperspectral image of a mud sample, whose pixel values are band interleaved by line (BIL) in order of increasing band number. BIL is one of three primary methods for encoding multiband image data, which has the advantage of accessing the spatial and spectral data easily.

4.1 Apply Vegetation Detection Algorithm

To find out the three most informative bands is the first step. The MATLAB function multibandread is one of the most important functions to read in the hyperspectral image. Therefore, an RGB image is imported into MATLAB, which red means the NIR band, green means the visible red band and blue means the visible green band. Figure 4.15 and 4.16 show the color infrared image.

Fig. 4.15. Entire color infrared (CIR).

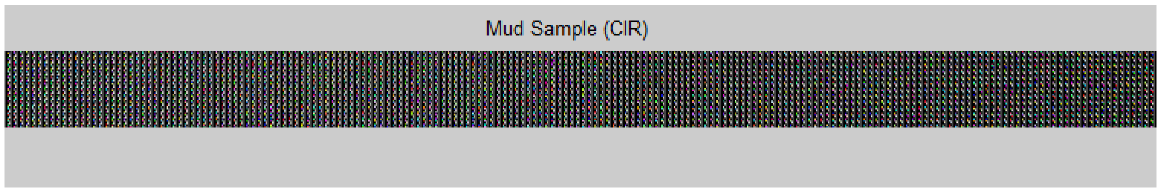


Fig. 4.16. Zoom in of CIR.

Decorrelation stretch is applied in the previous CIR image for better display. Decorrelation stretch can highlight the elements in a multichannel image by exaggerating the color differences. It is pretty wise to use option 'Tol'in decorrelation stretch, which can linearly contrast stretch. Tol = [LOW_FRACT HIGH_FRACT] specifies the fraction of the image to saturate at low and high intensities. Therefore, Tol is specified as 0.01, saturating at low and high intensities by one percent. Figure 4.17 and 4.18 show the color infrared image with decorrelation stretch.

Fig. 4.17. Entire color infrared (CIR) with decorrelation stretch.

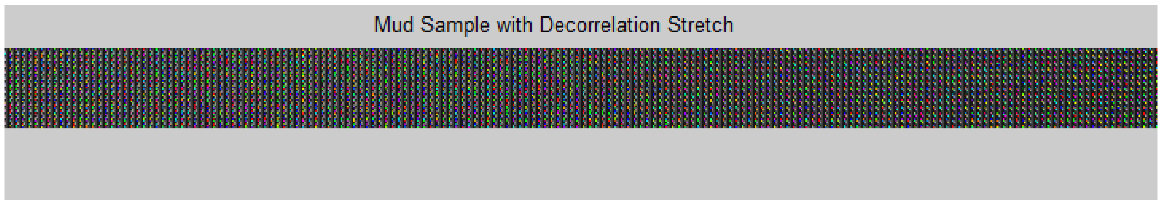


Fig. 4.18. Zoom in of CIR with decorrelation stretch.

An NIR-Red scatter plot is constructed to help us compare the NIR band (displayed as red) and the visible red band (displayed as green). As we said above, we can easily extract the NIR and red bands from the CIR composite. So we can see the difference between these two bands in grayscale images. Figure 4.19, 4.20, 4.21 and 4.22 show the near infrared band image and the visible red band image.

Fig. 4.19. Entire near infrared (NIR) band.

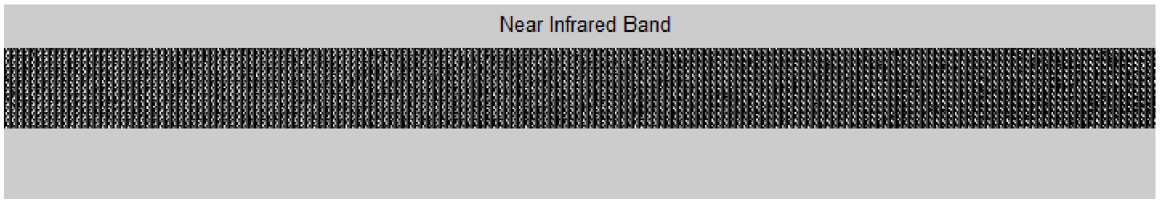


Fig. 4.20. Zoom in of NIR band.

Fig. 4.21. Entire visible red band.

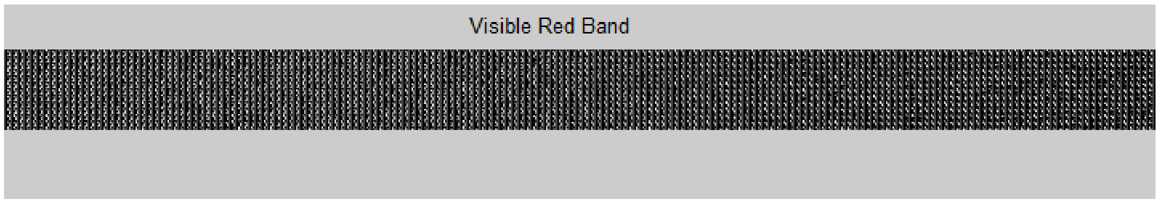


Fig. 4.22. Zoom in of visible red band.

Using the MATLAB function plot, we can easily have the scatter plot of each pixel with red level vs NIR level. For the purpose of the further computation, we scale the red level and NIR level values from 0 to 1. Figure 4.23 shows the scatter plot of visible red band and NIR band.

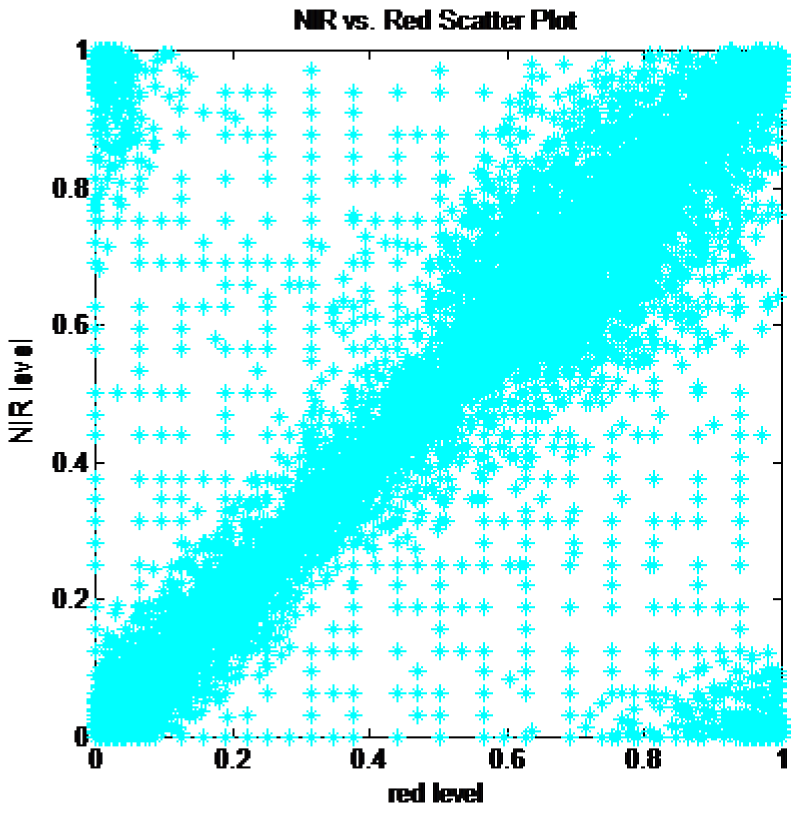


Fig. 4.23. NIR band vs visible red band.

From the scatter plot, most of pixels are along the diagonal, which means the NIR and red values are nearly the same. Also, we have some pixels at corners, which are the significant objects or pixels we are looking for.

To find out the significant objects, we have to compute the index of each pixel:

$$\text{Index} = (\text{NIR - red})\ ./\ (\text{NIR + red}) \tag{4.1}$$

From this equation, if index approaches to 0, pixels are along the diagonal; if index approaches to 1, pixels are at top left corner; if index approaches to -1, pixels are at bottom right corner. Figure 4.24 and 4.25 show the normalized difference image by using the index equation.

Fig. 4.24. Normalized difference image by using the index equation.

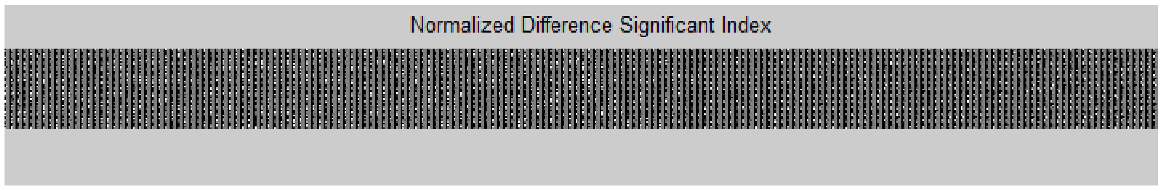


Fig. 4.25. Zoom in of normalized difference image by using the index equation.

In order to identify pixels most likely to contain significant objects, we apply a simple threshold to the previous image. We just need to create the significant objects image that have index greater than threshold. Figure 4.26 and 4.27 show the index image with threshold applied.

Fig. 4.26. Index image with threshold applied.

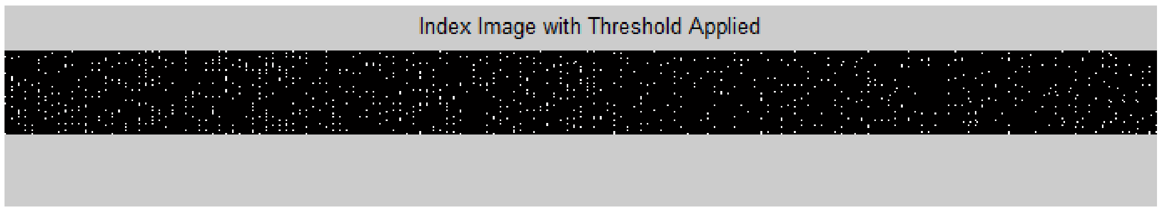


Fig. 4.27. Zoom in of index image with threshold applied.

Finally, we re-draw the useful scatter plot and the index image with threshold applied for better display. Figure 4.28 and 4.29 show the colored scatter plot and index image with threshold applied.

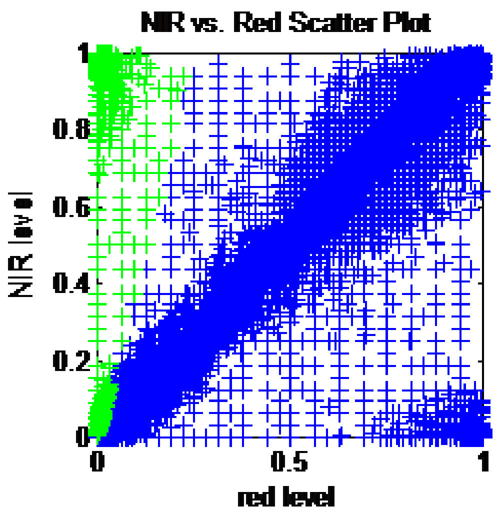


Fig. 4.28. Scatter Plot of NIR vs Red band and Index image with threshold applied with color. The green part is the significant objects; the blue part is the mud.

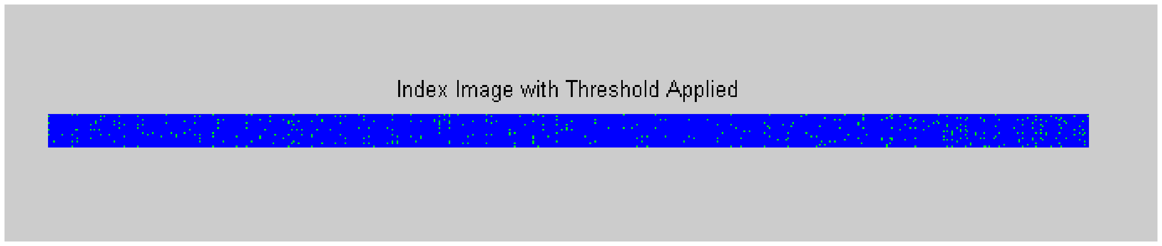


Fig. 4.29. Zoom in of Index image with threshold applied with color.

Percentage of significant objects is calculated by ratio of number of while pixels in binary image to total number of pixels in NIR image. It was found to be 3.5438%.

4.2 Apply the k-means Algorithm

The dataset here is the same as the one before. However, the hyperspectral image is read in single precision, and mathematically, the dataset is stored in two dimensional by having each column display the entire pixel values of each layer or band. Since the second method of how to determine k and the percentage of significant objects is used, here is the result of the k-means algorithm as follows: (start from $k = 2$)

If $k = 2$, it means that the result only has two clusters. The measured time for running the model is 200.49600 seconds. The hyperspectral image is divided into two clusters, one includes 34640 (43%) and the other includes 45360 (57%). If we see the original photo, we can see the tapes along the sides of the picture. It is reasonable that the 43% is the tape and that the 57% is the mud sample. Since the sum of squared errors of $k = 2$ is 731482.1304914309, the mud sample definitely can be divided into more clusters. Figure 4.30 shows the k-means algorithm with $k = 2$.

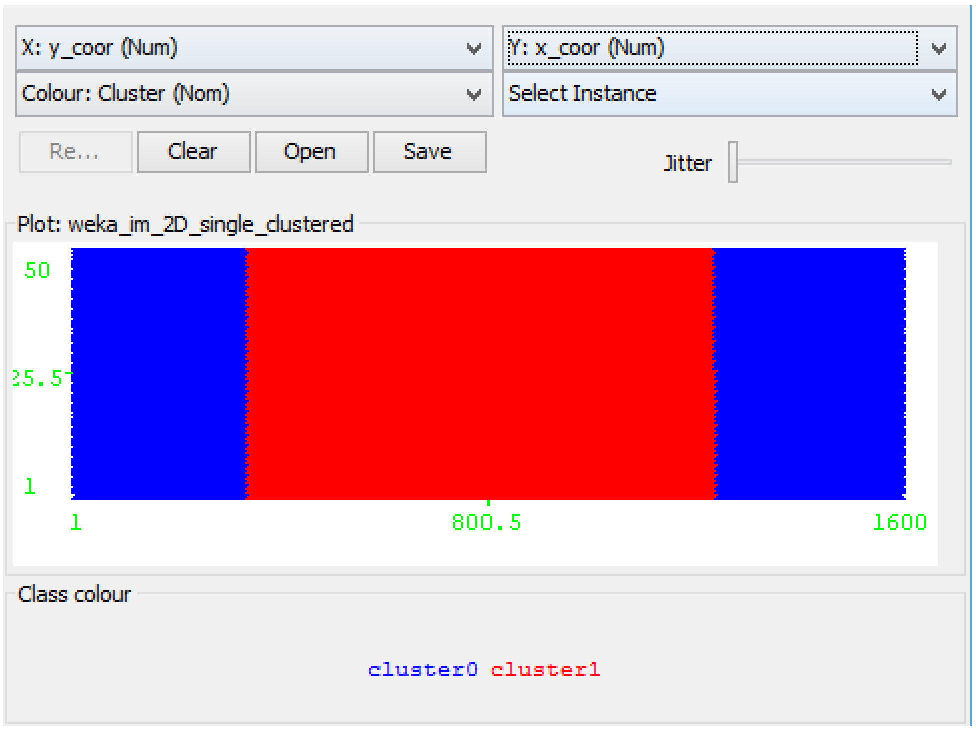

X: y_coor (Num)
Y: x_coor (Num)
Colour: Cluster (Nom)
Select Instance
Plot: weka_im_2D_single_clustered

Fig. 4.30. The k-means algorithm with k=2.

If $k = 3$, it means that the result only has three clusters. The measured time for running the model is 703.78100 seconds, and it is significantly increased from $k = 2$. The hyperspectral image is divided into three clusters, 33441 (42%), 11029 (14%) and

35530 (44%). It is still reasonable that the 42% is the tape and that the rest 58% is the mud sample with one more cluster. Figure 4.31 shows the k-means algorithm with $k = 3$.

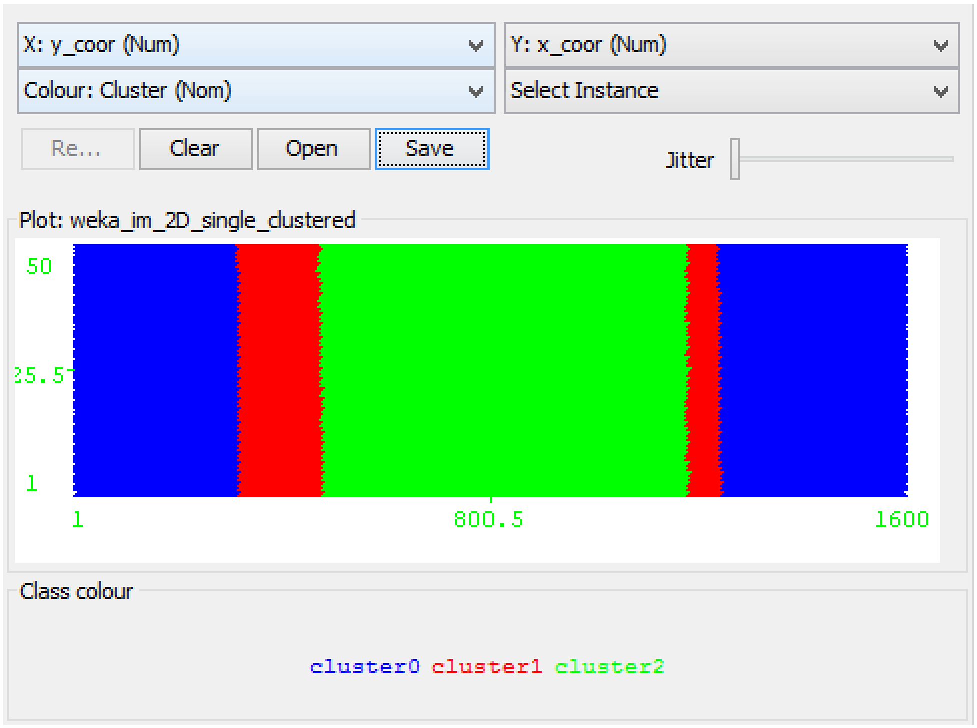


Fig. 4.31. The k-means algorithm with k=3.

If $k = 4$, it means that the result only has four clusters. The measured time for running the model is 858.32800 seconds. The hyperspectral image is divided into four clusters, 33132 (41%), 11105 (14%), 31009 (39%) and 4754 (6%). It is still reasonable that the 41% is the tape and that the rest 59% is the mud sample with two more clusters. It is noticed that the 6% cluster may be have some association with the significant objects and that the sum of squared errors of $k = 4$ is 138644.72818111547. Although the sum of squared errors is significantly reduced compared to $k = 2$, it is not stable yet. Figure 4.32 shows the k-means algorithm with $k = 4$.

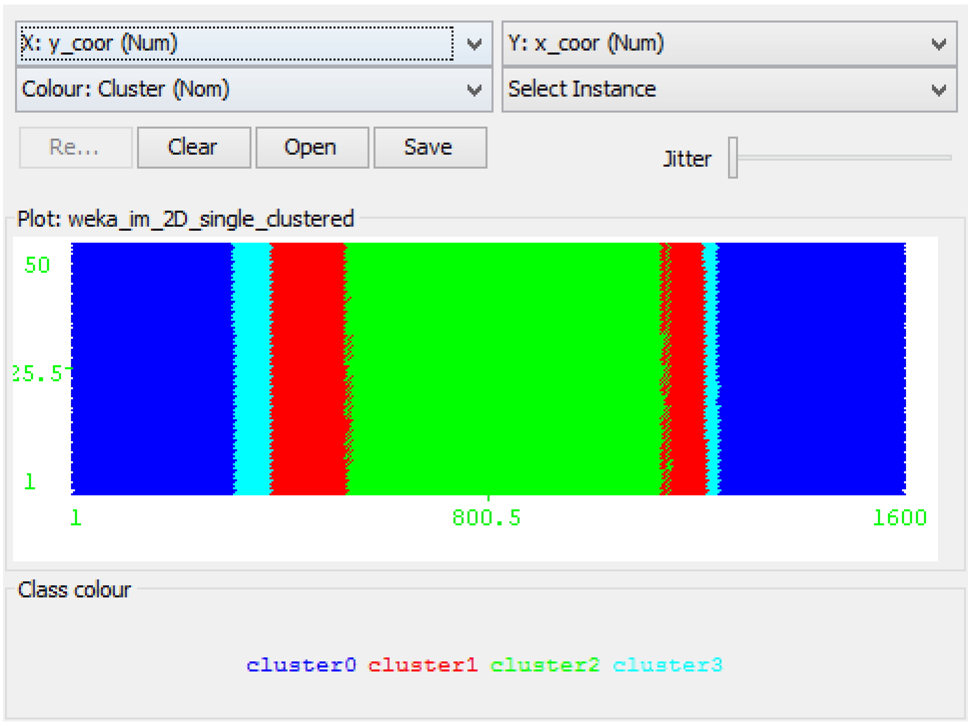


Fig. 4.32. The k-means algorithm with k=4.

The measured time for running the model of $k = 8$ is 1442.11500 seconds. The hyperspectral image is divided into eight clusters, 12882 (16%), 4537 (6%), 6740 (8%), 5238 (7%), 9517 (12%), 23895 (30%), 2237 (3%) and 14954(19%). It is noticed that the 3% cluster is below the percentage of the significant objects we got from the last section and that may be have some association with the significant objects. But the sum of squared errors of $k = 8$ is 61049.33552500921, and it is not stable yet. Figure 4.33 shows the k-means algorithm with $k = 8$.

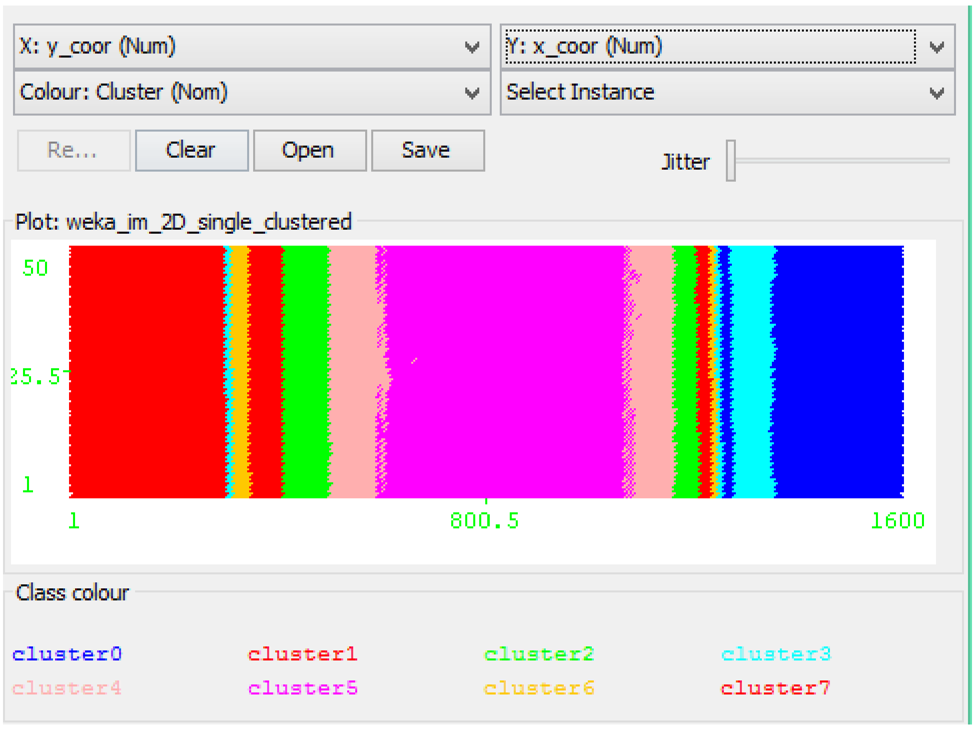


Fig. 4.33. The k-means algorithm with k=8.

The measured time for running the model of $k = 12$ is 1456.19100 seconds. The hyperspectral image is divided into twelve clusters, 6491 (8%), 5978 (7%), 6910 (9%), 5114 (6%), 10693 (13%), 15891 (20%), 1202 (2%), 7449 (9%), 2445 (3%), 3861(5%), 7509 (9%) and 6457 (8%). It is noticed that some clusters are below the percentage of the significant objects we got from the last section and that may be have some association with the significant objects. But the sum of squared errors of $k = 12$ is 40940.46539080349, and still it is not stable yet. Figure 4.34 shows the k-means algorithm with $k = 12$.

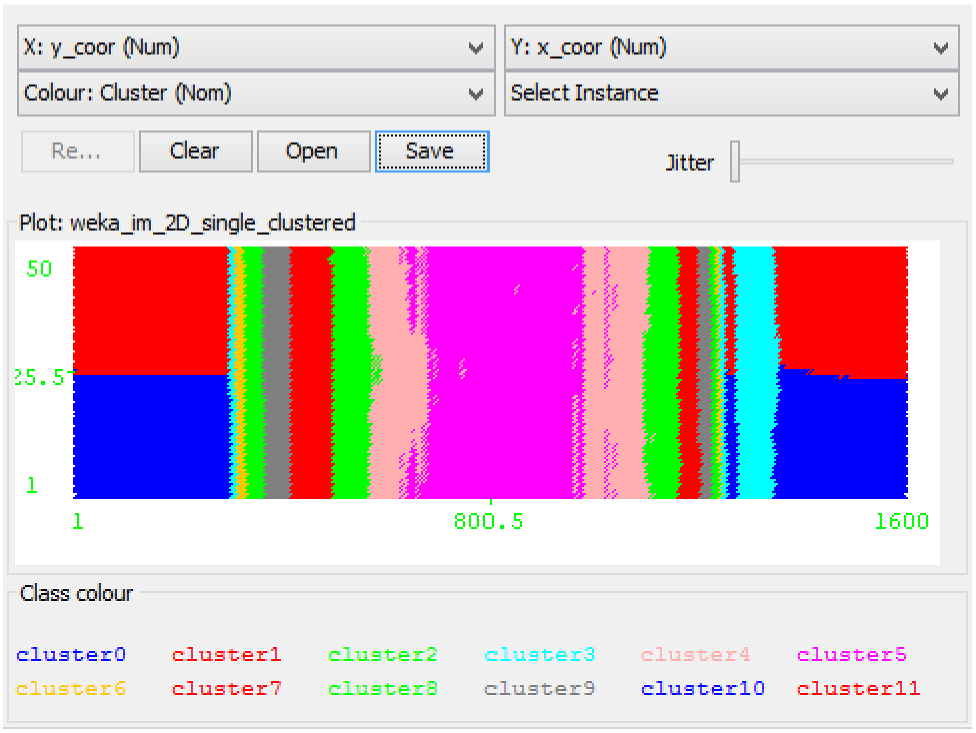


Fig. 4.34. The k-means algorithm with k=12.

The measured time for running the model of $k = 16$ is 1713.76900 seconds. The hyperspectral image is divided into sixteen clusters, 6370 (8%), 3707 (5%), 4405 (6%), 712 (1%), 9822 (12%), 14089 (18%), 1339 (2%), 2784 (3%), 1954 (2%), 2856(4%), 5352 (6%), 5169 (6%), 6457 (8%), 5055 (6%), 6855 (9%) and 3074 (4%). It is noticed that some clusters are below the percentage of the significant objects we got from the last section and that may be have some association with the significant objects. The sum of Squared Error of 16 clusters: 30656.364971050734, and still it is stable. Figure 4.35 shows the k-means algorithm with $k = 16$.

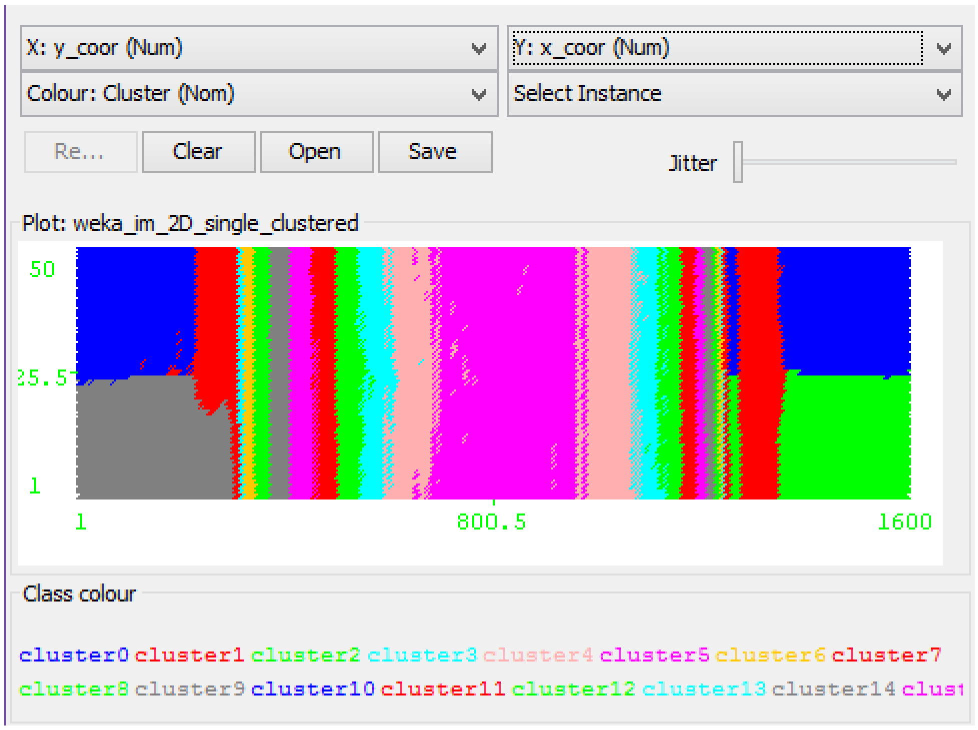

X: y_coor (Num)
Y: x_coor (Num)
Colour: Cluster (Nom)
Select Instance
Re...
Clear
Open
Save
Jitter
Plot: weka_im_2D_single_clustered
50
25.5
1
1
800.5
1600
Class colour
cluster0 cluster1 cluster2 cluster3 cluster4 cluster5 cluster6 cluster7
cluster8 cluster9 cluster10 cluster11 cluster12 cluster13 cluster14 clus

Fig. 4.35. The k-means algorithm with k=16.

Table 4.1 shows results of the k-means algorithm from $k = 2$ to $k = 24$, which includes the sum of squared error, percentage of clusters and running time.

Number of Clusters	Sum of Squared Error	Percentage of Clusters	Running Time (seconds)
2	731482.1304914309	34640(43%) and 45360(56%)	200.49600
3	250413.2621864793	33441(42%), 11029(14%) and 35530(44%)	703.78100
4	138644.72818111547	33132(41%), 11105(14%), 31009(39%) and 4754(6%)	858.32800

8	61049.33552500921	12882(16%), 4537(6%), 6740(8%), 5238(7%), 9517(12%), 23895(30%), 2237(3%) and 14954(19%)	1442.11500
12	40940.46539080349	6491(8%), 5978(7%), 6910(9%), 5114(6%), 10693(13%), 15891(20%), 1202(2%), 7449(9%), 2445(3%), 3861(5%), 7509(9%) and 6457(8%)	1456.19100
16	30656.364971050734	6370(8%), 3707(5%), 4405(6%), 712(1%), 9822(12%), 14089(18%), 1339(2%), 2784(3%), 1954(2%), 2856(4%), 5352(6%), 5169(6%), 6457(8%), 5055(6%), 6855(9%) and 3074(4%)	1713.76900

20	28912.487844715703	6174(7%), 2866(3%), 3309(4%), 756(0%), 7296(9%), 7369(9%), 1143(1%), 2650(3%), 1506(1%), 1754(2%), 4911(6%), 5434(6%), 6247(7%), 4950(6%), 2551(3%), 2214(2%), 279(0%), 4885(6%), 9362(11%) and 4344(5%)	6030.18200
24	26249.48621819265	5229(6%), 3600(4%), 2890(3%), 2503(3%), 4639(5%), 6802(8%), 1224(1%), 2255(2%), 1825(2%), 2822(3%), 3472(4%), 2460(3%), 5154(6%), 4122(5%), 3232(4%), 3006(3%), 2595(3%), 3639(4%), 4640(5%), 3924(4%), 786(0%), 6547(8%), 278(0%) and 2356(2%)	6376.13100

Table 4.1.: The results of the k-means algorithm from k=2 to k=24

Since the sum of squared errors from $k = 2$ to $k = 24$ are available, a curve fitting can be applied for them. Figure 4.36 shows curve fitting of the number of clusters versus the sum of squared error. According to the curve fitting, $k = 16$ has the stable sum of squared error.

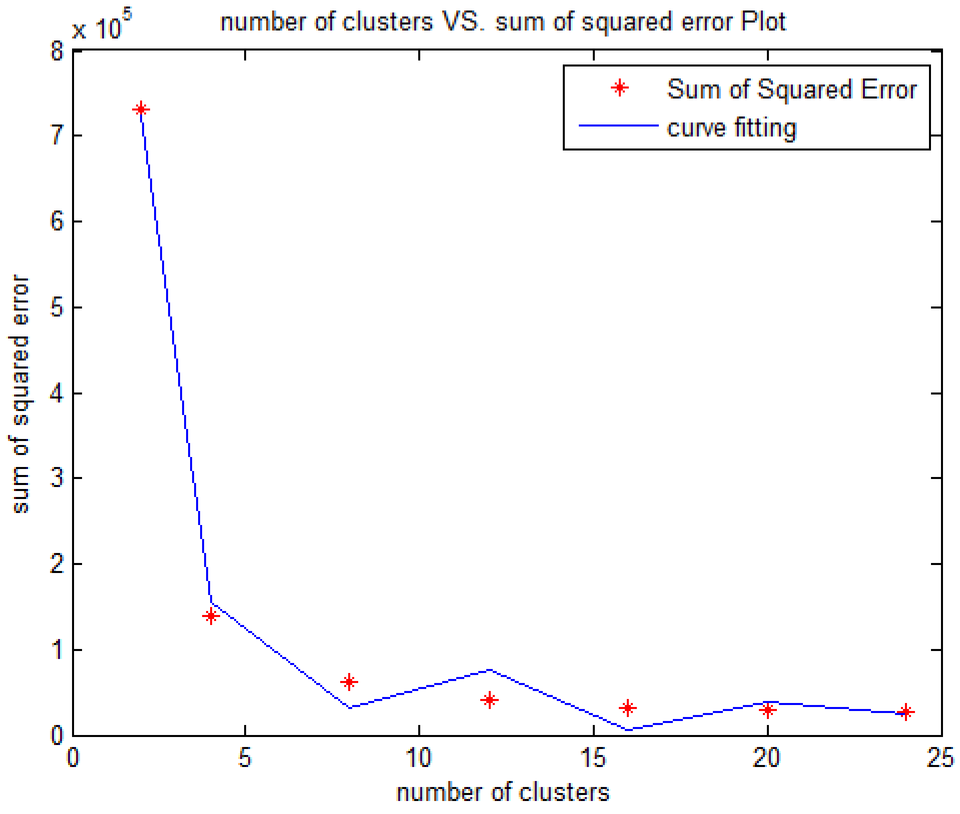


Fig. 4.36. The curve fitting for the sum of squared error.

4.3 Apply Support Vector Machine

According to the result of the previous section, since the hyperspectral image is partitioned into 16 classifications, this problem is considered as multi-class classification. The original SVM from MATLAB can only support two classification, so LIBSVM must be imported. LIBSVM is a library for support vector machines [9], and it supports regression, distribution estimation and even multi-class classification. LIBSVM is available in many programming languages, such as Python, R and MATLAB etc.

The output dataset is generated from the k-means algorithm, and it is now labelled with the class number. Two hyperspectral images are classified into 16 groups or classes, and tested by using 10-fold cross-validation. It means that total 160000 sample points are randomly divided into 10 sub-samples. The 9 out of 10 sub-samples are randomly chosen as the training dataset, and the remaining sub-sample is chosen as our testing dataset. Therefore, that 9 out of 10 sub-samples are now used as the input dataset for LIBSVM. Figure 4.37 and 4.38 are the two hyperspectral images with 16 classifications.

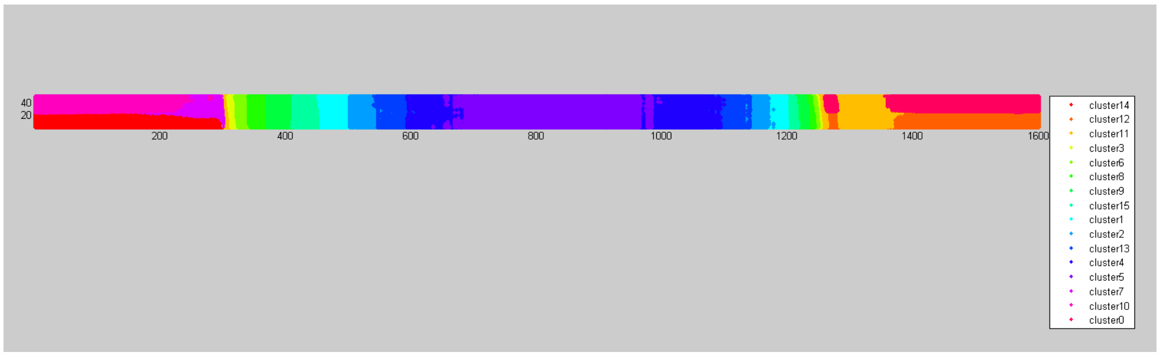


Fig. 4.37. 16 classifications of Sample 1.

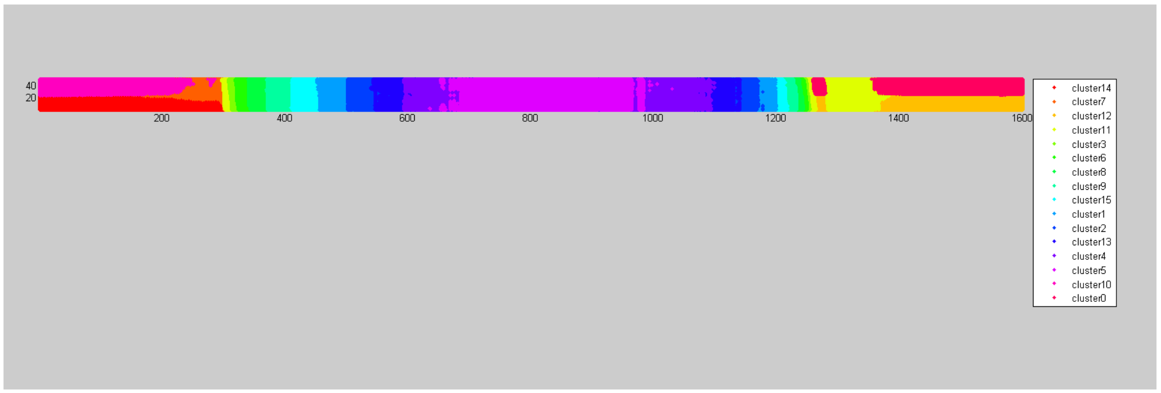


Fig. 4.38. 16 classifications of Sample 2.

It only takes two steps to get the accuracy of the testing data [10]. The first step is to train your prepared dataset which is the random 9 out of 10 sub-samples. The next step is to test the rest sub-sample. Since the test mode is 10-fold cross-validation, there are total 10 sub-samples can be used as the test dataset. 3 out of 10 sub-samples are chosen as testing data sets in our experiment, and trained with different kernel function which includes the linear kernel, the Gaussian radial basis function (RBF) kernel, the polynomial kernel and the sigmoid kernel. The average running time of training the sub-samples with the linear kernel is about 35 hours, and the average running time of testing the sub-sample with the linear kernel is about 30 minutes. The three accuracies with the linear kernel are 96.8875%, 96.3188% and 97.5875%, so the average accuracy is 96.9313%. The three accuracies with the Gaussian RBF kernel are 87.6438%, 87.4563%, and 88.6688%, so the average accuracy is 87.9230%. The three accuracies with the polynomial kernel are 80.0875%, 79.7313%, and 80.8625%, so the average accuracy is 80.2271%. The three accuracies with the sigmoid kernel are 83.0438%, 82.8063%, and 83.6438%, so the average accuracy is 83.1646%. Table 4.2 shows the results of SVM with each kernel function.

	Linear	Gaussian RBF	Polynomial	Sigmoid
Average accuracy	96.9313%	87.9230%	80.2271%	83.1646%
Average training time (seconds)	125832.172	118834.831	111687.569	118735.707
Average testing time (seconds)	1817.163	1446.932	1278.743	1489.109

Table 4.2. Comparing results of each kernel

5. DISCUSSION AND CONCLUSIONS

This Chapter discusses the findings in previous chapters and suggests the best strategies to do the unsupervised learning of hyperspectral images. Also, the limitations of the study and possible directions for future research are discussed.

5.1 Discussion of algorithms

The vegetation detection algorithm is adapted to the hyperspectral image of mud samples to determine the percentage of significant objects. The percentage of significant objects is easy and quick to get by using this algorithm. Since the three most informative bands are chosen for process, the computation is reduced even for large data sets. However, this can also be the disadvantage of this algorithm if there are more than three bands having the same information. If this happens, the three bands should be randomly chosen from those bands having the same information.

The k-means algorithm is used for partitioning the hyperspectral image with the pre-defined k. If a dataset is dense and clear, the result of the k-mean algorithm is better. According to the running time in the previous chapter, this algorithm is relatively efficient and scalable for large data sets. The sum of squared error is a very good standard to determine the value of k. Since k is user-defined and must be determined at first, finding out the value of k may take more time than just processing the dataset by the k-means algorithm. Iterations is processed according to the initial clustering, so the computation is costly comparing to the vegetation detection algorithm.

Support vector machine (SVM) is used to train the data and predict another data for better accuracy by using 10-fold cross-validation. SVM can use the three kernel function instead of non-linear mapping, which can save amount of running time. The problem can be treated as finding the optimal solution in SVM, so Non-linear data can be solved by finding the optimal hyper-plane in a high dimensional space. However, since SVM belongs to supervised learning, the dataset must be labelled and the computation is more costly than the k-means algorithm for large data sets.

5.2 Conclusions

Since we determine the percentage of the significant objects by using algorithm for vegetation detection, we have set the bar for the upcoming data mining technique, the k-means algorithm. When we find the groups whose percentage is less than 3.5%, we definitely need to pay an attention to the groups and find out whether they are what we want.

Vegetation detection helps us determine the percentage of the significant objects. The percentage is a reference of determining the k value. During the process of $k = 2$ to $k = 24$ in the k-means algorithm, if the percentage of a cluster is found to be below 3.5

10-fold cross validation is used for test mode with 16000 pixels in each fold. Since each pixel has many bands available, each pixel of the mud sample can be considered as a sample and a band as a feature. Therefore, we have 144000 training sample, 811 features each and 16000 test samples. The four kernel functions of support vector machine we used are the linear kernel, the Gaussian radial basis function, the polynomial kernel and sigmoid kernel. The polynomial kernel runs faster than the other three kernels, but the accuracy of the polynomial kernel is the worst. The linear kernel has the best accuracy of all the kernels, but the running time is the worst. The parameters of all the kernels here are default. Therefore, the accuracies of some kernels can be improve by tuning the parameters. That is one of the main reasons why the linear kernel has the best accuracy. However, the main purpose here is not to introduce how the parameters are tuned, but the hybrid system.

5.3 Limitations

Since we only have two hyperspectral images, the number of samples may not be enough. Therefore, 10-fold cross-validation is used as the test mode. More samples of hyperspectral images will improve the accuracy. The running time is more concerned as this system runs. A 64-bit operating system is a must, and there is no way you can run this kind of large datasets in a 32-bit OS because the minimum heap size of processing the dataset for memory is 6 GB.

5.4 Future Work

It will be desirable to include in future research some additional models, such as Hierarchical Methods, Bayesian Belief Networks and Probabilistic Model-Based Clustering. Adding more pre-processing or post-processing step is better to improve the accuracy, but more costly. Also, the hyperspectral images can be expanded to include additional factors that may be able to identify the objects with single pixel. The mud sample in this study is just a tool to demonstrate the hybrid system can have a better accuracy. The mud sample can be replaced by any other kinds of hyperspectral images, and even this hybrid system can be extended into other fields required for clustering or classifications. The scatter plot of NIR band and red band from vegetation detection algorithm can be developed with more index value. With more index value available, we can determine more percentage of clusters in order to determine the k value fast.

BIBLIOGRAPHY AND REFERENCES

[1] Landarebe, D. Hyperspectral image data analysis. *IEEE Signal Processing Magazine*(2002), 17-28.

[2] Li, Z., Li, L., Zhang, R., and Ma, J. An improved classification method for hyperspectral data based on spectral and morphological information. *International Journal of Remote Sensing*(2011), vol.32, 2919-2929.

[3] Gamps-Valls, G., Tuia, D., Bruzzone, L., and Atli Benediktsson, J. Advances in Hyperspectral Image Classification: Earth monitoring with statistical learning methods. *Signal Processing Magazein, IEEE Journal* (2014), 1007-1011.

[4] Starr, C. *Biology: Concepts and Application*, Thomson Brooks/Cole, 2005.

[5] Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques.*, Morgan Kaufmann Publishers, 2011.

[6] Mercier, G., and Lennon, M. Support vector machines for hyperspectral image classification with spectral-based kernels. *Geoscience and Remote Sensing Symposium*(2003), vol.1, 288-290.

[7] Chouhan, R., and Rao, N. Vegetation Detection in Multi Spectral Remote Sensing Images: Protective Role-analysis of Vegetation in 2004 Indian Ocean Tsunami. PDPM Indian Institute of Information Technology, 2011.

[8] Witten, I., Frank, E., and Hall, M. *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, 2011.

[9] Chang, C., and Lin, C. LIVSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*(2011), vol.2, 1-27.

[10] Hsu, C., Chang, C., and Lin, C. A paractical guide to support vector classification. *PDF Online*, May, 2009.

[11] Fehr, J., Arreola, K., and Burkhardt, H. Fast Support Vector Machine Classification of Very Large Datasets. *Studies in Classification, Data Analysis, and Knowledge Organization*(2008), 11-18.

[12] McCue, R. A Comparison of the Accuracy of Support Vector Machine and Naive Bayes Algorithms in Spam Classification, University of California at Santa Cruz, Nov,2009.

[13] Zhang, B., Li, S., Wu, C., Gao, L., Zhang, W., and Peng, M. A neighbourhood-constrained k-means approach to classify very high spatial resolution hyperspectral imagery. *Remote Sensing Letters*(2013), Vol.4, Issue.2, 161-170.

[14] Koonsanit, K., Jaruskulchai, C., and Eiumnoh, A. Parameter-Free K-Means Clustering Algorithm for Satellite Imagery Application. *Information Science and Applications (ICISA)*(2012), 1-6.

[15] Braun, A.C., Weidner, U., and Hinz, S. Support vector machines, import vector machines and relevance vector machines for hyperspectral classification C A comparison. *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*(2011), 1-4.

[16] Gao, H., Mandal, M.K., Guo, G., and Wan, J. Semisupervised Hyperspectral Image Classification With SVM and PSO. *Measuring Technology and Mechatronics Automation (ICMTMA)*(2010), vol.3, 312-324.