Use of Self Organized Maps for Feature Extraction of Hyperspectral Data

Thomas C. Null

USE OF SELF ORGANIZED MAPS FOR FEATURE EXTRACTION OF HYPERSPECTRAL DATA

By

Thomas Calvin Null III

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2001

USE OF SELF ORGANIZED MAPS FOR FEATURE EXTRACTION OF HYPERSPECTRAL DATA

By

Thomas Calvin Null III

Approved:

Roger L. King Professor of Electrical and Computer Engineering (Director of Thesis)	A. Wayne Bennett Dean of the College of Engineering
Randy Follett Assistant Professor of Electrical and Computer Engineering (Committee Member)	Nick Younan Graduate Coordinator of the Department of Electrical and Computer Engineering (Committee Member)

Name: Thomas Calvin Null III

Date of Degree: December 14, 2001

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: Dr. Roger King

Title of Study: USE OF SELF ORGANIZED MAPS FOR FEATURE
EXTRACTION OF HYPERSPECTRAL DATA

Pages in Study: 123

Candidate for the Degree of Master of Science

In this paper, the problem of analyzing hyperspectral data is presented. The complexity of multi-dimensional data leads to the need for computer assisted data compression and labeling of important features. A brief overview of Self-Organizing Maps and their variants is given and then two possible methods of data analysis are examined. These methods are incorporated into a program derived from som_toolbox2. In this program, ASD data (data collected by an Analytical Spectral Device sensor) is read into a variable, relevant bands for discrimination between classes are extracted, and several different methods of analyzing the results are employed. A GUI was developed for easy implementation of these three stages.

ACKNOWLEDGEMENTS

I would like to thank Dr. Roger King for all of his input, assistance, and encouragement in the research, preparation, and writing of this thesis, as well as for his overall guidance and instruction throughout my master's program studies at MSU.

I would like to thank Libba LaMastus for sharing her data with me for use in this thesis. Without her original research efforts and data, the amount of time required to complete this work would have been overwhelming.

I would like to thank Dr. David Shaw for his guidance throughout my master's program, as well as the suggestion on creating the GUI in the first place.

Finally, I would like to thank my fiancée, Sarah Nagle, for her enduring and overall support and assistance. In particular, her editorial and typing skills came in handy during the final days of this thesis—she is the reason this work maintains a certain level of technical knowledge while also being aesthetically appealing.

--TCN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Hyperspectral data of crops is being analyzed to extract some meaning of the full knowledge of the plants. Classification of images in a hyperspectral cube is one of the most difficult tasks when analyzing hyperspectral data. The high dimensionality of the data makes it very difficult for a human to visualize, and differences in similar plants' hyperspectral signatures are not obvious.

Typical hyperspectral signatures contain hundreds of frequency bands. Some of the bands contain information about the plant, some of the bands contain information about the atmosphere, and some bands are dominated by electronic noise. Researchers are most interested in the bands that contain information about the plant, but not just information that is unique to the plant kingdom. Researchers want to know what frequency bands are important for species identification. This means that irrelevant bands should be removed from consideration, and some method of determining which bands contain species-specific data must be developed.

Since pattern recognition is the primary emphasis, a neural network seems to be the most logical method for solving this problem. King *et al* presented a first look at the feasibility of analyzing hyperspectral data with a Self Organized Map (SOM) in

Classification of Weed Species Using Self-Organizing Maps [1]. This thesis is an expansion of King's initial work. In the following chapters, the process—in which a program that utilizes neural networks for feature extractions was developed—is described.

CHAPTER II

INTRODUCTION TO NEURAL NETWORKS

2.1 Introduction

Since pattern recognition is the primary emphasis in this thesis' problem, a neural network seems like the most logical method, or solution. In order to better visualize the steps taken toward making this decision, a general understanding of the basics of neural networks is necessary.

Kohonen suggests that three categories of neural networks exist: feedforward, feedback, and a third group known as "competitive, unsupervised, or self-organizing" [2]. Feedforward networks are formed by a set of inputs propagating through a system and resulting in an output [3]. The system is modified by externally adjusting the weights. Feedback networks start in an initial state and adjust their inputs, continuing until they approach a final output. Kohonen describes the third category as "neighboring cells in a neural network compete in their activities by means of mutual lateral interactions, and develop adaptively into specific detectors of different signal patterns." A self-organized map may be the best tool for extracting subtle features in similar hyperspectral images. The enigma of self-organizing maps is their ability to find regularities and correlations in the input layer and group them into vectors without any external adjustment or knowledge of expected outcomes. One interpretation of Kohonen’s definition of the SOM is that it is “an artificial neural network which defines a nonlinear transform from the input space to the set of nodes in the output space. Each node is associated with a model of the input space” [4]. Of self-organizing maps, basically two different models exist.

Willshaw and Von Der Malsburg developed a model that focuses on mapping inputs to outputs of the same dimension [5]. This model is not very useful in analysis of hyperspectral data, due to the high dimensionality of the data. The map must contain some method of compressing the data into a manageable form. Kohonen’s self-organizing map is one of the most referenced mapping techniques because of its ability to take high dimensional data and flatten it into two- or three-dimensional data. One extremely important attribute of Kohonen’s SOM is that it performs data compression without loss to relative distance between data points. A SOM typically uses the Euclidian distance formula to determine the relative closeness or similarity of data. Through an iterative process of selecting a neuron and determining whether other neurons in a “neighborhood” should move closer or further away, it is able to sort data into similar groups [6]. The learning rate controls how far a neuron is moved. At the beginning of the process, the neighborhood size is at least half the map [7], and the learning rate is close to one. Over the course of the process, the neighborhood size and learning rate are independently decreased until the map no longer makes significant adjustments [8][9].

In the case of hyperspectral data, there are three dimensions, consisting of sample number, band number, and amplitude. Though the actual dimensionality of the data is only three, the length of two of the three dimensions is very long. The number of samples can be extremely large and the number of bands is already several hundred. When these two numbers are multiplied together to yield the number of points, the data size becomes extremely large and potentially cumbersome. The SOM is capable of flattening the data into two or three dimensions of which the lengths can be set, therefore resulting in a manageable data size. In addition, the total number of points in the SOM can more closely match the number of types of samples multiplied by the number of bands. For instance, 400 hyperspectral signatures, containing 300 bands, of 10 types of plants would contain 120000 points in a data cube, but could contain as few as 3000 points in a SOM. The map also displays the data from a different viewpoint; instead of viewing the data as a cube, it can be viewed as a two-dimensional plot. This is where human analysis is aided. Instead of looking at the spectral signature of a sample and trying to determine what the sample type is, one can look at the location of the sample on the map and try to determine what the sample type is.

2.2 Applications of the SOM

Kohonen's SOM has been successfully applied to many areas in science, ranging from understanding the human brain "to subsystems for engineering applications"[9]. Many variants of the Kohonen SOM have appeared over the last couple of decades. Many problems with the architecture of the SOM have been addressed. H. S. Hosseini and R. Safabakhsh created a time-adaptive self-organizing map (TASOM) to handle non-stationary input distributions and changing environments. The TASOM assumes that each neuron has its own learning rate and neighborhood function [7].

Another problem with the basic SOM is that the size of the map has to be predefined. D. Alahakoon et al proposed a growing self-organizing map (GSOM) to address this problem. The GSOM starts out with a minimal number of neurons and increases the number as needed [10]. This can save processor time as well as human analysis time. Processor time is a huge concern when using neural networks. Depending on processor speed and the size of the application, a neural network can take days to train.

Another way of decreasing training time has been proposed by M. C. Su and H. T. Chang. Su and Chang use a three-step process in which a K-means algorithm is used to select N^2, and says, “then a heuristic assignment strategy is employed to organize the N^2 selected data points into an N x N neural array”[11]. Finally, the Kohonen SOM is employed to fine-tune the network.

2.3 Analysis of the SOM

The largest problem with the SOM comes from its inherent nature of use. The SOM is used to organize high dimensional data into a two- or three-dimensional map because humans are not very effective at handling hyper dimensional data. Since humans are not effective at handling the hyper dimensional data, just what the map should look like is unknown. Two basic facts need to be determined after the SOM has been created:

1. Is the SOM actually organized?

2. What features did the SOM use to separate the data?

A. P. Azcarraga presents a very legible overview of average unit disorder (AUD). Order is defined as “the degree by which physically (spatially) close map units are assigned values that are similar in the input environment”[12]. Azcarraga labels l_{ij} by

$$l_{ij} = \frac{1}{v} \sum_{k=1}^{v} \left| w_{ik} - w_{jk} \right| \qquad (1)$$

where w_{ij} is a weight of node u_i, v is the number of input units, and l_{ij} is the average absolute difference between the weights of nodes u_i and u_j [12]. Once l_{ij} is determined, AUD can be found by evaluating

$$AUD = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{j \neq i} l_{ij} / d_{ij}}{\sum_{j \neq i} 1 / d_{ij}} \qquad (2)$$

where “d_{ij} is the Euclidean distance between u_i and u_j” and “ N is the number of units in the map” [12].

According to A. P. Azcarraga, AUD can be used to evaluate the degree to which the map is organized regardless of the variant of Kohonen’s SOM. This is an especially important attribute of this method in that it will provide a solution to problem number one regardless of the way in which that state was achieved. The usefulness of the AUD in hyperspectral data should be apparent. It will give us the stopping point needed to be sure the map is organized. The most common approach to solve problem number one is to arbitrarily pick some fixed number of intervals. This technique is far superior in that it is more processor efficient and more reliable.

A. P. Azcarraga's study of the behavior of the AUD curve has led to a general graph that represents a correctly organized map. The graph displays three distinct behaviors. In the first part, AUD increases because "the map units are being sensitized to the different input patterns"[12]. In the second phase, AUD decreases; Azcarraga refers to this as "Global Ordering." In this phase, the different groups are being developed. This phase will continue until it becomes organized on a global scale. That is, large clusters of data are spatially organized, but the organization within the clusters is not complete. The final phase consist of moving units around within the clusters. This will affect the overall AUD very little. By measuring the change in AUD, the map will "know" when to stop [12].

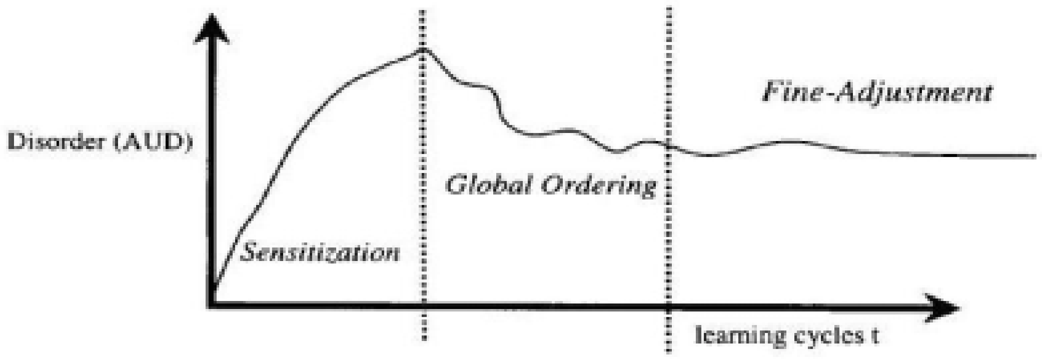


Figure 2-1. General curve for the AUD metrice [12]

A second characteristic of this process is that the actual shape of the AUD curve, as shown in Figure 2-1, must be closely matched or the data will not be clustered in ways that will make sense to humans [12]. Since parameters such as learning rate and neighborhood size are arbitrarily chosen in some variants of Kohonen's SOM, this would provide feedback as to whether or not the initial guesses were close enough.

This brings us to the second question: "What features did the SOM use to separate the data?". D. Merkl and A. Rauber devised another variant of Kohonen's SOM that provides a "label" for the different units on the map. Merkl and Rauber's *label*SOM derives its labels from the important features it used to determine where the unit should be. This technique was developed for archiving journal articles of similar topics. To apply this technique to hyperspectral data analysis would take some manipulation, but should be feasible.

D. Merkl and A. Rauber use Kohonen's SOM, also known as the basic SOM, to organize the data but, then, runs an additional algorithm to provide "labels" for the data clusters. In this algorithm, Merkl and Rauber analyze the co-occurrence patterns. That is, they use "the deviation between weight vector components and the respective components of the input vectors"[8]. Merkl and Rauber define δ_{ik} as

$$\delta_{ik} = \sqrt{\left(\sum_{xj\,\varepsilon Di} (m_{ik} - x_{jk})^2\right)} \qquad (3)$$

where *Di* denotes the set of documents mapped onto unit *i*. δ_{ik} is the deviation of a particular vector component k. Index terms that have a certain range of deviation are used as the labels for that unit. [Merkl] presents a successful application of this process in which the groupings of labels give a sufficient description of the set.

2.4 Conclusion

Hyperspectral images do not contain words as in D. Merkl and A. Rauber’s example, but hyperspectral images do contain spectral bands. Applying multiple models to hyperspectral data should allow critical knowledge of the data to be obtained. Specifically, by applying A. P. Azcarraga’s parameters of the AUD and D. Merkl and A. Rauber’s *label*SOM, knowledge of the important spectral bands in hyperspectral data can be extracted with high confidence. This is an important and necessary step towards reducing the amount of data required to sufficiently describe an image.

CHAPTER III

WHAT IS REMOTE SENSING?

Remote sensing is a method of acquiring information about an object from a distance without physically being in contact with the object. This is accomplished by sensing electromagnetic energy that is reflected or emitted by the object and analyzing its spectral signature. Imaging systems such as ASD have seven elements involved in remote sensing. The seven elements are: the energy source, radiation in the atmosphere, interaction with the target, recording of the energy by the sensor, processing the recording into an image, interpretation of the image, and, finally, application of the analysis to solve a problem.

A large part of remote sensing requires an interaction between incident radiation and the targets of interest. This means there has to be an energy source to provide the electromagnetic energy to the target. Electromagnetic energy varies along a spectrum of wavelengths. For an ASD, this is the sun. The sun is the equivalent of a black body radiator at 6000 degrees Kelvin. In other words, it radiates in wavelengths increasingly from .2 to .5 micrometers and then decreasingly from .5 micrometers. ASD operates in a spectral range from .35 to 2.15 micrometers, utilizing the highest energy range radiated from the sun. However, not all of this energy reaches the surface.

Some of the sun’s radiated energy is lost in the atmosphere. Losses of the sun’s incident energy are caused by scattering, reflection, and absorption. The losses in the atmosphere are not evenly distributed across the magnetic spectrum. Ozone filters the ultraviolet part of the spectrum out, while there are also wavelengths missing completely in the visible and infrared spectrum due to water vapor absorption. The electromagnetic energy absorbed by the water vapor is converted into heat. Other wavelengths can be diffused or reflected to varying degrees due to aerosols, depending on the thickness of the haze. Rayleigh scattering in the atmosphere is a function wavelength and haze. Scattering varies as λ^-x with x being less than or equal to 4 depending on the thickness of the haze. This has to be taken into account when processing the data. Radiation that passes through the atmosphere can then interact with the target. Energy that strikes the target, or is incident upon the surface, can be absorbed, transmitted, and reflected. Incident energy must interact in one or more of these ways. That is to say that the energy incident to the target must equal the sum of the absorbed, transmitted, and reflected energy. Energy is reflected in two forms: specular and diffuse. Specular reflection occurs on mirror-like surfaces—this is typical of roads. Diffuse reflection occurs on rough surfaces and the energy is reflected uniformly in all direction—this is typical of vegetation as seen in Figure 3-1.

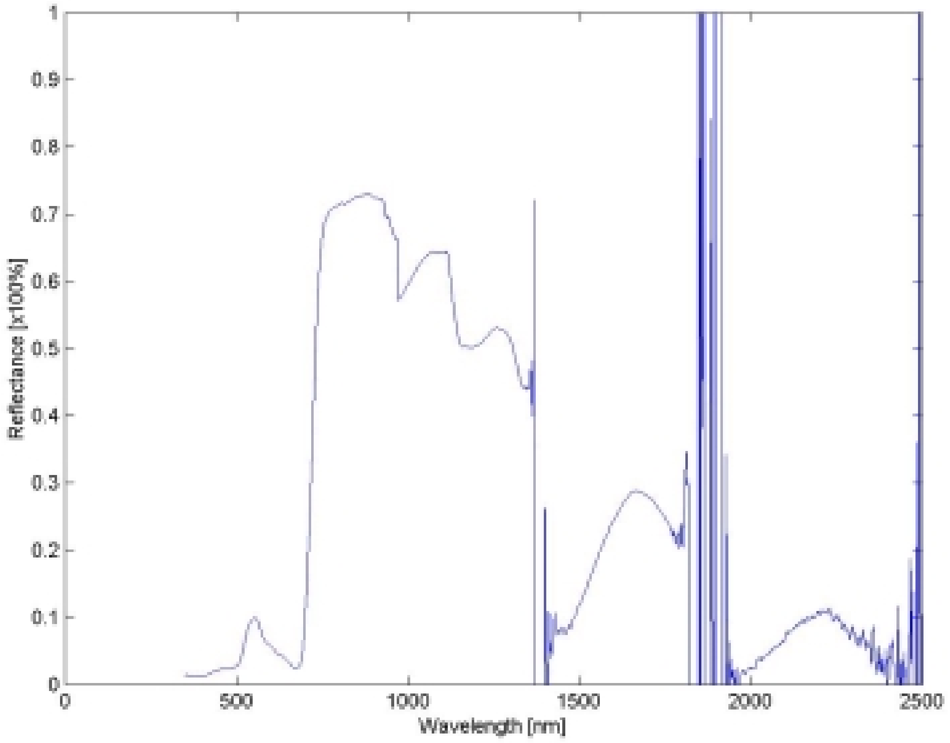


Figure 3-1. Typical vegetation reflectance spectrum

Leaves in the spring and summer reflect light in the green wavelengths and absorb light in red and blue part of the visible spectrum. Longer wavelengths such as 2000nm to 2400nm contain information on moisture content. It is this diffuse reflection characteristic of vegetation that is of interest to us. Data can be extracted from the diffuse energy telling us what kind of plant it is and how healthy it is. A detailed description of the spectral signature of a target is required to extract this kind of information. Hyperspectral sensors are the tools for completing this task.

Data collection is one of the most tedious and time-consuming parts of remote sensing research. Libba Lamastus provided several sets of data, which aided in the timely development of the final program.

To make good use of the data, we must be able to extract meaningful information from the image cube. In order to interpret and analyze the image cube, identification and measurement must be made of various targets. A target must be an object that is distinguishable from other features around it. Given that many objects are often very similar to their surrounding neighbors and the fact that most pixels suffer from spectral mixing, hyperspectral images are indispensable because of their ability to detect subtle differences in the electromagnetic spectrum. Spectral mixing occurs when a pixel is not homogeneous. For example, if an image of a leaf was collected and an aphid was on that leaf, the pixel would be mixed. There are subspace projection methods and linear unmixing techniques that allow the removal of undesired components from an image and detect a known target in the presence of mixed pixels. The SOM should be able to decide if the pixel is mixed and place it on the map accordingly.

Classification can be difficult with agriculture because the spectral signature of a plant changes over a growing season. Spectral signatures can also change according to how they are stressed. Stress can occur by under- or over-fertilizing, improper irrigation, and the presences of pests. Though this has a negative impact upon image classification, this is a very useful effect when it comes to management of agricultural resources. When managing a crop, classification of different species is not a high priority in analyzing hyperspectral data. The top priority is to ensure a healthy crop, one that is not stressed, which will maximize growth. After analyzing an image cube and determining how a crop is stressed, proper steps can be taken to rectify the situation; but this process can be seen as classification of variously stressed crop. Equipment can be sent to specific areas to treat the crops according to their specific needs. Multi-spectral satellites must be the final platform for remote sensing when it comes to agricultural management. The reasons for this are the enormous monetary and time cost associated with flying a sensor as well as collecting terabytes of data.

CHAPTER IV

INITIAL APPROACH TO PROGRAM CREATION

There were two initial reasons for finding a way to reduce the amount of data that is stored. The first reason is that hyperspectral data files are extremely large. They require massive data storage devices and take long periods of time to process. The second motivation came from researchers desires to know the locations of relevant spectral bands in order to implement other data analysis techniques such as NDVI (Normalized Difference Vegetation Index).

$$NDVI = \frac{NIR - red}{NIR + red} \qquad (4)$$

Where NIR (Near Infra-Red) is the reflectance value of a band around 1000nm and red is a value around 600nm. By offering a list of the most relevant spectral bands, NDVI could be more useful. This is just one example; there are many other programs that could benefit by being presented with only the most relevant bands.

As presented in the conclusion of Chapter 2, by employing a couple of techniques to the basic SOM, critical knowledge of the data can be obtained. After experimenting with various neural network platforms, Matlab was chosen. It was chosen because of the easy to use and well documented somtoolbox2, a program that runs under Matlab. Esa Alhoniemi, Johan Himberg, Juha Parhankangas and Juha Vesanto, the authors of somtoolbox2, offer this program for free. Somtoolbox2 has several capabilities. First and foremost, it will create a SOM if presented with the data and some input parameters. The second qualifying ability is that it keeps track of the AUD, as A. P. Azcarraga refers to it and as will be done throughout this paper, or quantization error as Alhoniemi *et al* refer to it. Another exciting attribute of this program is that it has a walk through tutorial explaining how to use most of the graphing features built into the program. Once this program was explored and chosen, programming for analysis of hyperspectral data began.

Initial developments were made using data recorded with a Geophysical Environmental Research 1500® (GER). The GER records less data than the ASD; therefore by using the GER data, as opposed to ASD data, processing time could be minimized. The program, that still needed to be developed for this project, required three phases: read the data, processes the data, and, finally, to present the data in a way that could be easily analyzed.

Reading in the GER data proved to be fairly simple. Somtoolbox2 contains a

```
intake =

            data: [80x512 double]
          labels: {80x1 cell}
      comp_names: {512x1 cell}
       comp_norm: {512x1 cell}
            type: ‘som_data’
          LABELS_NAMES: {80X1 CELL}
                NAME: ‘WEEDS2.M’
```

Figure 4-1. Example printout of a SOM data struct

program, som_read_data that is fairly universal. It reads data from an ASCII file, which is expected to be in som_pak format. The GER data that was provided was in SNNS format. With a simple modification to the GER data file and to the som_read_data program, the data was loaded into the proper variables. The various variables are all part of the SOM data struct. Figure 4-1 shows a typical display of a SOM data struct, where “intake” is the variable that receives the data read from a file. “data” contains 80 samples of GER data. “labels” contains the label for each sample, such as “CASOB.” “comp_names” contains the wavelength at which each band was recorded. “comp_norm” contains the normalizing variable for each wavelength if normalization is requested for the initial read, otherwise it contains nothing.

The next step is to process the data. This was initially envisioned to be a three-step process: make the original SOM, extract the relevant spectral bands, and finally to present a graph of the AUD and the plot of the final SOM created using just the extracted spectral bands.

Making the original SOM requires some planning. There are several options when making an SOM, which include specifying: initialization function, training algorithm, map size, map lattice, map shape, neighborhood function, and training rate. All of which have a default that works nicely, but in order to achieve optimum results each option had to be explored.

The first option is the initialization function. There are two settings for this, random and linear. For an undetermined reason, the linear initialization setting results in lower AUD and is a more repeatable process. In addition, the linear initialization setting is the preset default option in the program. It is for these two reasons that the linear initialization setting was chosen.

The second option is the training algorithm; the three available settings for this option are sequential, batch, and sompak. All three settings were explored at length, and the batch setting was determined to be the best option. Again, it happened that the batch training algorithm setting was the program's preset default option.

The third option is map size. The settings available for this are multi-fold: small, medium, and large, or grid size specification or the preferred number of units in the map. Only one of these options can be chosen at a time; in other words, the user can define either the grid size or the number of map units, but not both simultaneously. This is because they are mutually exclusive. The program automatically determines the exact grid size if small, medium, or large is specified. If grid size is specified, then the user set grid size will be used. The last option, preferred number of units, leaves the exact number of units in the x and y direction up to the program to decide, but x times y will result in a number close to that specified by the user. In this instance, an 8 x 8 map (or the grid size specification setting) was determined to deliver the results in a more preferable format than that of the other settings.

The fourth option, lattice, also has to do with the format of the resulting map. The map grids can appear as either rectangles or hexagons as seen in Figure 4-2. While the hexagon option occasionally outperformed the rectangular option, the rectangular option was selected for the overall project. There was only a marginal difference between the performances of the two options; quite simply, the rectangular option provided a more concise and clear result.

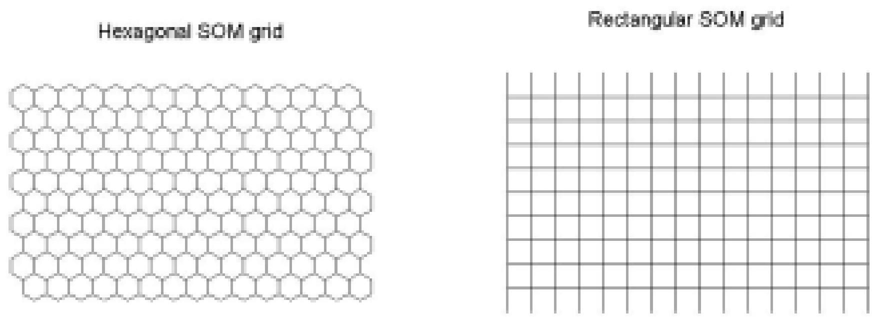


Figure 4-2. Hexagonal and rectangular grids used for SOMs

The fifth option, map shape, is another aspect of the end result's format. The available options are sheet, cylinder, and toroid. Figure 4-3 shows an example of what the three shapes look like in map form. It can be seen that the different shapes cause the data to interact differently. Since the SOM attempts to put like objects together and different objects away from each other, folding the sheet into a cylinder takes away two of the edge positions forcing the data into a different region. If the cylinder is then folded into a toroid, there are no edge positions. This forces the data into yet another region. For this project, the sheet setting was chosen; again, the sheet setting provided the clearest visualization of the results. Future researchers should note that more in-depth exploration of the map shape option may result in further enhanced classifications.

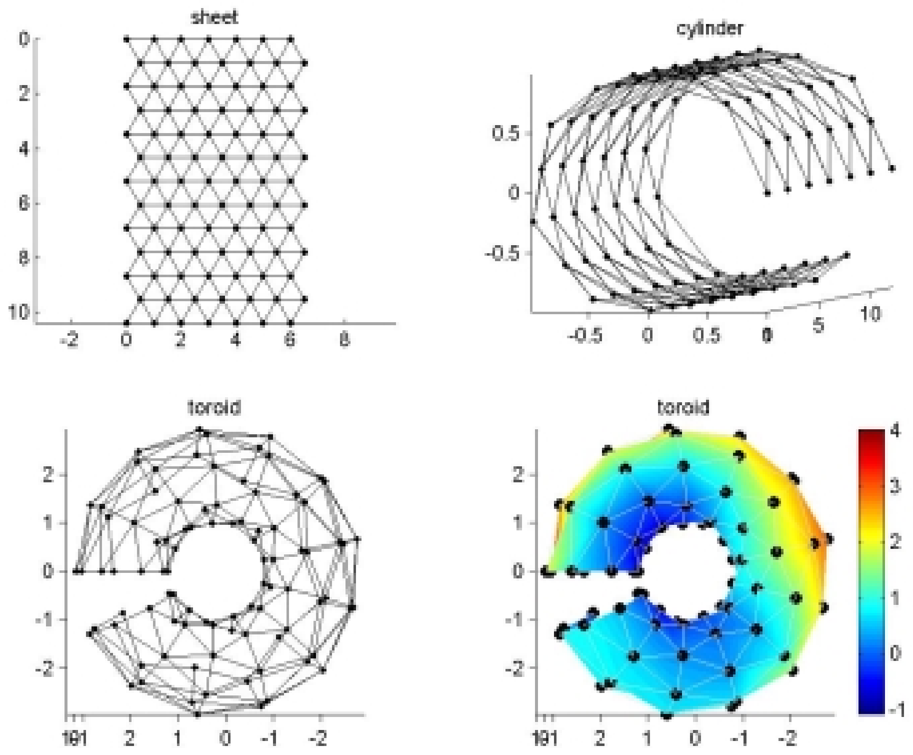


Figure 4-3. Sheet, cylinder, and toroid SOMs

The sixth option is neighborhood function. The available options for this are bubble, Gaussian (the default), cutgauss, and ep, which are defined by equations 5, 6, 7, and 8, respectively. These are the functions that are used to try to fit the map to the data. As stated in the som_batchtrain function,

> In principle the updating step goes like this: replace each map unit by the average of the data vectors that were in its neighborhood. The contribution, or activation, of data vectors in the mean can be varied with the neighborhood function [given by equation 9]. This activation is given by matrix H. So, for each map unit the new weight vector is where i denotes the index of data vector. Since the values of neighborhood function h_i are the same for all data vectors belonging to the Voronoi set of the same map unit, the calculation is actually done by first calculating a partition matrix P with elements p_ij=1 if the BMU of data vector j is i.

Gaussian worked best for organizing agricultural data onto an 8 x 8 sheet map. It is possible that some of the other functions may work better for maps of different shapes or sizes.

$$H = \begin{cases} \dfrac{1, Ud \leq r}{0, Ud \geq r} \end{cases} \tag{5}$$

$$H = e^{-\frac{Ud}{2*r}} \tag{6}$$

$$H = \begin{cases} \dfrac{e^{-\frac{Ud}{2*r}}, Ud \leq r}{0, Ud \geq r} \end{cases} \tag{7}$$

$$H = \begin{cases} \dfrac{1 - \frac{Ud}{r}, Ud \leq r}{0, Ud \geq r} \end{cases} \tag{8}$$

$$m = \frac{\sum hi * di}{\sum hi} \tag{9}$$

The last option, training rate, has three settings. The default setting defines the number of epochs, or training cycles, as seen in equation (10) where *mpd* is the number of map units divided by the number of samples as given in equation (11).

$$epochs = 20 * mpd \tag{10}$$

$$mpd = munits \,/\, dlen \tag{11}$$

$$epochs = 20 * mpd \,/\, 4 \tag{12}$$

$$epochs = 20 * mpd * 4 \tag{13}$$

The training rate can also be set to short or long. In the short setting the number of epochs to be used is divided by four as seen in equation (11) and the long setting results in a multiplication by four as seen in equation (12). After experimenting with the three different settings, it was found that the default setting resulted in the best organization of the map.

With the SOM created and optimized, the process of extracting relevant spectral bands began. By employing a version of equation (1), a program called som_bandfinder was written. The exact details of this program will be discussed further in Appendix C. som_bandfinder accepts in as arguments: a SOM, a set of samples, an integer constant that increases the number of bands kept, and a threshold percentage. The SOM used was created, as described previously, by using GER data, employing the linear initialization function, the batch training algorithm, a grid size of 8 x 8, a rectangular map lattice, map shape set to sheet, Gaussian neighborhood function, and the default training rate. The set of samples was the same GER data that was used to create the SOM. Various constants were chosen as well as various threshold percentages. The best combination of which was a constant of one and a threshold of ten percent. This yielded forty-one relevant spectral bands ranging from the start of the recorded spectrum to the end of the recorded spectrum.

The return from som_bandfinder is the indices of the recommended relevant bands. With the relevant bands located, som_modify_dataset, a built-in function in somtoolbox2, is utilized. The function som_modify_dataset can add/remove components/samples from a dataset by specifying the indices of the dataset that need to be added or removed. The result from this function is a dataset containing only the relevant spectral bands. This leads to the final stage of the preliminary design.

Analyzing the data is the most difficult part. There are many different ways in which the data can be viewed and analyzed. After investigating the options, five graphs—two of which overlap—were chosen to display the data: the U-matrix, a hit histogram, a labeled grid, k-means clustering, and the AUD.

The U-matrix is the backbone of an SOM. The U-matrix, as seen in Figure 4-4, is a shaded grid containing the spatial distance between units. For example, the

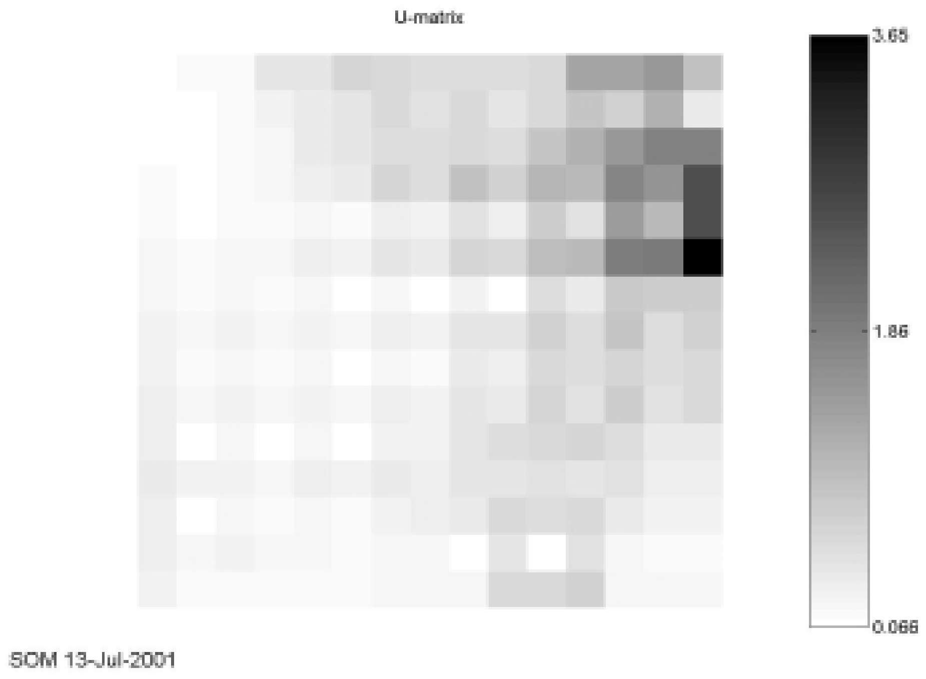
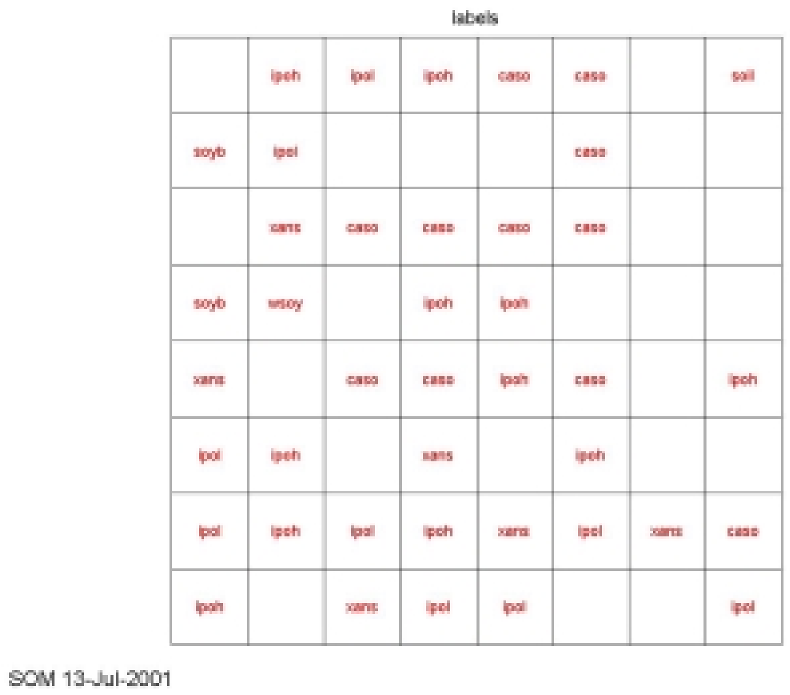


Figure 4-4. U-matrix of an SOM

unit in the top right corner of Figure 4-4 is further away from the unit in the bottom right corner than the unit in the top left corner is from the unit in the bottom left corner. By examining Figure 4-4, it can be see that whatever the SOM put in the top right corner is the least similar of all samples to any other sample on the map. Once the U-matrix has been created, labels can be applied to the map.

There are four ways in which to apply labels to the SOM. Somtoolbox2 provides a function called som_autolabel, which has labeling options of add, add one, frequency, and vote. Add, the least aesthetically pleasing option, puts a copy of the label of each sample into the unit that it best matches. Add one is similar to add except that add one only adds one instance of each label to a unit that is matched. Frequency is extremely useful when calculating classification statistics. The frequency option is similar to add one in that it adds one instance of each label, but the frequency option also puts a multiplier on the end of the label representing the number of samples of that type that were mapped to that unit. Vote was the option that was initially chosen for analyzing the data. The vote option keeps only the label with the most instances. This seems like the best way to make a decision as to how a sample mapped to a unit should be classified. An example of this can be seen in Figure 4-5. From Figure 4-5, it can be seen that soil is placed in the top right corner. This makes intuitive sense, since all the other samples are vegetation and soil should not be easily confused with vegetation.



labels

	ipoh	ipol	ipoh	caso	caso		soil
soyb	ipol				caso		
	xans	caso	caso	caso	caso		
soyb	wsoy		ipoh	ipoh			
xans		caso	caso	ipoh	caso		ipoh
ipol	ipoh		xans		ipoh		
ipol	ipoh	ipol	ipoh	xans	ipol	xans	caso
ipoh		xans	ipol	ipol			ipol

SOM 13-Jul-2001

Figure 4-5. SOM 8 x 8 grid with labels chosen by vote

The third data visualization method chosen was a hit histogram. In order to create a hit histogram, a program that separated the data by type and assigned a color to each type had to be created. Creating the function som_colorhits did this. Further details about the operation of som_colorhits will be discussed in Appendix C. Once som_colorhits was created, it was overlaid on the U-matrix. This can be seen in Figure 4-6. Figure 4-6 is an example of a preliminary results map. Though there is some order to the map, clustering is not very strong and there are several map units with multiple types mapped to it. If multiple types are mapped to the same unit, the SOM is not finding a way to distinguish between those samples of those types.

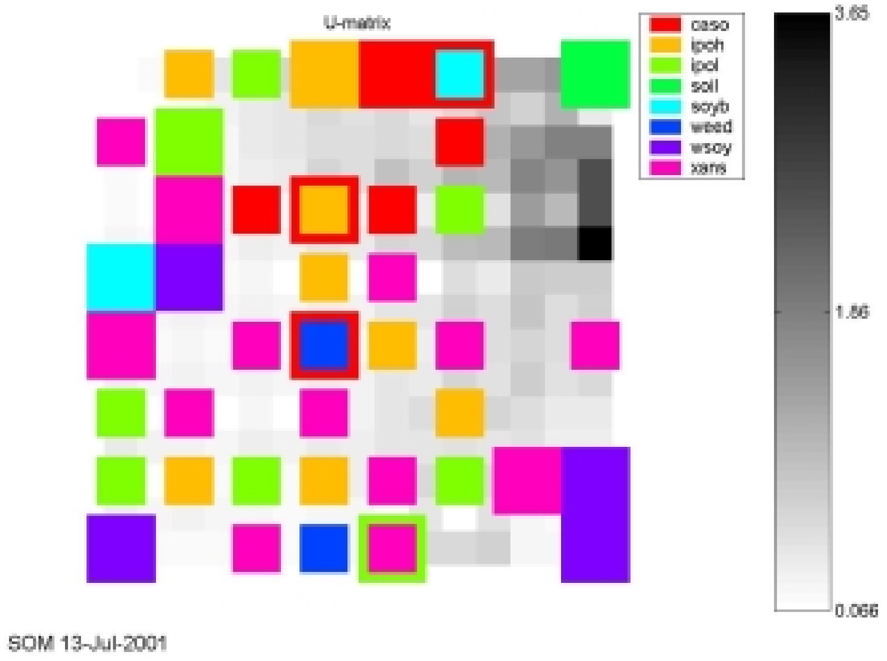


Figure 4-6. U-matrix with hit histogram overlaid

The fourth visualization method employed was to have the map determine the best grouping as decided upon by k-means clustering. Somtoolbox2 contains a function that performs this task called som_kmeanscolor. An example of this can be seen in Figure 4-7. Figure 4-7 has nine subplots; the first eight show the clustering according to k-means with k taking on the value of one through eight respectively. The last subplot shows the best clustering according to minimum quantization error. By examining Figure 4-7, once again it can be seen that the unit in the top right-hand corner is definitely different from the rest of the units in the map. In every iteration of the program, with exception to the first one, the top right corner is separated from the rest of the map. The best clustering subplot of Figure 4-7 shows four groups.

When comparing the best clustering of Figure 4-7 to the labels of Figure 4-5, it can be seen that the four groups are soil, casob, a mixture of xanst, iphol, and iphog, and a

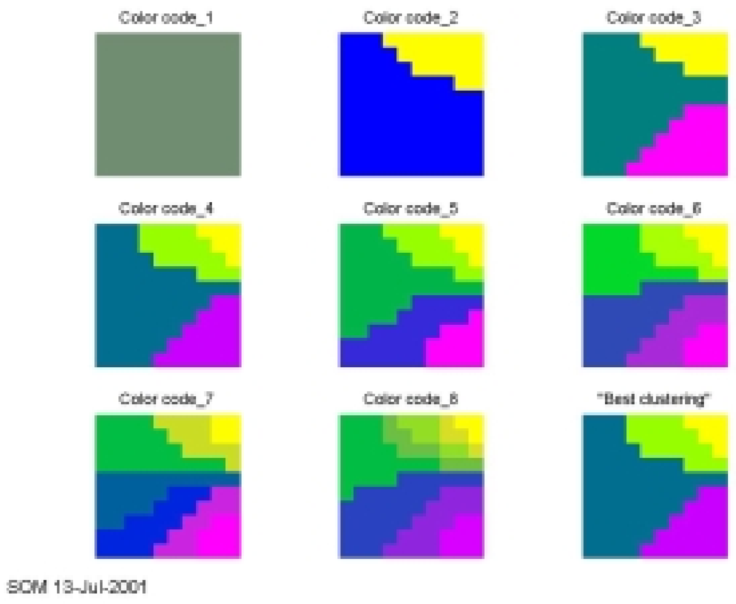


Figure 4-7. K-means cluster graphs of GER data

final group that consists of the outliers of every type. For preliminary testing this was an encouraging result. Separating soil from vegetation should be easy for most any discrimination program, but separating different types of vegetation can be vary difficult and the program automatically separated soil from the other samples.

The last method used for analyzing the data was the AUD. As stated before, the AUD graph monitors the organization of the map. In addition to the training parameter given to som_make, som_make can also be told to return a graph of the AUD by setting the tracking to three. Since A. P. Azcarraga has studied the AUD graph and determined that its shape should be similar to that of Figure 2-1, it was determined that adjustments that caused the AUD graph to become more similar to the shape of Figure 2-1 were beneficial. Figures 4-8 and 4-9 show an SOM AUD with all bands and with relevant bands respectively.

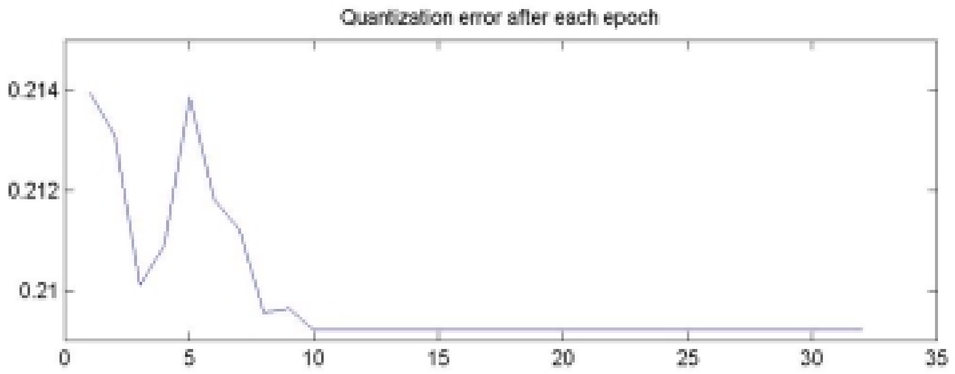


Figure 4-8. AUD graph of an 8 x 8 SOM using all bands of the GER data set

Having successfully completed the preliminary process of reading in the data, processing the data, and viewing acceptable results, the fine-tuning and application of data phase began.

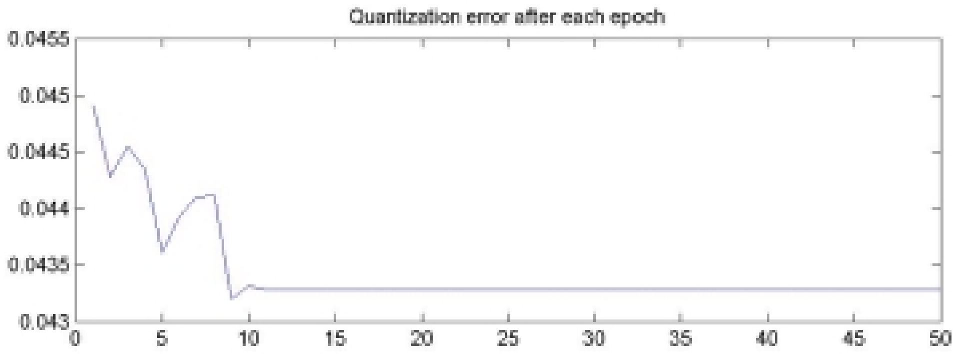


Figure 4-9. AUD graph of an 8 x 8 SOM using 41 bands of the GER data set

CHAPTER V

FINE-TUNING AND APPLICATION OF DATA

This chapter will discuss the fine-tuning and application of data phase, including data collection, format conversion, and troubleshooting of anticipated and unanticipated issues with both the data and the programming. In addition, there will be discussion of data file creation, header file creation, filtering of the data, additional visualization tools, and improved classification parameters.

5.1 Data Collection

Please refer to Appendix A, Materials and Methods, for the full specifications for data collection that was applied for the data used in this project. As regards to the data collection criteria, the specifications state:

> Weed species evaluated were pitted morningglory, entireleaf morningglory, sicklepod, and common cocklebur. Mixed vegetation including experimental units (EU) of soybean with a mixed weed composition and EU's of mixed weed composition without soybean, and bare soil were also included in the study for comparison.

This is further illustrated by Table 5-1.

Table 5-1. Data Collection Names and Classifications

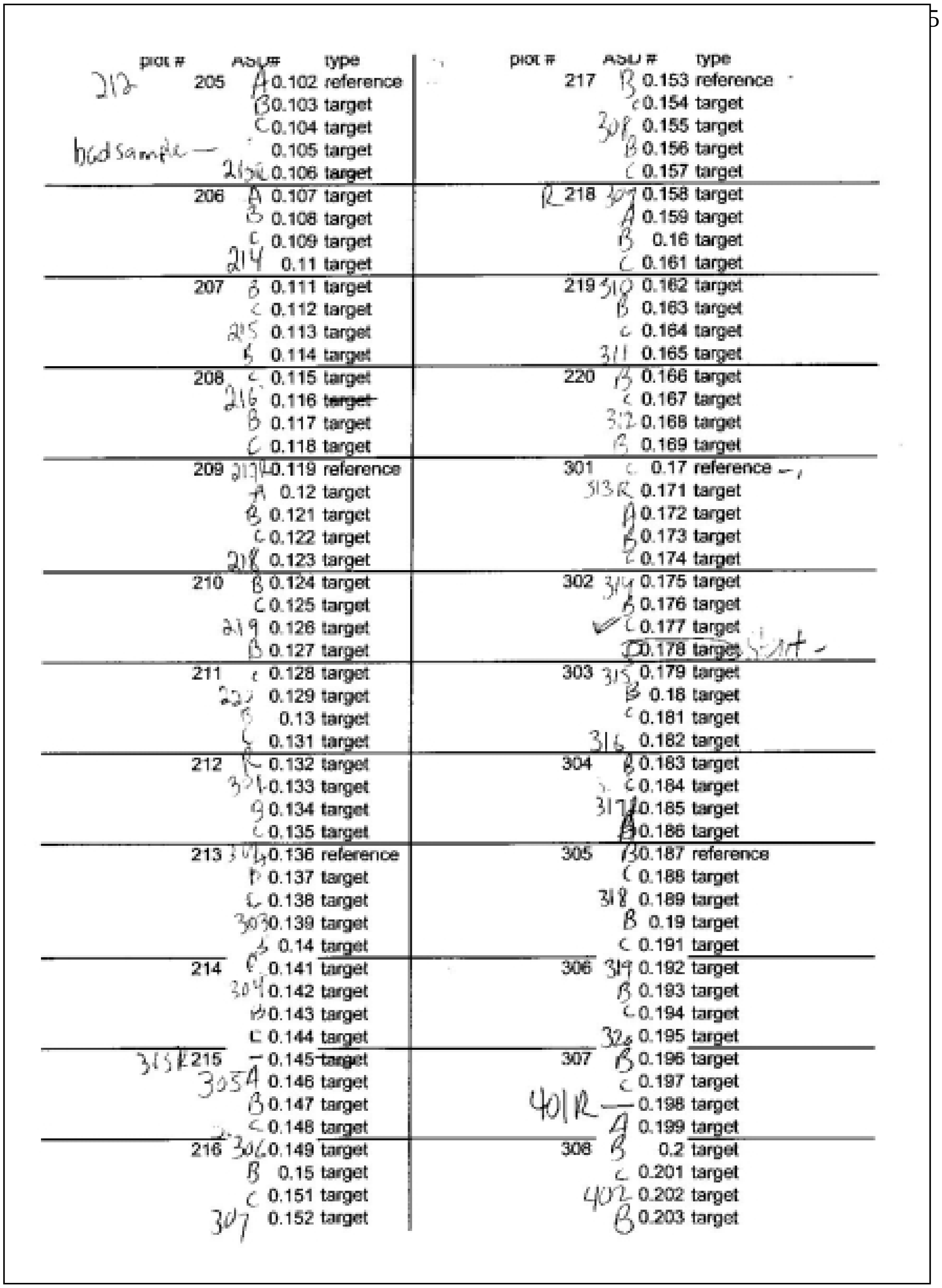
Classification	**Acronym Used**	**Description**
Pitted morningglory	IPOL	Weed
Entireleaf morningglory	IPOH	Weed
Sicklepod	XANS	Weed
Common cocklebur	CASO	Weed
Weeds	weed	Excludes soil and soyb
Soybean	Soyb	Plant
Weedy Soybean	Wsoy	Excludes soil
Soil	soil	soil

The data recorded by the ASD is stored in binary format and contains metadata including instrument number and date and time of acquisition. Each sample is collected and stored into a binary file with the name of the file being the date of acquisition and a three-digit extension that increments with each sample collected, as seen in Figure 5-1.

```
072500.001
072500.018
072500.028
```

Figure 5-1. Sample List of ASD Files

In this example, “072500.001” is XANST, “072500.018” is “weedy soybean,” and “072500.028” is “soybean.” All three files were collected on July 25, 2000; hence, the same prefix in use on each file. In addition, these files must be manually labeled by the field technician for classification. The labels were hand-written on a table as seen in Figure 5-2.



	plot #		ASD #	type
212	205	A	0.102	reference
		B	0.103	target
		C	0.104	target
bad sample —			0.105	target
		213R	0.106	target
	206	A	0.107	target
		B	0.108	target
		C	0.109	target
		214	0.11	target
	207	B	0.111	target
		C	0.112	target
		215	0.113	target
		B	0.114	target
	208	C	0.115	target
		216	0.116	~~target~~
		B	0.117	target
		C	0.118	target
	209	217R	0.119	reference
		A	0.12	target
		B	0.121	target
		C	0.122	target
		218	0.123	target
	210	B	0.124	target
		C	0.125	target
		219	0.126	target
		B	0.127	target
	211	C	0.128	target
		220	0.129	target
		B	0.13	target
		C	0.131	target
	212	R	0.132	target
		301	0.133	target
		B	0.134	target
		C	0.135	target
	213	302R	0.136	reference
		B	0.137	target
		C	0.138	target
		303	0.139	target
		B	0.14	target
	214	C	0.141	target
		304	0.142	target
		B	0.143	target
		C	0.144	target
315R	215	—	0.145	~~target~~
		305A	0.146	target
		B	0.147	target
		C	0.148	target
	216	306	0.149	target
		B	0.15	target
		C	0.151	target
		307	0.152	target

	plot #		ASD #	type	
	217	B	0.153	reference	
		C	0.154	target	
		308	0.155	target	
		B	0.156	target	
		C	0.157	target	
R	218	309	0.158	target	
		A	0.159	target	
		B	0.16	target	
		C	0.161	target	
	219	310	0.162	target	
		B	0.163	target	
		C	0.164	target	
		311	0.165	target	
	220	B	0.166	target	
		C	0.167	target	
		312	0.168	target	
		B	0.169	target	
	301	C	0.17	reference	—,
		313R	0.171	target	
		A	0.172	target	
		B	0.173	target	
		C	0.174	target	
	302	314	0.175	target	
		B	0.176	target	
		✓ C	0.177	target	
		R	0.178	target	Start —
	303	315	0.179	target	
		B	0.18	target	
		C	0.181	target	
		316	0.182	target	
	304	B	0.183	target	
		C	0.184	target	
		317A	0.185	target	
		B	0.186	target	
	305	B	0.187	reference	
		C	0.188	target	
		318	0.189	target	
		B	0.19	target	
		C	0.191	target	
	306	319	0.192	target	
		B	0.193	target	
		C	0.194	target	
		320	0.195	target	
	307	B	0.196	target	
		C	0.197	target	
401R —			0.198	target	
		A	0.199	target	
	308	B	0.2	target	
		C	0.201	target	
		402	0.202	target	
		B	0.203	target	

Figure 5-2. Example of Handwritten Classification Labels on Data Tables

As is clearly seen in the previous figure, there is not a guaranteed match between hand-written data and the data recorded by the ASD.   Also, there is not a guaranteed preset order to the data results; this is the very definition of raw and unfiltered data. This is one area where further enhancements could be explored and utilized by future researchers. For this process, however, due to the hand-written data's imprecise nature, further visualization techniques were utilized to remove extreme outliers, erroneous classifications, and other extraneous data. In order to utilize the visualization techniques, a header file containing classification labels and subclassification labels had to be created.

5.2 Format Conversion

A header file containing classification and subclassification labels with reference to the file name, as seen in Figure 5-3, was created as a spreadsheet in Microsoft Excel.

```
072500.042      IPOHG110
072500.043      IPOHG110
072500.044      IPOHG111
072500.045      IPOHG111
072500.046      IPOHG111
072500.047      IPOHG111
072500.048      IPOHG112
072500.049      IPOHG112
072500.050      IPOHG112
072500.051      IPOHG112
072500.052      ref152
072500.053      CASOB113
072500.054      CASOB113
072500.055      CASOB113
072500.056      CASOB113
072500.057      CASOB114
072500.058      CASOB114
072500.059      CASOB114
072500.060      CASOB114
072500.061      CASOB115
072500.062      CASOB115
072500.063      CASOB115
072500.064      CASOB115
072500.065      CASOB116
072500.066      CASOB116
072500.067      CASOB116
072500.068      CASOB116
072500.069      ref117
072500.070      IPOLA117
072500.071      IPOLA117
072500.072      IPOLA117
072500.073      IPOLA117
```

Figure 5-3. Sample Header File with Classification and Subclassification Labels

The Excel spreadsheet provided a simple way to manipulate the raw data into a usable format for this process. As can be seen in the last entry of Figure 5-3, 072500.073 is referenced to IPOLA117, which means ASD file 072500.073 is of IPOLA species and was collected in the first plot.

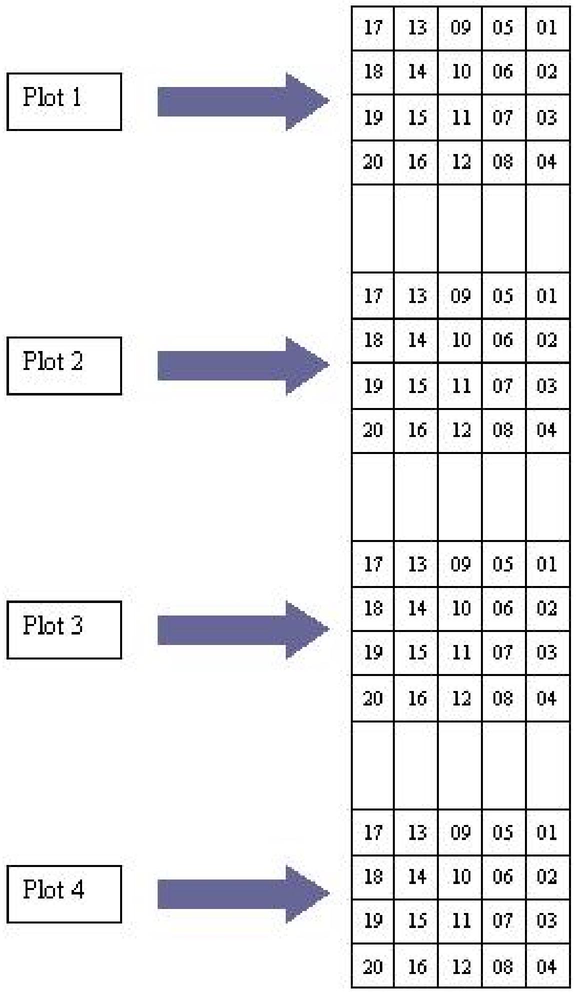


Figure 5-4. Example of Crop Outlay Grid

Figure 5-4 demonstrates the layout of the crops as described in Appendix A. The first digit in IPOLA117 indicates plot 1 while the last two digits indicate that it is grid unit seventeen, as can be seen in Figure 5-5 where IPOLA117 is highlighted in red.

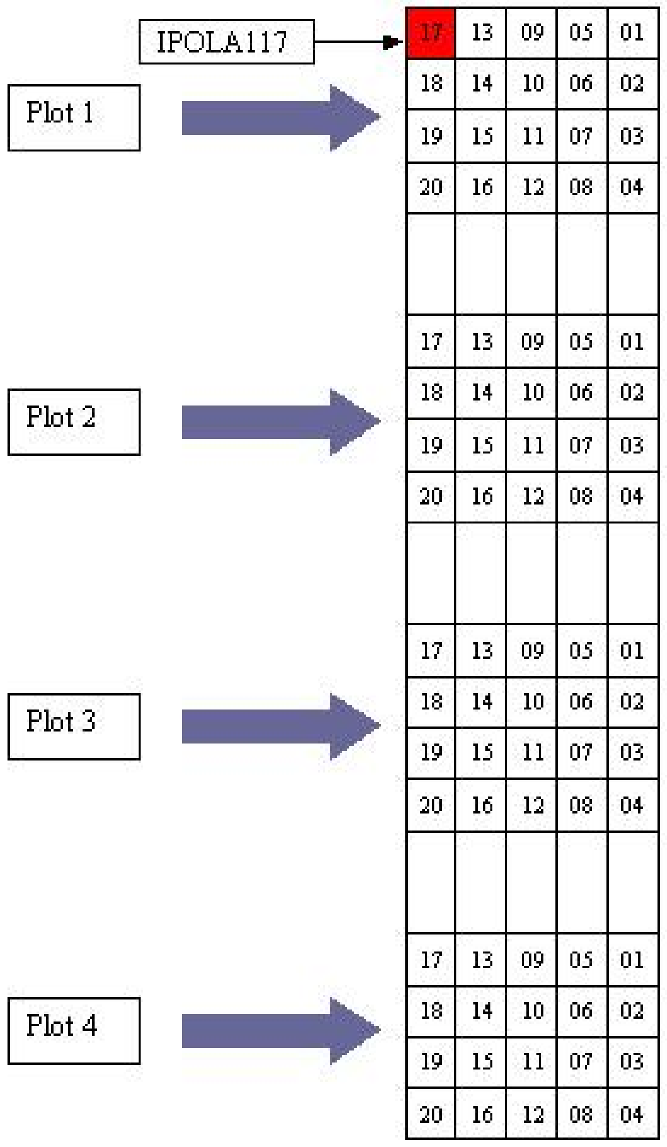


Figure 5-5. Crop Grid with IPOLA117 Highlighted

Conversion of binary files into a single ASCII text document was the necessary next step in the process. ASD ViewSpecPro was used to implement this step. It has a built-in function that allows for ASCII exportation of all files to a single document. It also has a limited capability, as it was unable to convert 612 or more samples into one file. This caused the creation of multiple header files. Since the ASD ViewSpecPro was capable of converting 349 files into a single file—which was the largest number of samples collected on a single date—a conversion function that allows for one file and one header file to be read in and added to the total data pool was necessary, though initial ASD data testing was only performed on the data collected on July 7, 2000.

Two functions were created: asd_read_data and asd_read_header. These functions were created to meet the design requirements as determined during the data acquisition phase. One additional formatting procedure must be done prior to the asd_read_data function being implemented: the data file must be opened using a text editor and "#n " must be inserted before the term "wavelength." The asd_read_data function reads the file created by ASD ViewSpecPro, and the asd_read_header function reads the header file that corresponds to that same data and combines the two sets of information into one single readable format for the next step in the process.

There are two options in the asd_read_header function. The first option allows the user to remove samples from consideration. All samples that have labels that match inputs specified by the user are eliminated. For example, if the user desires to remove reference files, an input of "ref" would accomplish this. The second option allows a choice of label length. By varying the label length of the single file, the user can specify the degree to which the data is subdivided. This is vitally important here as this is where subclassification labels can be utilized. By selecting a label length of four, eight different classes of data will be created. Yet with a simple extension of the label length to six, twenty-four different classes of data will be created. This only applies if the labeling scheme as described previously in this chapter is used. Both lengths (four and six) were used at varying times in this project.

Initially, a label length of four was selected because it allowed a sufficient number of letters to differentiate between classes and fit nicely onto a side-by-side U-matrix with colored hits and 8 x 8 label grid as seen in Figure 5-6. Since this label length had worked for the GER data, it should be sufficient for the ASD data as well.

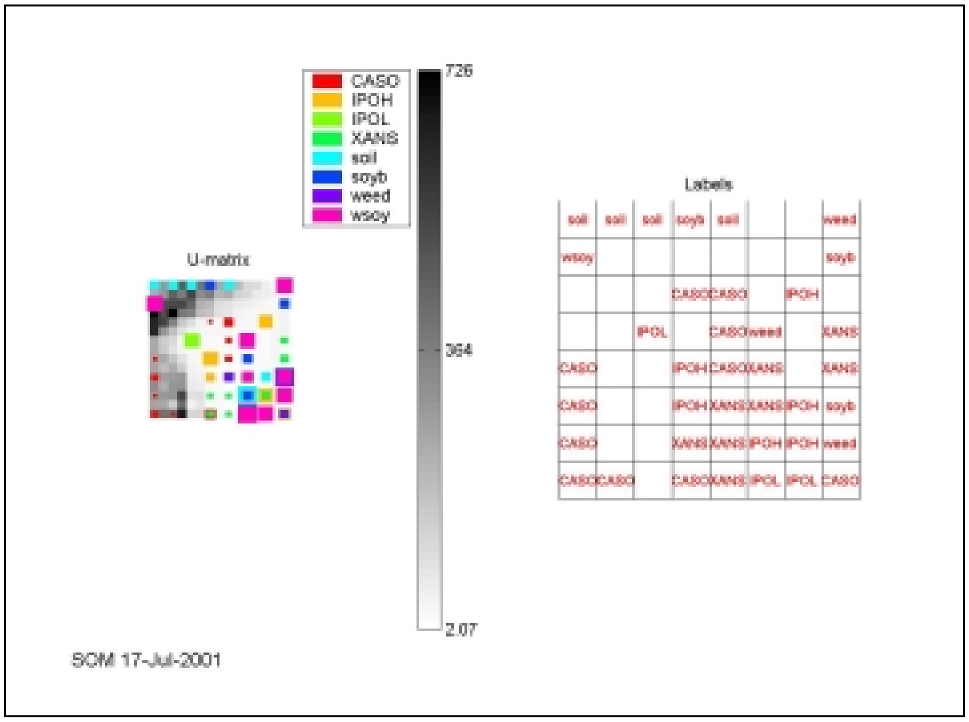


Figure 5-6. Example of Grid with Bad Samples

Several trial runs produced SOMs similar to those seen in Figure 5-6. It can be seen that some of the classes are separated nicely, such as those of the “CASO” in the lower left corner and “soil” in the top left. However, there are several samples that should not be placed where they are, such as the “soyb” sample that is in with “soil.” The most significant setback encountered at this point is that of the bottom right corner of the U-matrix, which is enlarged in Figure 5-7.

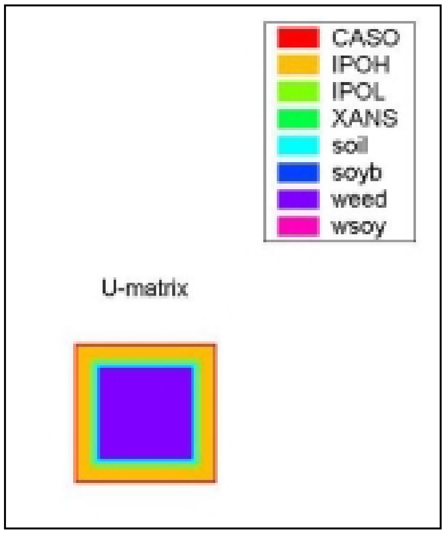


Figure 5-7. Inset of Lower Right Corner of U-Matrix of Figure 5-6

In Figure 5-7 there are five different classes of samples mapped to the same grid component. There are several samples of each class mapped to this grid component as well. This is not a desired result, as it will result in a massive number of incorrect classifications. “CASO” is the largest square, signifying the most hits, so the grid coordinate would receive the label of “CASO”. All other samples mapped to this grid coordinate would then be a misclassification. As “IPOH” has almost as many hits as “CASO” this would drive the classification rate, for this cell, close to fifty percent. Adding to the misclassification total, the hits resulting from “IPOL,” “soil,” and “soybean” would drive the classification rate of this cell to a number much closer to zero.

Given that the SOM was resulting in a somewhat organized map, it was determined that a vast number of the misclassifications were coming from either mislabeled data or extreme outliers. Since at that time there was no way of knowing exactly which samples were erroneous, the som_classplots function was created. This function plots a graph for each class, plotting all class respective samples to each graph. This is also the first step in which the label length specification of six was used. Setting the label length to six allowed the sample to be subdivided into smaller groups making it easier to locate the erroneous samples. Figure 5-8 shows an example of the first set of graph created by som_classplots.

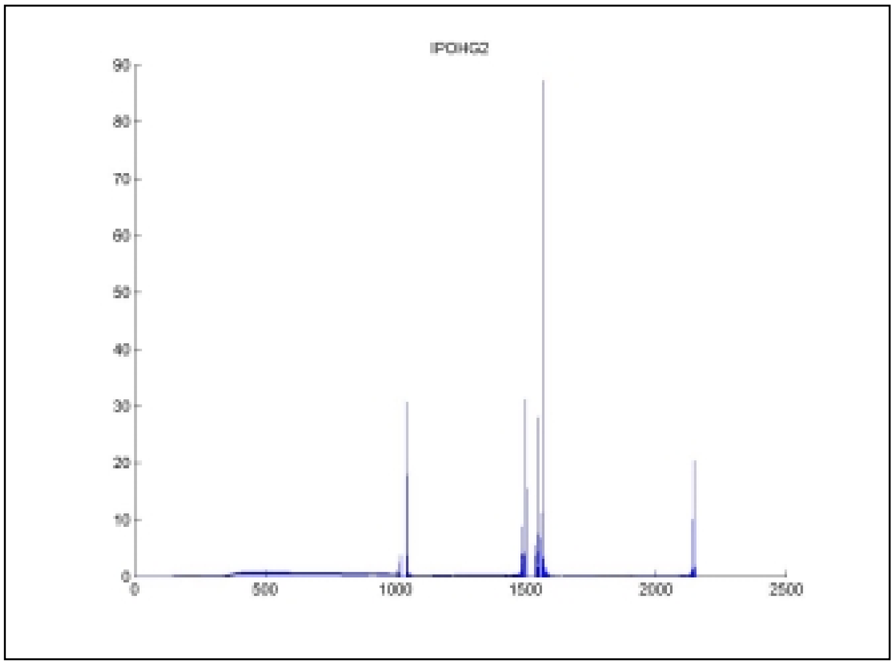


Figure 5-8. Example Graph of Bad and Mislabeled Data with All Bands

In Figure 5-8 it can be seen that several bands contain values inconsistent with the rest of the signal. These bands are referred to as water bands. They are bands in which water in the atmosphere plays a large role in determining the amount of energy that is transmitted. Since these bands mostly contain information about the atmosphere and not about the desired targets, they were eliminated from consideration. Since the beginning wavelength is 350nm the band number and band wavelength differ by this amount. This is the reason Figure 5-8 does not have values past 2150. Somtoolbox2 provides a data manipulation function called som_modify_dataset. In this function, as previously stated in Chapter 4, som_modify_dataset can add/remove components/samples from a dataset by specifying the indices of the dataset that need to be added or removed. The band ranges chosen for removal were 1285nm to 1500nm and 1763nm to 2500nm. Figure 5-9 is an example of the first set of graph created by som_classplots with the water bands removed.

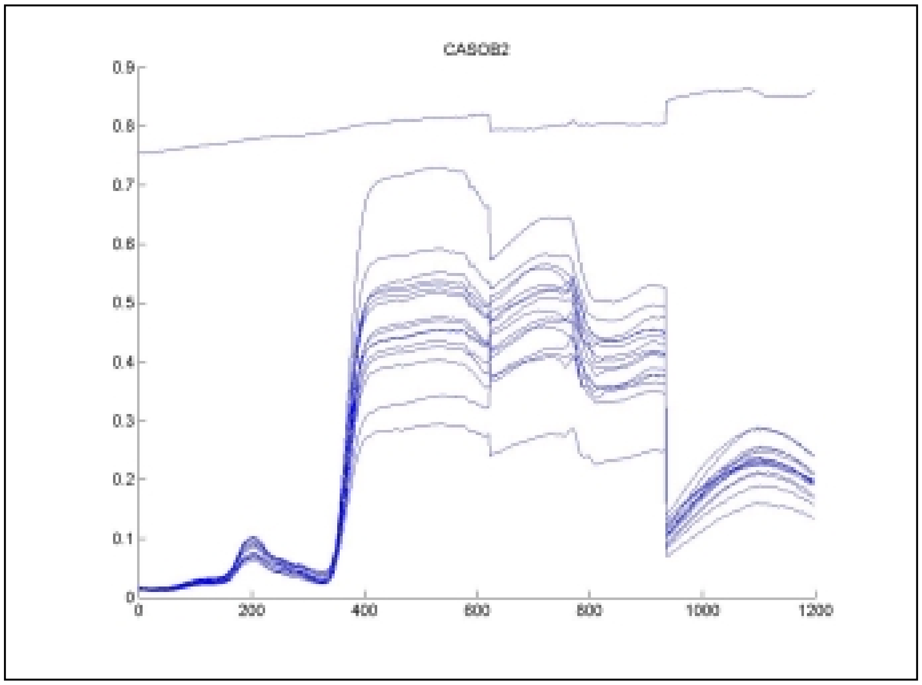


Figure 5-9. Example Graph of Mislabeled Data with Water Bands Removed

In Figure 5-9 it can easily be seen that one of the samples is a misclassification. Since the sample cannot be correctly identified, the sample should be removed. Knowing that the sample is a "CASOB" in plot 2 allows for cross-referencing of the file numbers by means of the header file. ASD ViewSpecPro can then be used to manually locate the mislabeled file. At this point, there are two options: delete the mislabeled file and recreate the data file or simply type "ref" in front of the label in the header file. Either option will result in the sample not being used.

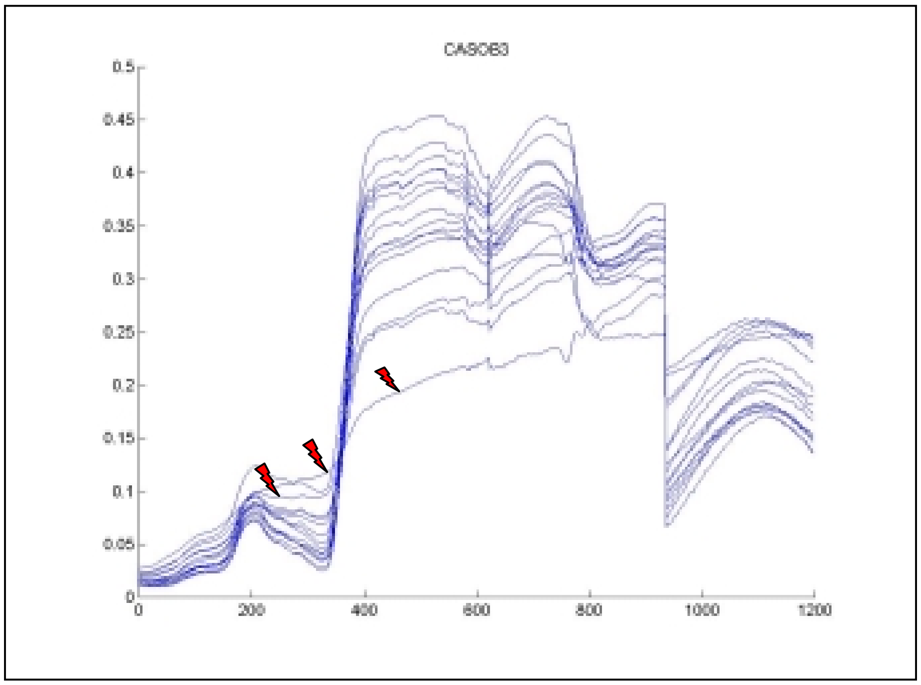


Figure 5-10. Example Graph of Bad Data With Water Bands Removed

The second largest reason for misclassified data can be seen in Figure 5-10. As indicated by the red marks in Figure 5-10, the samples do not have characteristics remotely similar to the group. These samples are known as outliers. Many different situations can cause outliers, such as a gust of wind causing the leaf to move or fatigue of the technician causing the ASD to move. The waveform marked by the middle pointer in Figure 5-10 contains equal energy in the green and red bands, but has little energy in the blue region. This sample, if properly recorded, is of a brown leaf, which is probably dead. Since the objective of this program is to distinguish between different living plants, not dead plants, this sample was eliminated. Whatever the reason for these extreme outliers, these samples were subsequently removed from the data pool.

Another function for identification of outliers was attempted, som_map_classes. In this function, the SOM was utilized. The idea was that since the SOM is specifically designed to separate data, any outliers should be moved away from the mass. Each class of samples was applied a separate SOM and then the SOM was analyzed. There were two problems with this method. The first problem came from the way in which the SOM separated the data. It was difficult to determine which sample was an outlier. Since the SOM did not contain samples of other species, the SOM did not group the samples into a cluster with outliers. The SOM separated the data in other ways, not relevant to species separation. On the rare instances that a sample was distinctly separated, determining which sample was separated was very difficult. In order to make us of this method, much more programming would have been required, including adding a field to the SOM datastruct. Comparing the som_map_classes method with the som_classplots method resulted in abandonment of som_map_classes.

The next step taken to improve the classification rate was to normalize the data and set the range. This was done in two parts: subtract the minimum value of each sample out of each sample and divide each sample by the maximum value in each sample. This was done for several reasons. Since the amount of solar energy reaching the target is not constant throughout the collection phase, the magnitude of a single spectral band without reference to at least one other spectral band is of no use in classification. However, the SOM could decide to separate the samples into groups with high-energy values and low-energy values. This would not be a desired outcome. By forcing the minimum value of a sample to be zero and the maximum value of a sample to be one, the energy value should be more uniform within a class.

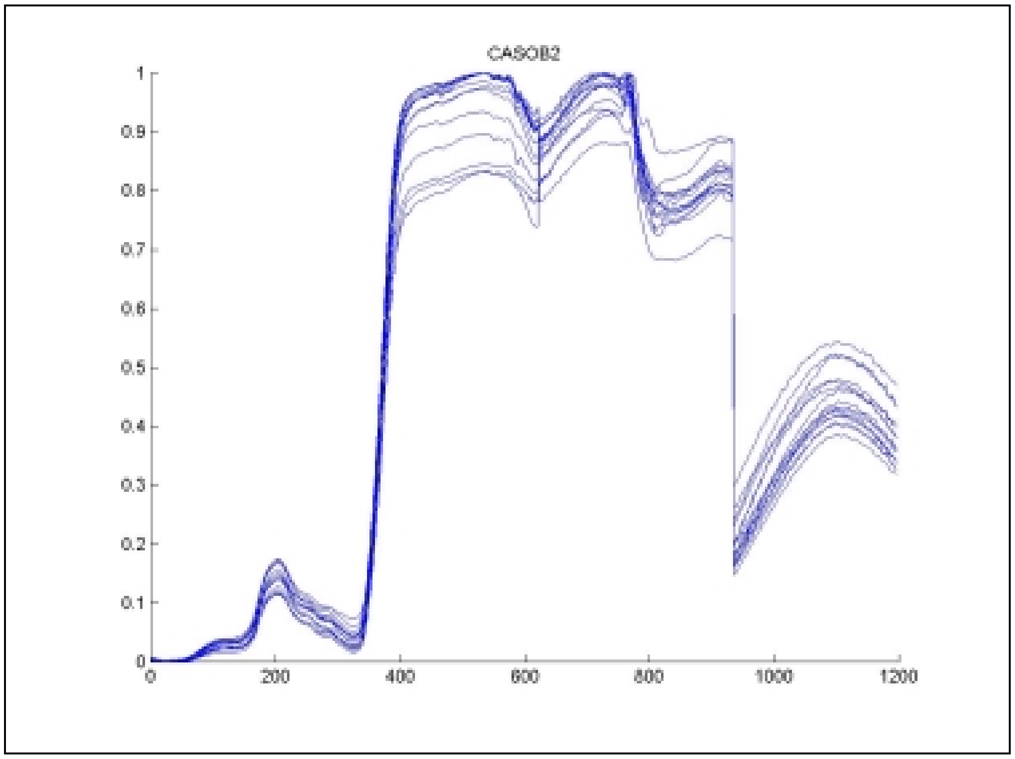


Figure 5-11. Example of Processed Data

Figure 5-11 is an example of a subclass that has had mislabeled samples removed and data values set to a range of zero to one. It can be seen, by comparing Figure 5-9 with Figure 5-11, that these steps result in much more uniform data. This improved classification rates slightly, but not as much as expected. Since classification rates of processed data sets with only water bands removed was still well below fifty percent, additional steps to improve classification rates were needed.

The next step to improve classification was to increase the size of the data set. Combining the sets of ASD data collected July 25, 2000, and August 9, 2000, accomplished this. The combined data set resulted in a classification rate ranging between fifty-six and sixty-two percent, almost doubling the classification rate. There were two reasons that the combined data set resulted in a better classification rate. The first reason was that the data collected on August 9, 2000, was much more consistent than the data collected on July 25, 2000. That is to say that there were not as many obvious outliers especially in the "CASOB" samples. The second reason for an improved classification rate has to do with the nature of neural networks. The more samples of a particular type that a neural network uses to train, the better the network is capable of finding the subtle differences between classes. As combining the two data sets did drastically improve the classification rate, the classification rate was still far below a desired value.

Som_doublesize was created. It doubles the size of classes that have less than half the number of samples of the largest class. One aspect of the som is to display probability density of the data. That means that if a class makes up less than 10 percent of a data set, then the number of cells labeled as that class will probably be less than 10 percent of the total number of cells. In some cases, this can mean complete elimination of a class from the map. Using som_doublesize had various results, but typically increased classification rates, particularly in smaller maps. The unacceptable classification rate then led to further exploration of the somtoolbox2.

Upon further investigation of somtoolbox2, som_supervised was discovered. Som_supervised works similar to som_make except that it adds a vector, with length equal to the number of different classes, to each data sample. The component corresponding to the sample in the new vector is set to one and all other components in the new vector are set to zero. This is only used during the training phase of the map. Once the map is trained, the new vector is removed. Since the component vector of the data is several hundred components and the new vector attached is very short, eight in this case, the overall shape of the map is not severely affected. However, it does cause clustering of classes. The clustered classes provide a large spatial target for the test samples. Most of the misclassifications up to this point have been along the borders of correct labels. Employing the som_supervised function raised classification rates to a range of 72 percent to 76 percent. This is an acceptable classification rate. Once an acceptable level for classification was reached, testing of the som_bandfinder function began.

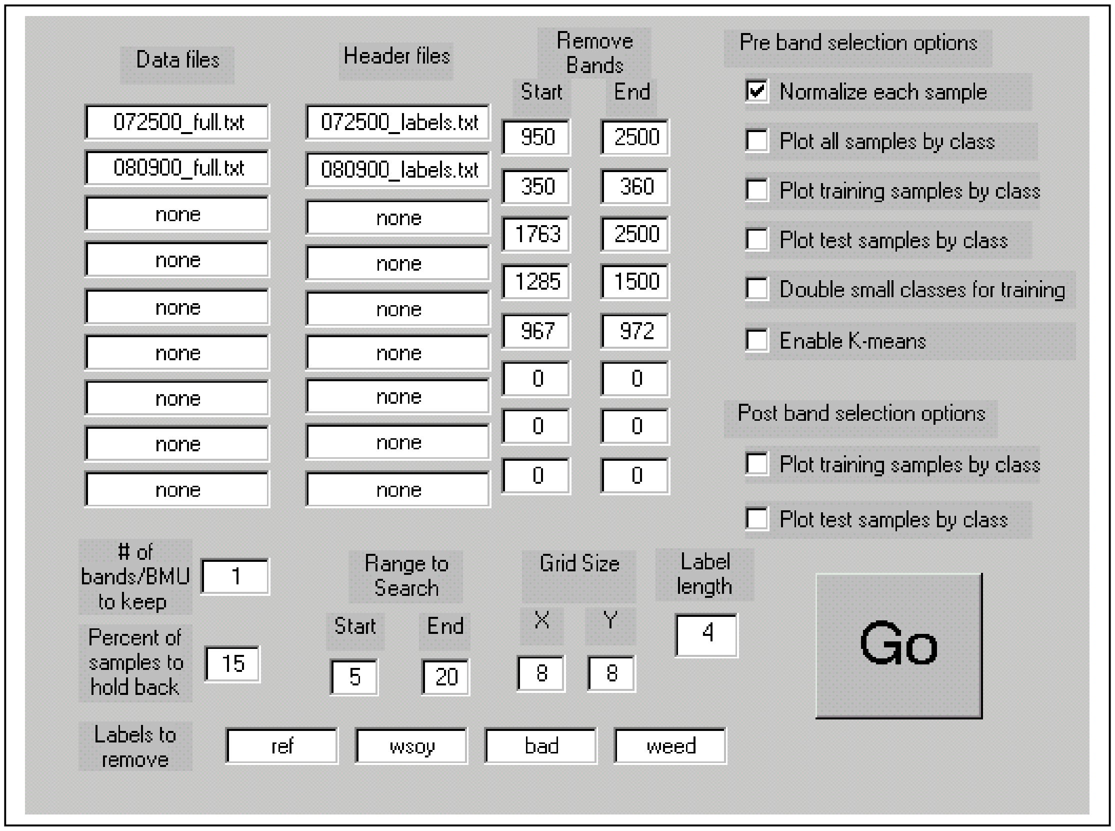
In order to test som_bandfinder, the data had to be split into two sets: a training set and a test set. It is customary to withhold 15 percent of the samples for testing. Testing on data that was used to train the map is, in essence, cheating. In addition, testing on training data does not simulate a real-world application. Because of this, som_split_data was created. Som_split_data takes a data set and a percent as inputs and returns two mutually exclusive data sets containing (1 -%) and (%) of the original data set.

5.3 Testing the Data

Several attributes were explored in the testing phase using the classification rate as the measuring device. The controllable variables that affect the classification rate are threshold, map size, and number of bands per best matching unit to keep.

Threshold directly affects which bands are kept by som_bandfinder, whereas map size and number of bands per best matching unit directly affect the total amount of bands that are kept. But, before testing began, a Graphical User Interface (GUI) was developed.

Figure 5-12 shows somasdgui. It allows for the input of up to nine datasets and their respective header files. In addition, up to eight ranges of bands and four types of samples can be removed. On the right side of the screen there are several options, which include pre and post band selection; graphing; enabling kmeans; and doubling small classes. The lower left-hand corner contains inputs for the control variables.



Data files	Header files	Remove Bands Start	Remove Bands End
072500_full.txt	072500_labels.txt	950	2500
080900_full.txt	080900_labels.txt	350	360
none	none	1763	2500
none	none	1285	1500
none	none	967	972
none	none	0	0
none	none	0	0
none	none	0	0
none	none		

Pre band selection options

- [✓] Normalize each sample
- [ ] Plot all samples by class
- [ ] Plot training samples by class
- [ ] Plot test samples by class
- [ ] Double small classes for training
- [ ] Enable K-means

Post band selection options

- [ ] Plot training samples by class
- [ ] Plot test samples by class

# of bands/BMU to keep: 1

Percent of samples to hold back: 15

Range to Search: Start 5, End 20

Grid Size: X 8, Y 8

Label length: 4

Labels to remove: ref, wsoy, bad, weed

Figure 5-12. Somasdgui Screen Shot

The purpose of the first experiment was to see how the classification rate changed as the threshold was adjusted to a series of values between 1 percent and 99 percent. In Figure 5-13, each trial maintains the same training and test set throughout the range of thresholds. The random number generator—which is used to split the data—was set to state 0, 10, and 100 for trials 1, 2, and 3, respectively.

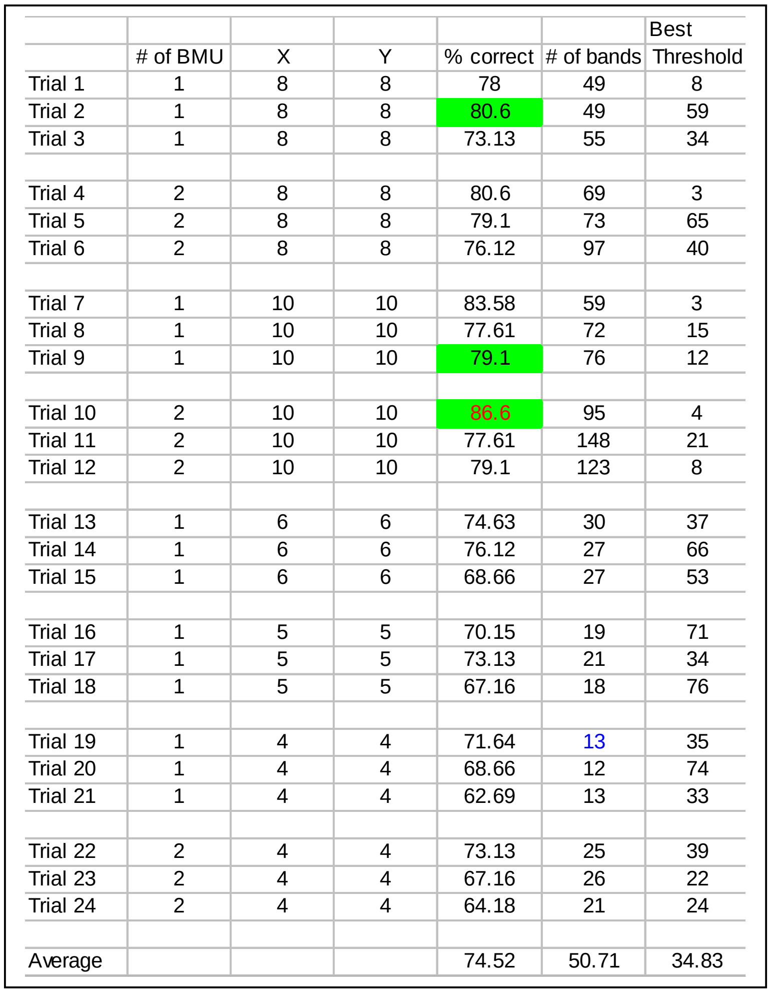
As can be seen in Figure 5-13, the classification rate increases to an acceptable level at a threshold ranging from approximately 10 percent up to about 75 percent. The highest values for threshold percentage occur at 15 and 50 percent, with relatively high classification rates occurring in between these values. This is a desired trend, because, as stated earlier, the threshold should be set high enough so that features that are only relevant to individual samples are not selected. However, if the threshold is set too high, the features that are needed to distinguish between the five classes will be eliminated. The 15 to 50 percent range seems to satisfy this condition.

				Trial 1	Trial 2	Trial 3
range percent	# of BMU	X	Y	% correct	% correct	% correct
1	1	8	8	55.22	44.78	44.78
5	1	8	8	62.69	68.66	64.18
10	1	8	8	76.12	71.64	67.16
15	1	8	8	80.6	71.64	71.74
20	1	8	8	79.1	62.69	68.66
25	1	8	8	73.13	76.12	70.15
30	1	8	8	73.13	71.64	70.15
35	1	8	8	71.64	71.64	73.13
40	1	8	8	76.12	76.12	67.16
45	1	8	8	76.12	71.64	71.64
50	1	8	8	74.63	77.61	65.67
55	1	8	8	77.61	76.12	62.69
60	1	8	8	73.13	68.66	67.16
65	1	8	8	68.66	73.13	70.15
70	1	8	8	65.67	73.13	68.66
75	1	8	8	65.67	62.69	62.69
80	1	8	8	65.67	62.69	62.69
85	1	8	8	67.16	62.69	67.16
90	1	8	8	62.69	71.64	65.67
95	1	8	8	67.16	59.7	64.18
99	1	8	8	68.66	64.18	56.72

Figure 5-13. Classification Results as a Function of Varying Threshold Percentage

The results from the second test of som_bandfinder can be seen in Figure 5-14. There were 24 trial runs, with 8 changes in parameters. Every time a change in parameter occurs, the somasdgui is activated with the random number generator state set to 0, 10, and 100.

In Figure 5-14, map size and number of bands per BMU were varied to explore their effects on classification rate and total number of kept bands. Another difference in this experiment versus experiment one is that every threshold in one percent increments was tested to find the highest classification rate. Only the highest classification rate was kept and then, all information resulting from using the threshold associated with the highest classification rate was recorded and can be seen in Appendix D. Important observations that can be made regarding this experiment's results are that classification rates of 80 percent were achieved with as few as 49 bands, yet the best classification rate—86.6 percent—was recorded when 95 bands were used.



						Best
	# of BMU	X	Y	% correct	# of bands	Threshold
Trial 1	1	8	8	78	49	8
Trial 2	1	8	8	80.6	49	59
Trial 3	1	8	8	73.13	55	34
Trial 4	2	8	8	80.6	69	3
Trial 5	2	8	8	79.1	73	65
Trial 6	2	8	8	76.12	97	40
Trial 7	1	10	10	83.58	59	3
Trial 8	1	10	10	77.61	72	15
Trial 9	1	10	10	79.1	76	12
Trial 10	2	10	10	86.6	95	4
Trial 11	2	10	10	77.61	148	21
Trial 12	2	10	10	79.1	123	8
Trial 13	1	6	6	74.63	30	37
Trial 14	1	6	6	76.12	27	66
Trial 15	1	6	6	68.66	27	53
Trial 16	1	5	5	70.15	19	71
Trial 17	1	5	5	73.13	21	34
Trial 18	1	5	5	67.16	18	76
Trial 19	1	4	4	71.64	13	35
Trial 20	1	4	4	68.66	12	74
Trial 21	1	4	4	62.69	13	33
Trial 22	2	4	4	73.13	25	39
Trial 23	2	4	4	67.16	26	22
Trial 24	2	4	4	64.18	21	24
Average				74.52	50.71	34.83

Figure 5-14. Classification Results as a Function of Quantity of Kept Bands

There was a large fluctuation in the bands chosen in trials 1 through 24, but several bands made repeated appearances and were present in more than one trial. The first band, 361 nm, was present in every sample and the last band, 949 nm, was present in most samples. This is due to the characteristics of som_bandfinder; the first band and last band not removed from consideration will be chosen by default in certain situations. In order to find the bands most commonly chosen, and therefore the most relevant, a histogram was created, which can be seen in Figure 5-14. It should be noted that future researchers may certainly utilize alternate techniques for organizing the data created by som_bandfinder.

5.4 Analysis of Results

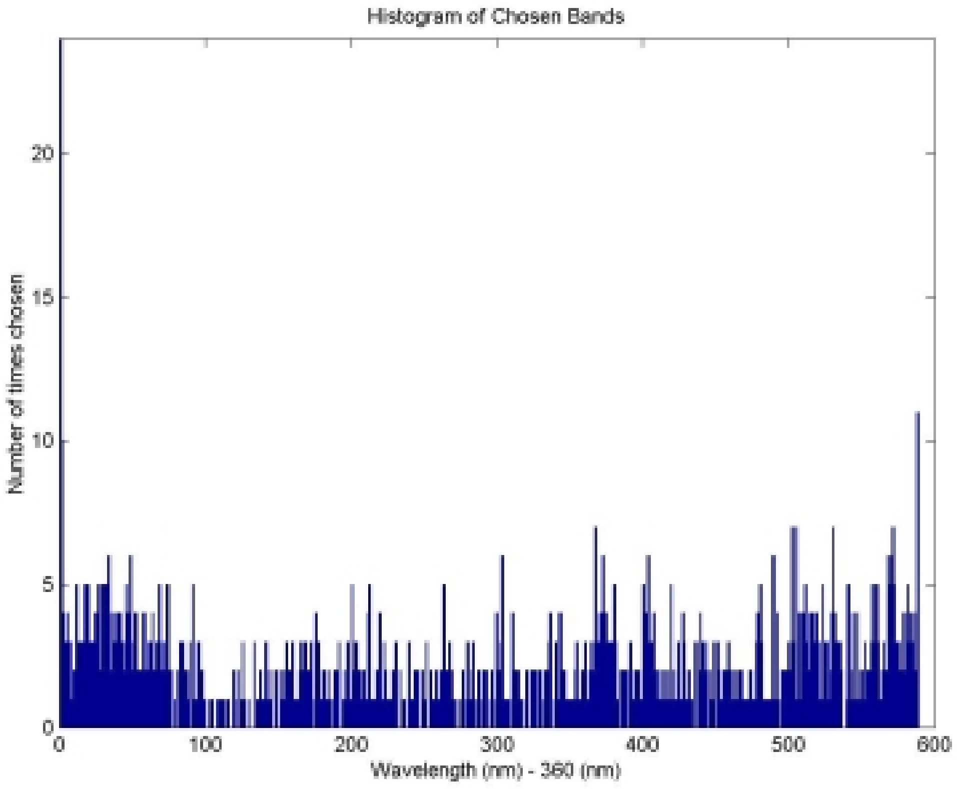


Figure 5-15. Histogram of Selected Bands

In looking at the histogram, it is clear that, despite the unsupervised selection process of som_bandfinder, particular bands are repeatedly chosen in the highest classification thresholds. Since these bands appear to be significant in the classification process, one could theorize that these bands are related to some key physical characteristics that are different in each class of plant species. On the basis of that theory, an additional experiment was performed, the results of which can be seen in Figure 5-16.

Histogram Threshold	***Total # of Bands***	***Map Size***	***% Correct***		
			Trial 1	***Trial 2***	***Trial 3***
5 or more	48	10x10	70.15	80.6	65.67
6 or more	17	10x10	61.19	67.16	64.18
7 or more	8	10x10	61.19	62.69	50.12

Figure 5-16. Histogram Results Experiment

In order to analyze the results of the histogram, three thresholds were examined in-depth. Particular attention was paid to achievable classification rates with relation to total number of bands selected. A map size of 10x10 was chosen due to its prior positive track record in these experiments. Ideally, three separate data sets should have been used for the three trials of the selected thresholds to eliminate any unintentional correlation between the data and the selected bands; however, limited time and resources made that option unfeasible. Instead, the original data set was used for three trials with three different test groups.

In the last experiment, it can be seen that a decent classification rate can be achieved with the eight most commonly chosen bands. Lowering the histogram threshold from seven to six more than doubled the number of kept bands and yet had marginal effect on the classification rate, while lowering the histogram threshold even further to five had a clear and positive effect on the classification rate. It became evident while looking at the results of the final experiment, the histogram, that a more preferable classification rate requires a higher number of the most common bands. This is an understandable result in that keeping additional bands provides additional information, from which the SOM can make better decisions. However, there is the law of diminishing returns stating “the application of additional units of any one input…to fixed amounts of the other inputs yields successively smaller increments in the output of a system of production” [13]. Not only does the classification rate stop increasing, it actually decreases. This could be due to the inclusion of bands not related to the five preset classes.

CHAPTER VI

CONCLUSIONS

As the overall experiment contained many smaller but interrelated experiments, there are several conclusions to make. Some of the conclusions have to do with the approach to the training sets, some with species classification and how these results are obtained, and still others involve the usefulness of the threshold requirement. These conclusions, and more, will be discussed in this chapter.

As obvious as it appears in hindsight, a vital part of the experiment setup necessary for workable/usable results is the preprocessing of the training sets. Through trial and error, it became clear that elimination of extreme outliers—which could be mislabeled data or simply an anomaly—prior to conducting the data-mining portion of the experiment reduces the possibility of the extreme outliers dominating the experiment's findings and ending up with bad results. Oftentimes, the bad results were not immediately obvious as such, and only later in the process would they arise and cause further difficulty in the overall experiment, requiring backtracking and duplication of effort.

Similar to the necessity for elimination of outlier samples, the elimination of irrelevant bands is required and tremendously useful towards obtaining clean and workable results. Determining which bands are relevant and which are irrelevant requires some in-depth knowledge of the physical characteristics of the subject being examined and the conditions in which the data was acquired. In the case of this experiment, it was vitally necessary to remove water bands, contaminated bands, and bands that drastically fluctuated within expected classes. Water bands typically contain information about the atmosphere, contaminated bands are those in which the physical characteristics of the sensor caused minor malfunctions in the gathered data, and the drastically fluctuating bands are in the SWIR (or short wave infrared) range. Removal of these outlaw bands prior to performing the experiments using the data sets not only enabled a faster running time for the experiment, but also resulted in a significant increase in classification rate.

As a side finding to the overall experiment, the self-organized map using the selected bands does an adequate job of species classification. While its classification rate could be improved, it nevertheless shows value for this particular use. Further research using this technique in the field of plant species classification should consider using the self-organized map. However, researchers using the self-organized map who are not familiar with or well versed in the subject's physical properties ought not to rely solely upon it for accurate classification.

Finally, similar to Merkel and Rauber's findings with the label_som, it has been determined through these experiments that a threshold high enough to eliminate bands that are too specific, yet low enough to keep bands that are relevant to species classification is absolutely essential to the success of the experiment when using this approach to data mining. The use of a discriminating threshold provides results that are satisfactory, particularly as pertains to species classification. The bands chosen by setting a selective or proper threshold will generally return bands that are significant in class separation. Future researchers should regard the bands chosen in the final stages of the experiment as a priority for further examination.

REFERENCES

[1] R. L. King, C. Ruffin, L. Lamastus, and D. Shaw, “Classification of Weed Species Using Self-Organizing Maps” Proceedings of the 2nd ICGIA&F, vol. II, pp. 151–158, Jan. 2000

[2] T. Kohonen, “The self-organizing map,” Proceedings of the IEEE, vol. 78, pp. 1464-1480, Sept. 1990.

[3] T. Kohonen, “Self-Organized formation of topologically correct feature maps,” Biological Cybernetics, vol. 43, pp. 59-69, 1982.

[4] P. Somervuo, “Competing hidden Markov models on the self-organizing map,” IEEE Trans. Nerual Networks, vol. 3, pp. 169-174, 2000.

[5] D. J. Willshaw and C. Malsburg, “How patterned neural connections can be set up by self-organization,” Proc. Roy. Soc. London B, vol. 194, pp. 431-445, 1976.

[6] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, “Engineering application of the self-organizing map,” Proc. IEEE, vol. 84, pp. 1358-1383, Oct. 1996.

[7] J. Kangas, T. Kohonen, J. Laaksonen, O. Simula, and O. Venta, “Variants of the self-organizing maps,” IEEE Trans. Nerual Networks, vol. 1, pp. 93-99, March 1990.

[8] H. S. Hosseini and R. Safabakhsh, “TASOM: the time adaptive self-organizing map,” Trans. Information Technology: Coding and Computing, pp. 422-427, 2000.

[9] D. Merkl and A. Rauber, “Automatic labeling of self-organizing maps for information retrieval,” ICONIP, vol. 1, 1999.

[10] M. C. Su and H. T. Chang, “A new model of self-organizing neural networks and its application in data projection,” IEEE Trans. Nerual Networks, vol. 12, pp. 153-158, Jan. 2001

[11] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan, “Dynamic self-organizing Maps with controlled growth for knowledge discovery,” IEEE Trans. Neural Networks, vol. 11, pp. 601 614, May 2000.

[12] M. C. Su and H. T. Chang, “Fast self-organizing feature map algorithm,” IEEE Trans. Nerual Networks, vol. 11, pp. 721-733, May 2000.

[13] A. P. Azcarraga, “Assessing self-organization using Order Metrics,” IEEE Trans. Nerual Networks, vol. 6, pp. 159-164, 2000.

[14] Klaus Krippendorf, “Dictionary of Cybernetics,” unpublished report dated February 2, 1986, University of Pennsylvania.

APPENDIX A

MATERIALS AND METHODS

By Libba Lamastus

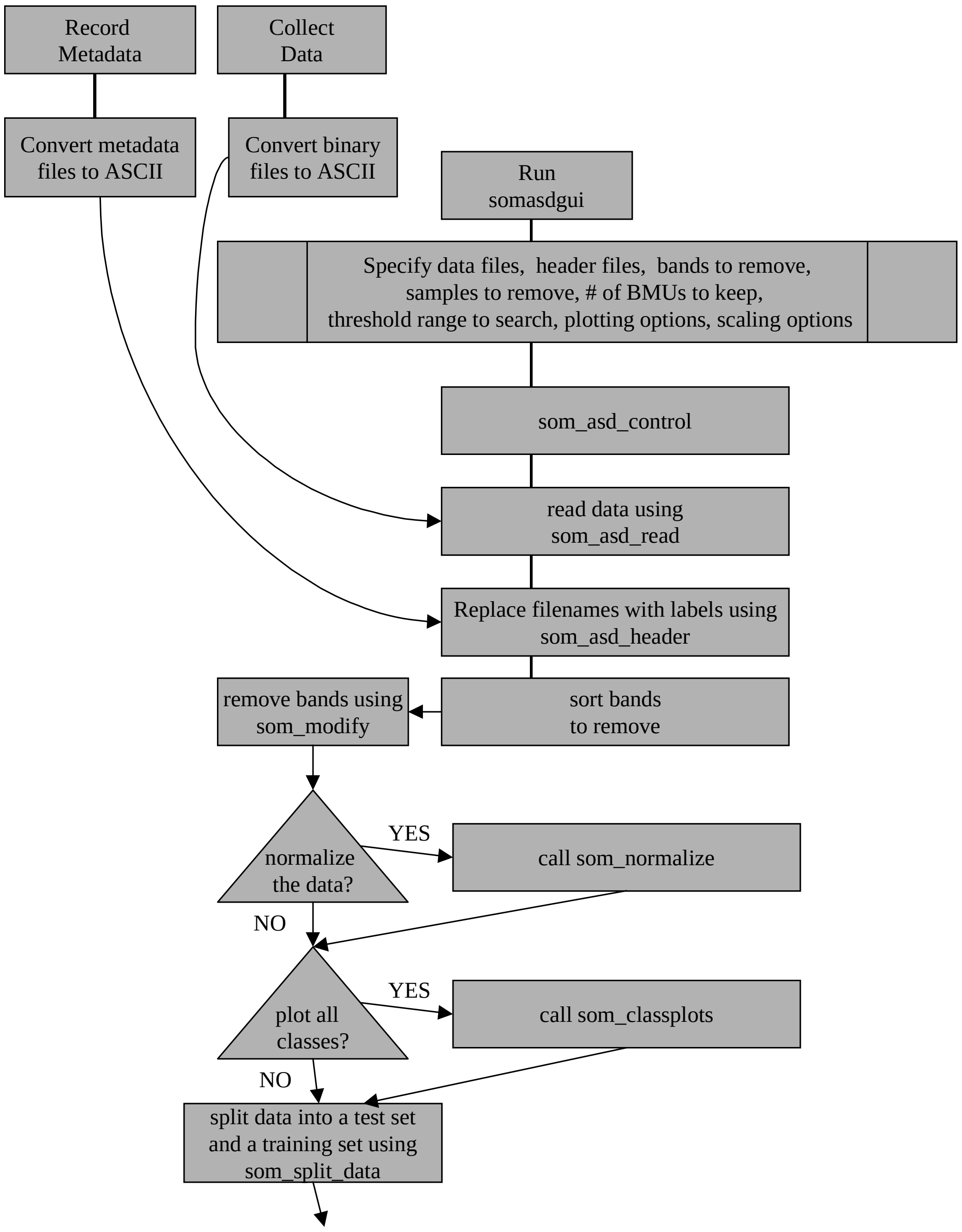
This experiment was designed to observe the spectral response of four different weed species and soybean. Mixed vegetation including experimental units (EU) of soybean with a mixed weed composition and EUs of mixed weed composition without soybean, and bare soil were also included in the study for comparison. Weed species evaluated were pitted morningglory, entireleaf morningglory, sicklepod, and common cocklebur. Experimental units were 4 x 4 m, with a 2-m alley surrounding each unit to help prevent contamination across experimental units. For ease of management, a grid was established to separate each plot into 16 1-m by 1-m units. All experimental units were hand planted on June 10, 1998, and June 1, 1999. Due to extremely dry environmental conditions in 1998, experimental units were irrigated approximately two weeks after planting, resulting in late emergence of the weeds. Experimental units were maintained to specific species by hand removal of all other vegetation on a weekly basis.

Leaf reflectance was detected using two different narrow-bandwidth visible/near-infrared spectroradiometers, an Analytical Spectral Device Full Range FieldSpec Pro® (ASD) and a Geophysical Environmental Research 1500® (GER). The ASD has a spectral range of 350 nm to 2500 nm with sampling intervals of 1.4 nm in the 350 - 100 nm range and 2 nm in the 1000 - 2500nm range. The GER had a spectral range of 350- 1050nm with a sampling interval of 1.5nm. GER spectral data were collected August 17-18, 1998, and August 5, 1999, using a 3-degree field of view (FOV) optic, at a nadir height of 12 to 18 cm from the target, depending on leaf size. ASD data were collected on August 5, and 26, 1999, using an 8-degree FOV optic at a nadir height of 10-15 cm. When possible, the uppermost leaf of the plant was used as the target for data collection. All measurements were made during highest sun angle, 10:00 a.m. - 2:00 p.m. in cloud-free conditions.   Four random observations were recorded in each experimental unit, totaling 320 observations for each data collection timing. Optimizing the instrument for the environmental conditions at the time of the measurements standardized spectral responses. This was achieved by using a calibrated, tripod-mounted, level white reflectance Spectralon® panel as a standard reference. Reflectance standard measurements were made after the completion of each experimental unit, or every four samples. An average of 6 scans for the GER and 10 scans for the ASD was used as the sample reflectance response for each observation. Data collected on August 26, 1999, were collected in the same manner as previously stated; however, due high temperatures and the sensitivity of the ASD to heat, only one sample was collected from each experimental unit.

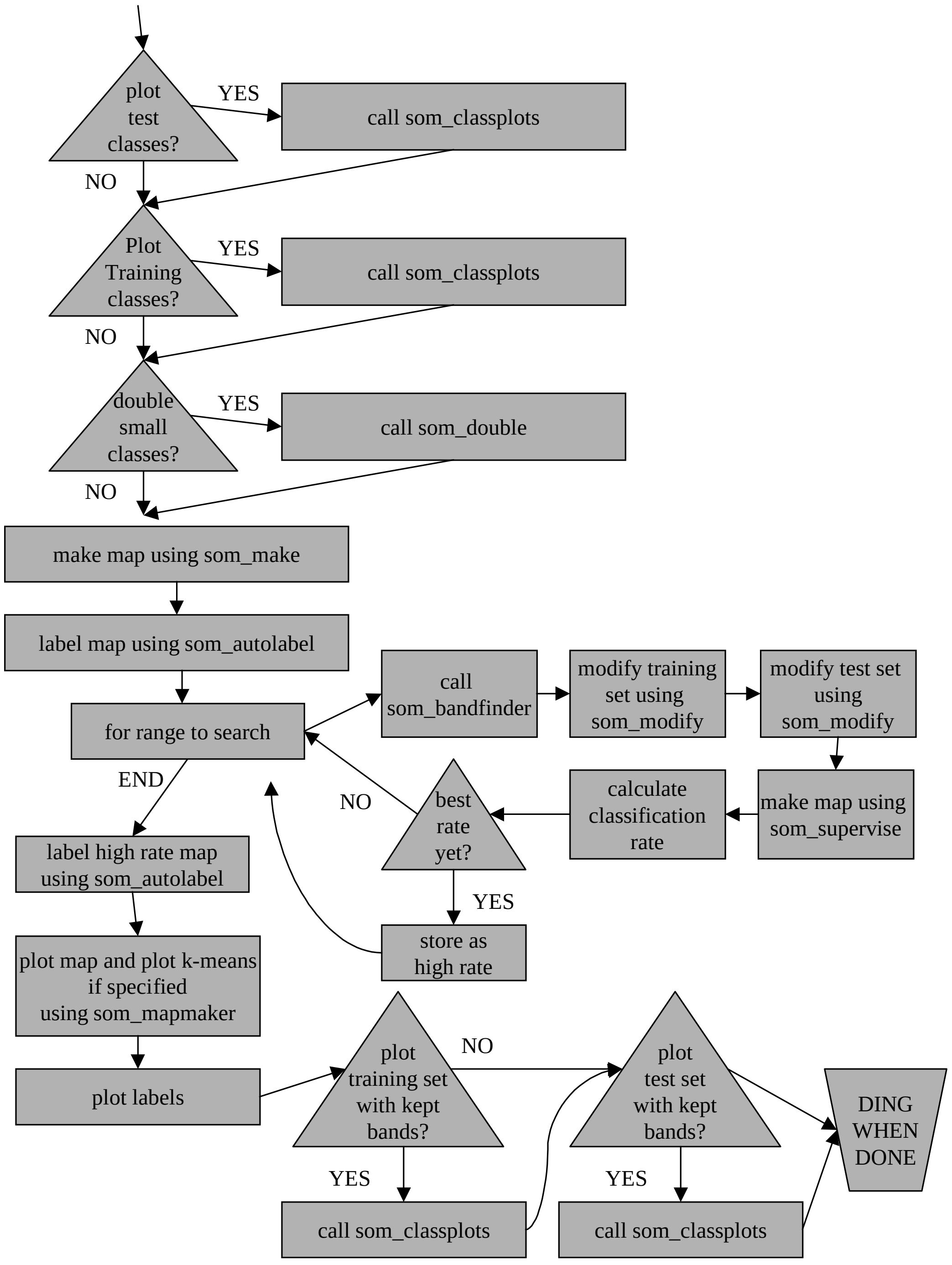
APPENDIX B

FLOW CHART AND USER MANUAL

B.1 Flow Chart of Process



CONTINUED ON NEXT PAGE


plot test classes?
YES
call som_classplots
NO
Plot Training classes?
YES
call som_classplots
NO
double small classes?
YES
call som_double
NO
make map using som_make
label map using som_autolabel
for range to search
call som_bandfinder
modify training set using som_modify
modify test set using som_modify
make map using som_supervise
calculate classification rate
best rate yet?
NO
YES
store as high rate
END
label high rate map using som_autolabel
plot map and plot k-means if specified using som_mapmaker
plot labels
plot training set with kept bands?
NO
YES
call som_classplots
plot test set with kept bands?
YES
call som_classplots
DING WHEN DONE

B.2 Process Manual

This manual is a walkthrough of how to analyze ASD data using somasdgui.

1. Collect data using the ASD while recording the class description of each target.
2. Convert files from binary to ASCII using ASD Viewspec Pro. Select all files, click process, click ASCII export, and check the box that says, "output to a single file". Then click "ok".

   Note: There is a limit of about 400 files that ASD Viewspec Pro can convert into one file. This is not a limit of somasdgui.
3. Open the ASCII file using Windows WordPad and replace "wavelength" with "#n w".
4. Create a two-column header file that contains the binary file name in the first column and the class name in the second column.

   Note: Choose class and subclass label lengths to be the same for each type of class or subclass. Consider IPOH1, CASO1, IPOH2, and CASO2 where the first four characters represent the class and the fifth character represents the subclass. This can be useful if trying to find differences between subclasses.
5. Put somtoolbox2 and asdsomtoolbox in the toolbox directory and set the path in MATLAB.
6. Move data files and header files to the bin directory.
7. Type asdsomgui from the MATLAB command window.
8. Enter the name of the data files in the first column of "none" and the corresponding header files in the second column of "none".

9. Under the label that reads “Remove Bands” is where entire ranges of bands can be removed. There is no problem if two sets of ranges overlap.
10. Beside the label that reads, “# of bands/BMU to keep”, enter the number of bands per best matching unit to keep.

    Note: At least one band per BMU must be kept. Increasing this number will increase the total number of bands kept.
11. Beside the label that reads, “Percent of samples to hold back”, enter the percentage of the total number of samples to hold back for testing purposes.
12. Under the label that reads, “Range to Search”, enter the start and end threshold percentages to search through.

    Note: This is done in one percent increments.
13. Under the label that reads, “Grid Size”, enter the size of the SOM to create.

    Note: As the total number of cells increases, the total number of bands that will be kept is increased.
14. Beside the label that reads, “Labels to remove”, enter the names of the classes that should not be used.
15. Under the label that reads, “Label length”, enter the length of the class label to be used for distinguishing between classes or subclasses. Refer to step 4.
16. On the right-hand side of the GUI screen is a list of self-explanatory options that can be selected.

17. Click “Go”.

    Note: This may take a long time. A bell will ring when done if speakers are on. Do not switch to figures until the program finishes or errors will occur in the plots.

18. The best classification rate will be displayed to the command window.

    Other information about the best classification rate is stored in variables.

a.	Bands chosen	highstrain.comp_names
b.	Indices of bands chosen	highkeptbands
c.	Threshold percentage	highpercent
d.	Classification rate	highrate
e.	Training data struct	highstrain
f.	Test data struct	highstest
g.	Map	highsmap

19. Data structs created before band selection are stored in other variables.

a.	Training set	trainset
b.	Test set	testset
c.	Map created using all bands	smap

APPENDIX C

CODE

C.1 somasdgui.m

```
clc;
clear;
load fig_test
%This file contains the callback function initializations for som_asd_gui
global h_normsamp h_plotall h_plottest h_plottrain h_smalldouble;
global h_plotstrain h_plotstest h_runkmeans h_labellegth;
global h_data1 h_data2 h_data3 h_data4 h_data5 h_data6 h_data7 h_data8 h_data9;
global h_head1 h_head2 h_head3 h_head4 h_head5 h_head6 h_head7 h_head8 h_head9;
global h_start1 h_start2 h_start3 h_start4 h_start5 h_start6 h_start7 h_start8;
global h_end1 h_end2 h_end3 h_end4 h_end5 h_end6 h_end7 h_end8 h_bmubands h_pset h_runkmeans;
global h_pstart h_pend h_y h_x h_remlabel1 h_remlabel2 h_remlabel3 h_remlabel4;



h0 = figure('Units','points', ...
        'Color',[0.8 0.8 0.8], ...
        'Colormap',mat0, ...
        'FileName','C:\MATLABR11\bin\Thesis\fig_test.m', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[279.75 146.25 420 315], ...
        'Tag','Fig2', ...
        'ToolBar','none');
h_normsamp = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[283.5 277.5 113.25 15], ...
        'String','Normalize each sample', ...
        'Style','checkbox', ...
        'Tag','Checkbox1', ...
        'Value',1);
h_plotall = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[283.5 258 115.5 15], ...
        'String','Plot all samples by class', ...
        'Style','checkbox', ...
        'Tag','Checkbox2');
h_plottest = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[283.5 219 113.25 15], ...
        'String','Plot test samples by class', ...
        'Style','checkbox', ...
        'Tag','Checkbox3');
h_plottrain = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[283.5 238.5 116.25 15], ...
        'String','Plot training samples by class', ...
        'Style','checkbox', ...
        'Tag','Checkbox4');
h_smalldouble = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[283.5 199.5 130.5 15], ...
        'String','Double small classes for training', ...
        'Style','checkbox', ...
        'Tag','Checkbox5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[37.5 288 45 15], ...
        'String','Data files', ...
        'Style','text', ...
        'Tag','StaticText1');
h_pstart = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[120 46.5 19.5 15], ...
        'String','5', ...
        'Style','edit', ...
        'Tag','EditText1', ...
        'UserData','[ ]');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','som_asd_control', ...
        'FontSize',24, ...
        'ListboxTop',0, ...
        'Position',[311.25 37.5 66 57.75], ...
        'String','Go', ...
        'Tag','Pushbutton1', ...
        'UserData','[ ]');
h_data1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[23.25 265.5 73.5 15], ...
        'String','none', ...
        'Style','edit', ...
        'Tag','EditText2', ...
        'TooltipString','Enter a data file');
h_data2 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[23.25 247.40625 73.5 15], ...
        'String','none', ...
        'Style','edit', ...
        'Tag','EditText2', ...
        'TooltipString','Enter a data file');
h_data3 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[23.25 229.3125 73.5 15], ...
        'String','none', ...
        'Style','edit', ...
        'Tag','EditText2', ...
        'TooltipString','Enter a data file');
h_data4 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
```

```
		'Callback','', ...
		'ListboxTop',0, ...
		'Position',[23.25 211.21875 73.5 15], ...
		'String','none', ...
		'Style','edit', ...
		'Tag','EditText2', ...
		'TooltipString','Enter a data file');
h_data5 = uicontrol('Parent',h0, ...
		'Units','points', ...
		'BackgroundColor',[1 1 1], ...
		'Callback','', ...
		'ListboxTop',0, ...
		'Position',[23.25 193.125 73.5 15], ...
		'String','none', ...
		'Style','edit', ...
		'Tag','EditText2', ...
		'TooltipString','Enter a data file');
h_data6 = uicontrol('Parent',h0, ...
		'Units','points', ...
		'BackgroundColor',[1 1 1], ...
		'Callback','', ...
		'ListboxTop',0, ...
		'Position',[23.25 175.03125 73.5 15], ...
		'String','none', ...
		'Style','edit', ...
		'Tag','EditText2', ...
		'TooltipString','Enter a data file');
h_data7 = uicontrol('Parent',h0, ...
		'Units','points', ...
		'BackgroundColor',[1 1 1], ...
		'Callback','', ...
		'ListboxTop',0, ...
		'Position',[23.25 156.9375 73.5 15], ...
		'String','none', ...
		'Style','edit', ...
		'Tag','EditText2', ...
		'TooltipString','Enter a data file');
h_data8 = uicontrol('Parent',h0, ...
		'Units','points', ...
		'BackgroundColor',[1 1 1], ...
		'Callback','', ...
		'ListboxTop',0, ...
		'Position',[23.25 138.84375 73.5 15], ...
		'String','none', ...
		'Style','edit', ...
		'Tag','EditText2', ...
		'TooltipString','Enter a data file');
h_data9 = uicontrol('Parent',h0, ...
		'Units','points', ...
		'BackgroundColor',[1 1 1], ...
		'Callback','', ...
		'ListboxTop',0, ...
		'Position',[23.25 120.75 73.5 15], ...
		'String','none', ...
		'Style','edit', ...
		'Tag','EditText2', ...
		'TooltipString','Enter a data file');
h1 = uicontrol('Parent',h0, ...
		'Units','points', ...
		'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
		'ListboxTop',0, ...
		'Position',[123.75 289.5 45 15], ...
		'String','Header files', ...
		'Style','text', ...
		'Tag','StaticText2');
h_head9 = uicontrol('Parent',h0, ...
		'Units','points', ...
```

```
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[110.25 120.75 73.5 15], ...
                'String','none', ...
                'Style','edit', ...
                'Tag','EditText2', ...
                'TooltipString','Enter a data file');
h_head8 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[110.25 139.5 73.5 15], ...
                'String','none', ...
                'Style','edit', ...
                'Tag','EditText2', ...
                'TooltipString','Enter a data file');
h_head7 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[110.25 157.5 73.5 15], ...
                'String','none', ...
                'Style','edit', ...
                'Tag','EditText2', ...
                'TooltipString','Enter a data file');
h_head6 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[110.25 174.75 73.5 15], ...
                'String','none', ...
                'Style','edit', ...
                'Tag','EditText2', ...
                'TooltipString','Enter a data file');
h_head5 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[110.25 192 73.5 15], ...
                'String','none', ...
                'Style','edit', ...
                'Tag','EditText2', ...
                'TooltipString','Enter a data file');
h_head4 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[110.25 210 73.5 15], ...
                'String','none', ...
                'Style','edit', ...
                'Tag','EditText2', ...
                'TooltipString','Enter a data file');
h_head3 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[110.25 228 73.5 15], ...
                'String','none', ...
                'Style','edit', ...
                'Tag','EditText2', ...
```

```
            'TooltipString','Enter a data file');
h_head2 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[1 1 1], ...
            'Callback','', ...
            'ListboxTop',0, ...
            'Position',[110.25 246.75 73.5 15], ...
            'String','none', ...
            'Style','edit', ...
            'Tag','EditText2', ...
            'TooltipString','Enter a data file');
h_head1 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[1 1 1], ...
            'Callback','', ...
            'ListboxTop',0, ...
            'Position',[110.25 265.5 73.5 15], ...
            'String','none', ...
            'Style','edit', ...
            'Tag','EditText2', ...
            'TooltipString','Enter a data file');
h1 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
            'ListboxTop',0, ...
            'Position',[201.75 290.25 45 20.25], ...
            'String','Remove Bands', ...
            'Style','text', ...
            'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
            'ListboxTop',0, ...
            'Position',[192 275.25 23.25 15], ...
            'String','Start', ...
            'Style','text', ...
            'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
            'ListboxTop',0, ...
            'Position',[228.75 275.25 20.25 15], ...
            'String','End', ...
            'Style','text', ...
            'Tag','StaticText4');
h_start1 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[1 1 1], ...
            'Callback','', ...
            'ListboxTop',0, ...
            'Position',[187.5 259.1250000000001 27.75 15], ...
            'String','950', ...
            'Style','edit', ...
            'Tag','EditText3');
h_end1 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[1 1 1], ...
            'Callback','', ...
            'ListboxTop',0, ...
            'Position',[226.5 259.1250000000001 27.75 15], ...
            'String','2500', ...
            'Style','edit', ...
            'Tag','EditText3');
h_end2 = uicontrol('Parent',h0, ...
            'Units','points', ...
            'BackgroundColor',[1 1 1], ...
            'Callback','', ...
```

```
                'ListboxTop',0, ...
                'Position',[226.5 240.1607142857143 27.75 15], ...
                'String','360', ...
                'Style','edit', ...
                'Tag','EditText3');
h_start2 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[187.5 240.1607142857143 27.75 15], ...
                'String','350', ...
                'Style','edit', ...
                'Tag','EditText3');
h_end3 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[226.5 221.1964285714286 27.75 15], ...
                'String','2500', ...
                'Style','edit', ...
                'Tag','EditText3');
h_start3 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[187.5 221.1964285714286 27.75 15], ...
                'String','1763', ...
                'Style','edit', ...
                'Tag','EditText3');
h_end4 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[226.5 202.2321428571429 27.75 15], ...
                'String','1500', ...
                'Style','edit', ...
                'Tag','EditText3');
h_start4 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[187.5 202.2321428571429 27.75 15], ...
                'String','1285', ...
                'Style','edit', ...
                'Tag','EditText3');
h_end5 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[226.5 183.2678571428572 27.75 15], ...
                'String','972', ...
                'Style','edit', ...
                'Tag','EditText3');
h_start5 = uicontrol('Parent',h0, ...
                'Units','points', ...
                'BackgroundColor',[1 1 1], ...
                'Callback','', ...
                'ListboxTop',0, ...
                'Position',[187.5 183.2678571428572 27.75 15], ...
                'String','967', ...
                'Style','edit', ...
```

```
        'Tag','EditText3');
h_end6 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',mat1, ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText3');
h_start6 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',mat2, ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText3');
h_end7 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[226.5 145.3392857142857 27.75 15], ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText3');
h_start7 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[187.5 145.3392857142857 27.75 15], ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText3');
h_end8 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[226.5 126.375 27.75 15], ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText3');
h_start8 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[187.5 126.375 27.75 15], ...
        'String','0', ...
        'Style','edit', ...
        'Tag','EditText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[20.625 78.75 45 30], ...
        'String','# of bands/BMU to keep ', ...
        'Style','text', ...
        'Tag','StaticText5');
h_bmubands = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[69 86.25 27.75 15], ...
        'String','1', ...
        'Style','edit', ...
        'Tag','EditText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[20.625 44.25 45 30.75], ...
        'String','Percent of samples to hold back', ...
        'Style','text', ...
        'Tag','StaticText6');
h_pset = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[70.5 52.125 22.5 15], ...
        'String','15', ...
        'Style','edit', ...
        'Tag','EditText5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[127.5 83.25 45 21], ...
        'String','Range to Search', ...
        'Style','text', ...
        'Tag','StaticText7');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[155.25 64.5 20.25 15], ...
        'String','End', ...
        'Style','text', ...
        'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[118.5 64.5 23.25 15], ...
        'String','Start', ...
        'Style','text', ...
        'Tag','StaticText4');
h_pend = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[156 46.5 19.5 15], ...
        'String','20', ...
        'Style','edit', ...
        'Tag','EditText1', ...
        'UserData','[ ]');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[275.25 294.75 105.75 15], ...
        'String','Pre band selection options', ...
        'Style','text', ...
        'Tag','StaticText8');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[275.25 148.5 108 15], ...
        'String','Post band selection options', ...
        'Style','text', ...
        'Tag','StaticText9');
h_plotstrain = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[283.5 130.5 116.25 15], ...
        'String','Plot training samples by class', ...
        'Style','checkbox', ...
        'Tag','Checkbox4');
h_plotstest = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[283.5 108.75 113.25 15], ...
        'String','Plot test samples by class', ...
        'Style','checkbox', ...
        'Tag','Checkbox3');
h_y = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[221.25 48 19.5 15], ...
        'String','8', ...
        'Style','edit', ...
        'Tag','EditText1', ...
        'UserData','[ ]');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[195 66.75 17.25 15], ...
        'String','X', ...
        'Style','text', ...
        'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[220.5 66.75 20.25 15], ...
        'String','Y', ...
        'Style','text', ...
        'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[195.75 83.25 45 21], ...
        'String','Grid Size', ...
        'Style','text', ...
        'Tag','StaticText7');
h_x = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Callback','', ...
        'ListboxTop',0, ...
        'Position',[193.5 48 19.5 15], ...
        'String','8', ...
        'Style','edit', ...
        'Tag','EditText1', ...
        'UserData','[ ]');
```

```
h1 = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
	'ListboxTop',0, ...
	'Position',[20.625 17.25 45 19.5], ...
	'String','Labels to remove', ...
	'Style','text', ...
	'Tag','StaticText10');
h_remlabel1 = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[1 1 1], ...
	'Callback','', ...
	'ListboxTop',0, ...
	'Position',[78.75 19.5 45 15], ...
	'String','ref', ...
	'Style','edit', ...
	'Tag','EditText6');
h_remlabel2 = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[1 1 1], ...
	'Callback','', ...
	'ListboxTop',0, ...
	'Position',[129.75 19.5 45 15], ...
	'String','wsoy', ...
	'Style','edit', ...
	'Tag','EditText6');
h_remlabel3 = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[1 1 1], ...
	'Callback','', ...
	'ListboxTop',0, ...
	'Position',[180.75 19.5 45 15], ...
	'String','bad', ...
	'Style','edit', ...
	'Tag','EditText6');
h_remlabel4 = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[1 1 1], ...
	'Callback','', ...
	'ListboxTop',0, ...
	'Position',[231.75 19.5 45 15], ...
	'String','weed', ...
	'Style','edit', ...
	'Tag','EditText6');
h_runkmeans = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
	'Callback','', ...
	'ListboxTop',0, ...
	'Position',[283.5 179.25 130.5 15], ...
	'String','Enable K-means', ...
	'Style','checkbox', ...
	'Tag','Checkbox5');
h1 = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
	'ListboxTop',0, ...
	'Position',[248.25 84 39 21], ...
	'String','Label length', ...
	'Style','text', ...
	'Tag','StaticText11');
h_labellength = uicontrol('Parent',h0, ...
	'Units','points', ...
	'BackgroundColor',[1 1 1], ...
	'Callback','', ...
	'FontSize',10, ...
	'ListboxTop',0, ...
```

```
        'Position',[255.75 61.5 25.5 18], ...
        'String','4', ...
        'Style','edit', ...
        'Tag','EditText7');
```

C.2 som_asd_gui.m

```
%This program reads ASD data
%clear;
%clc;
%remstart(1)=950;          %used to set the start of the bands to be removed before calculations
%remend(1)=2500;           %used to set the end of the bands to be removed before calculations
%remstart(2)=1;
%remend(2)=10
%rem_ind=0;                %stores the total array of bands to be removed
%normsamp=1;                          %should the samples be normalized 0 => no (default) 1 => yes
%plotall=0;                %plot all samples by class
%plottest=0;               %plot test samples by class
%plottrain=0;      %plot training samples by class
%smalldouble=1; %double small sets
%bmubands=2;                          %number of bands to keep
%pstart=5;         %start of range of percentages to search
%pend=10;                             %end of range of percentages to search
%plotstrain=0;             %plot results of training set by class
%plotstest=1;              %plot results of test set by class
%pset=15;          %percent to hold back for testing
%X=8;                                 %x size of grid
%Y=8;                                 %y size of grid
%labellength=4;    %length of label to keep
%runkmeans=0;      %run kmeans section in mapmaker yew or no

%This section converts GUI values to local values
normsamp=get(h_normsamp,'value');
plotall=get(h_plotall,'value');
plottest=get(h_plottest,'value');
plottrain=get(h_plottrain,'value');
smalldouble=get(h_smalldouble,'value');
pstart=str2num(get(h_pstart,'string'));
pend=str2num(get(h_pend,'string'));
plotstrain=get(h_plotstrain,'value');
plotstest=get(h_plotstest,'value');
runkmeans=get(h_runkmeans,'value');
labellength=str2num(get(h_labellength,'string'));

data{1}=get(h_data1,'string');
data{2}=get(h_data2,'string');
data{3}=get(h_data3,'string');
data{4}=get(h_data4,'string');
data{5}=get(h_data5,'string');
data{6}=get(h_data6,'string');
data{7}=get(h_data7,'string');
data{8}=get(h_data8,'string');
data{9}=get(h_data9,'string');
head{1}=get(h_head1,'string');
head{2}=get(h_head2,'string');
head{3}=get(h_head3,'string');
head{4}=get(h_head4,'string');
head{5}=get(h_head5,'string');
head{6}=get(h_head6,'string');
head{7}=get(h_head7,'string');
head{8}=get(h_head8,'string');
head{9}=get(h_head9,'string');
remstart(1)=str2num(get(h_start1,'string'));
remstart(2)=str2num(get(h_start2,'string'));
remstart(3)=str2num(get(h_start3,'string'));
remstart(4)=str2num(get(h_start4,'string'));
remstart(5)=str2num(get(h_start5,'string'));
remstart(6)=str2num(get(h_start6,'string'));
remstart(7)=str2num(get(h_start7,'string'));
remstart(8)=str2num(get(h_start8,'string'));
remend(1)=str2num(get(h_end1,'string'));
remend(2)=str2num(get(h_end2,'string'));
remend(3)=str2num(get(h_end3,'string'));
remend(4)=str2num(get(h_end4,'string'));
remend(5)=str2num(get(h_end5,'string'));
remend(6)=str2num(get(h_end6,'string'));
remend(7)=str2num(get(h_end7,'string'));
remend(8)=str2num(get(h_end8,'string'));
bmubands=str2num(get(h_bmubands,'string'));
pset=str2num(get(h_pset,'string'));


Y=str2num(get(h_y,'string'));
X=str2num(get(h_x,'string'));
remlabel1=get(h_remlabel1,'string');
remlabel2=get(h_remlabel2,'string');
remlabel3=get(h_remlabel3,'string');
remlabel4=get(h_remlabel4,'string');

%read the data
sintake=asd_read_data(data{1});
sintake=asd_read_header(sintake,head{1},'len',labellength,remlabel1,remlabel2,remlabel3,remlabel4);
%determine how many data files to use
for m=2:9
  temptest=data{m};
  if temptest(1:4)~='none'
            aintake=asd_read_data(data{m});
                        aintake=asd_read_header(aintake,head{m},'len',labellength,remlabel1,remlabel2,remlabel3,remlabel4);
                        sintake = som_modify_dataset(sintake, 'addsamp', aintake);
                        [sintake.labels b]=sort(sintake.labels);
    sintake.data=sintake.data(b,:);
  end
end

%This can be used to aid in data reduction by eliminating entire blocks
%sintake=som_modify_dataset(sintake,'removecomp',(1763-350):length(sintake.comp_names));
%sintake=som_modify_dataset(sintake,'removecomp',(1285-350):(1500-350));
%sintake=som_modify_dataset(sintake,'removecomp',(967-350):(972-350));
%sintake=som_modify_dataset(sintake,'removecomp',(950-350):length(sintake.comp_names));
%sintake=som_modify_dataset(sintake,'removecomp',1:10);

%This section determines exactly which band numbers are to be removed
temp=350;
step=0;
for m=1:length(remstart)
    if remstart(m)>349
            for n=1:remend(m)-(remstart(m)-1)
            temp(n+step)=remstart(m)+n-1;
     end
    end
    step=length(temp);
end
temp=sort(temp);
rem_ind(1)=temp(1);
count=1;
for m=2:length(temp);
  if rem_ind(count)<temp(m)
    count=count+1;
    rem_ind(count)=temp(m);
  end
end
%Remove undesired bands
sintake=som_modify_dataset(sintake,'removecomp',rem_ind-349);
```

```
if normsamp==1;
            %Normalize the data
            %Sets the range of all samples as [0 1]
  sintake=som_asd_norm(sintake);
end

%This section maps the individual classes to determine if there are any extreme outliers
%%%%%%REMOVED BECAUSE OF COMPLICATIONS AND REPLACED WITH som_classplots
%%%%%%%%%      %som_map_classes(sintake);


if plotall==1
  %This creates plots of classes, usefull in locating misslabeled data and outliers
            som_classplots(sintake);
end

%This section creates two data sets, one for training and one for testing
[trainset testset]=som_split_data(sintake,pset*.01);

if plottest==1
  %plot testset by class
  som_classplots(testset);
end

if plottrain==1
  %plot training set by class;
  som_classplots(trainset);
end

if smalldouble==1
            %This doubles the importance of classes that have less than half the number of sample of the largest class
            %for training purposes only
   trainset=som_double(trainset);
end

%make original map
smap=som_make(trainset,'rect','init','lininit','msize',[Y X]);
smap = som_autolabel(smap,trainset,'vote');
%som_mapmaker(smap,trainset,runkmeans);

%%%%REMOVED AND REPLACED BECAUSE OF GUI INTERFACE
%find some relivant bands
%[intake1,smap1,tb{1},tbsize1]=som_newmap(smap,trainset,1,.001);
%[intake2,smap2,tb{2},tbsize2]=som_newmap(smap,trainset,1,.01);
%[intake3,smap3,tb{1},tbsize3]=som_newmap(smap,trainset,1,.09);
%[intake4,smap4,tb{2},tbsize4]=som_newmap(smap,trainset,1,.15);
%[intake5,smap5,tb{3},tbsize5]=som_newmap(smap,trainset,1,.2);
%[intake6,smap6,tb{4},tbsize6]=som_newmap(smap,trainset,1,.05);

%%%%%%GUI METHOD OF FINDING RELIVANT BANDS
%%%%%%%%%%%%%%%evaluate data
highrate=0;             %initialize high classification rate
for percent=pstart:pend
            %Call som_bandfinder using preset parameters and incremental parameter percent
   [keptbands,tbsize]=som_bandfinder(smap,trainset,bmubands,.01*percent);

            %This section creates a map using all "keptbands"
            strain = som_modify_dataset(trainset, 'extractcomp', keptbands);
            stest=som_modify_dataset(testset,'extractcomp',keptbands);
     %make comparison map
            smap2 = som_supervised(strain,'rect','msize',[Y X]);%'init','lininit',

            %calculate classification rate
            templabels=smap2.labels;
            tempmap = som_autolabel(smap2,stest,'add');
            [rows cols]=size(tempmap.labels);
            correct=0;
            for n=2:cols
              for m=1:rows
              if isempty(templabels{m}), % ignore empties
            elseif strcmp(templabels{m},tempmap.labels{m,n}),
                                                correct=correct+1;
            end
                        end
            end
            total=length(testset.labels);
  rate=correct/total;

  %keep track of best percent
  if rate>highrate
    highkeptbands=keptbands;
    highsmap=smap2;
    highrate=rate;
    highstrain=strain;
    highstest=stest;
    highpercent=percent;
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%end of find best
percent%%%%%%%%%%%%%%%%%%%%%%%%%
highrate    %Display highest classification rate

%Display labeled map
smap2 = som_autolabel(highsmap,highstest,'freq');

som_mapmaker(smap2,highstest,runkmeans);
figure;
som_show(smap2,'empty','Labels');
som_show_add('label',smap2,'Textsize',8,'TextColor','r');

if plotstrain==1
   %plot post band selection traing sets
  som_classplots(highstrain);
end

if plotstest==1
   %plot post band selection test sets
            som_classplots(highstest);
end

%Ding when done
y=wavread('Ding.wav');
soundsc(y);
```

C.3 asd_read_data.m

```
function sintake = asd_read_data(filename)
%This program reads properly formated ASD data

%This section calls som_read_data and manipulates the data to be in the correct format
dummy=som_read_data(filename);
sintake.data=dummy.data(:,2:end)';
sintake.labels=dummy.comp_names(2:end);
for m=1:length(sintake.data)
            sintake.comp_names{m,1}=dummy.data(m,1);
end
sintake.comp_norm=cell(length(sintake.data),1);
sintake.type=dummy.type;
sintake.label_names=sintake.labels;
sintake.name=dummy.name;
```

C.4 asd_read_header.m

```
function sintake = asd_read_header(sintake, headerfile, varargin)
%This program reads the header file created for asd data and replaces the filenames stored in
%sintake.labels with the new labels from the header file the data is sorted by label name when returned
%optional arguments are to remove white reference files and length of data label
%   'remstring' (scalar) any string other than "len" and files with labels that begin with "varargin" will %be removed
%  'len'       (scalar) any value other than zero and the label will be read to the specified number of letters
% <<< Do Not enter a number longer than the shortest label>>>
% defaults
len=0;                                  %if no length for label read is given, the default is to read the entire label
remref=0;                               %if not specified all labels begining with "ref" will be removed

% varargin
i=1;
m=0;
while i<=length(varargin),
  argok = 1;
  if ischar(varargin{i}),
    switch varargin{i},
      % argument IDs
     case 'len',       i=i+1; len = varargin{i};
   otherwise argok=0;
                 m=m+1;
        remstring{m}=varargin{i};
     end
  end
  %if ~argok,
  %  disp(['(som_make) Ignoring invalid argument #' num2str(i+1)]);
  %end
  i = i+1;
end



%This section replaces the individual file names of ASD data with their labels from a header file
[filenames labels] = textread(headerfile,'%s %s');
n=length(filenames);
next=1;
for m=1:length(sintake.labels);
   count=1;
   while count<=n
      if sintake.labels{m}==filenames{count};
         a=labels{count};
         if len==0
            sintake.labels{m}=a(1,1:end);
         else
               sintake.labels{m}=a(1,1:len); %This can be adjusted to select labels length
         end
         count=n;
      end
      count=count+1;
   end
   %This part keeps track of files to remove
   for remsize=1:length(remstring)
               if a(1,1:length(remstring{remsize}))==remstring{remsize}
                  ind(next)=m;
               next=next+1;
      end
   end
end

%This section removes bad files
sintake=som_modify_dataset(sintake,'removesamp',ind);

%This section organizes the data
```

```
[sintake.labels b]=sort(sintake.labels);
sintake.data=sintake.data(b,:);
```

C.5 som_asd_norm.m

```
function sintake=som_asd_norm(sintake)
%this function set the range from 1 to 0 for asd data
[r c]=size(sintake.data);
            for a=1:r
             sintake.data(a,:)=sintake.data(a,:)-min(sintake.data(a,:));
             sintake.data(a,:)=sintake.data(a,:)/max(sintake.data(a,:));
  end
```

C.6 som_classplots.m

```
function tsize=som_classplots(sintake)
%This function plots all samples of each class seperately on each graph

%find the size and location of each class
tsize=som_sizefinder(sintake);

%initialize counter
start=0;

%create x axis for plots
for k=1:length(sintake.comp_names)
            x(k)=sintake.comp_names{k};
end

%create plots
for m=1:length(tsize)
  figure;
  hold on;
  title(sintake.labels(start+1));
  for n=start+1:start+tsize(m)
            plot(x,sintake.data(n,:),'x');
            end
  hold off;
  start=start+tsize(m);
end
```

C.7 som_sizefinder.m

```
function tsize=som_sizefinder(sintake)
%This function finds the locations and sizes of a sorted list of the different classes
tcount=0;
newtype=1;
for n=1:length(sintake.labels),
  if n==newtype,
    tcount=tcount+1;
    tsize(tcount)=0;
    for m=n:length(sintake.labels),
      if sintake.labels{n}==sintake.labels{m};
        tsize(tcount)=tsize(tcount)+1;
                                    end
    end
    newtype=n+tsize(tcount);
  end
end
```

C.8 som_double.m

```
function train=som_double(train)
%This program doubles the number of samples of classes that have less than half the
%number of samples of the largest class

%set the size for the training set
[trsize]=som_sizefinder(train);

%This section increases the importance of smaller samples
for m=1:length(trsize)
  if trsize(m)<(max(trsize)/2)
    start=0;
    start=sum(trsize(1:m-1));
    temp=som_modify_dataset(train,'extractsamp',start+1:start+trsize(m));
    train = som_modify_dataset(train,'addsamp',temp);
    [train.labels b]=sort(train.labels);
                    train.data=train.data(b,:);
    trsize(m)=2*trsize(m);
  end
end
```

C.9 som_bandfinder.m

```
function [Tb,tbsize]=som_bandfinder(smap,intake,bands,Tpercent)
%This function attempts to find the relevant inputs for sorting an som

%this section computes the Euclidean distance of inputs to there BMU
[Bmus Qerrors]=som_bmus(smap,intake);
eucdist=zeros(size(smap.codebook));
for m=1:length(Bmus)
  dist=smap.codebook(Bmus(m),:)-intake.data(m,:);
          dist=dist.^2;
  eucdist(Bmus(m),:)=dist+eucdist(Bmus(m),:);
end
  eucdist=eucdist.^(.5);


%This section computes the threshold
range=max(eucdist')-min(eucdist');
thershold=(Tpercent*range+min(eucdist'))';

%This section finds the (bands) number of important inputs for map units that have labels
for m=1:length(thershold)
  [y x]=sort(eucdist(m,:));
  found=0;
  n=0;
  while ~found
    n=n+1;
    if thershold(m)<=y(n)
      for k=1:bands
        if n-1+k<=length(x)
          bandselected(m,k)=x(n-1+k);
        end
      end
      found=1;
    end
  end
end

%This section elminates duplicate bands
[l w]=size(bandselected);
for m=1:l;
  for n=1:w
    pos=(m-1)*w+n;
    tb(pos)=bandselected(m,n);
  end
end
tb=sort(tb);
%ensure that 0 is not selected
for m=1:length(tb)
  if tb(m)==0
    tb(m)=1;
  end
end

tbcount=0;
newtb=1;
for n=1:length(tb),
  if n==newtb,
    tbcount=tbcount+1;
    tbsize(tbcount)=0;
    for m=n:length(tb),
      if tb(n)==tb(m);
        tbsize(tbcount)=tbsize(tbcount)+1;
      end
            Tb(tbcount)=tb(n);
    end
    newtb=n+tbsize(tbcount);
  end
end
```

C.10 som_colorhits.m

```
function tsize=som_colorhits(smap,intake);
%this program determines how many different classes there are and colors the U-matrix accordingly
tsize=som_sizefinder(intake);

m=0;
a=hsv(length(tsize));
for n=1:length(tsize),
  h = som_hits(smap,intake.data(m+1:m+tsize(n),:));
  newtype(n)=m+1;
  m=m+tsize(n);
  b(n)=som_show_add('hit',h,'MarkerColor',a(n,:),'Subplot',1);
end

legend([b],intake.labels{newtype},-1);
```

C.11 makehist.m

```
function keptbands=makehist(bandlist,threshold);
%this function makes a list of bands that were repeted atleast "threshold" times
n=hist(bandlist,max(bandlist));
step=1;
for x=1:length(n)
  if n(x)>=threshold
    keptbands(step)=x;
            step=step+1;
  end
end
hist(keptbands,max(keptbands));
```

C.12 som_split_data.m

```
function [trainset,testset]=som_split_data(sintake,percent)
%This function splits a som data struct into two different sets.  A percent of each class will be split.
```

```
%   sintake    (struct) data struct
%              percent                    (scalar)

%This section calculates the location of the samples to extract

[sintake.labels b]=sort(sintake.labels);
sintake.data=sintake.data(b,:);

[tsize]=som_sizefinder(sintake);
start=0;
testcount=0;
for m=1:length(tsize)
   ksize=floor(percent*tsize(m));
   testind(testcount+1:testcount+ksize)=round((tsize(m)-1)*rand(ksize,1))+1+start;
   testcount=ksize+testcount;
   start=start+tsize(m);
end

%This seperates the data into 2 sets, a training set and a test set
testind=sort(testind);
testset=som_modify_dataset(sintake,'extractsamp',testind);
trainset=som_modify_dataset(sintake,'removesamp',testind);
```

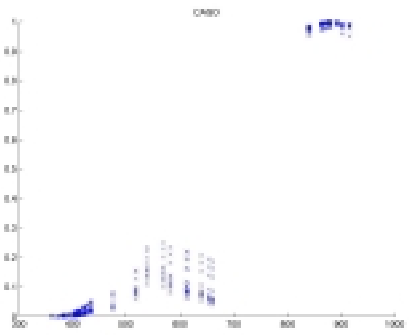
C.13 som_mapmaker.m

```
function tsize=som_mapmaker(smap,intake,runkmeans);
%This function creates color maps using som_colorhits, som_kmeanscolor, som_show, and som_show_add
figure;
colormap(1-gray);
som_show(smap,'umat','all','empty','Labels');
som_show_add('label',smap,'Textsize',8,'TextColor','r','Subplot',2);
tsize=som_colorhits(smap,intake);
if runkmeans
		figure;
		[color,b]=som_kmeanscolor(smap,8);
		som_show(smap,'color',color,'color',{color(:,:,b),'"Best clustering"'});
end
```

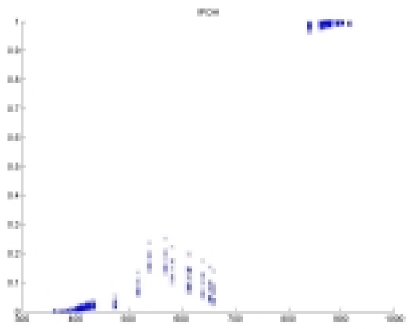
APPENDIX D

RESULTS GRAPHS

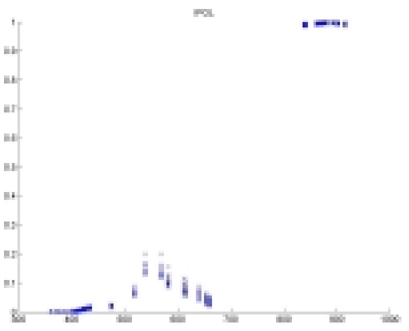
D.1 Trial 1



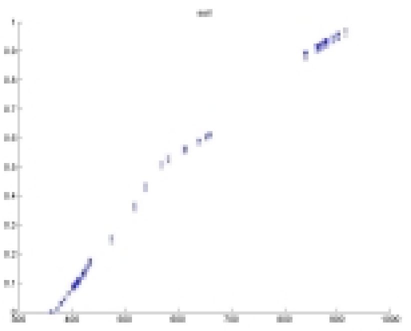
CASO



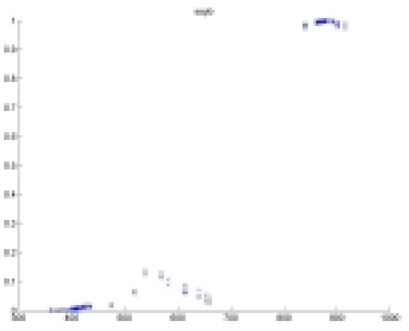
IPOH



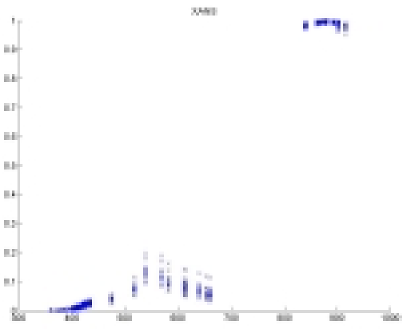
IPOL



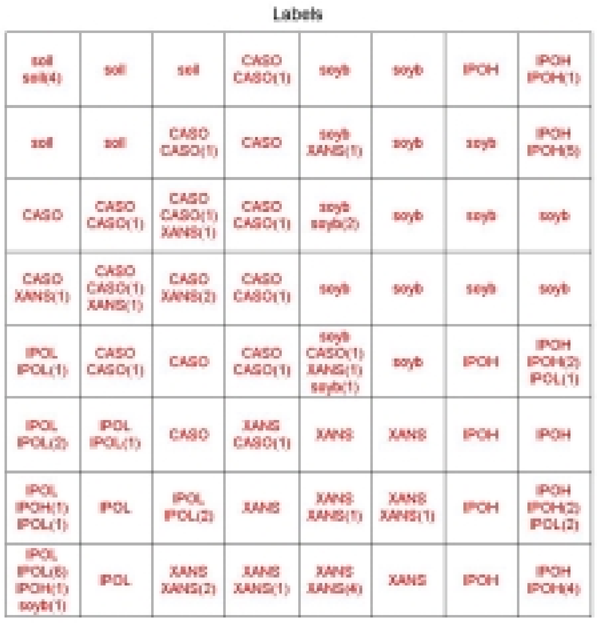
SOIL



SOYB

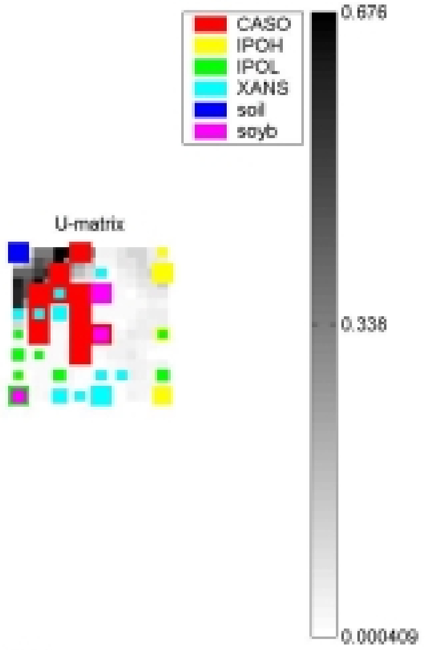


XANS



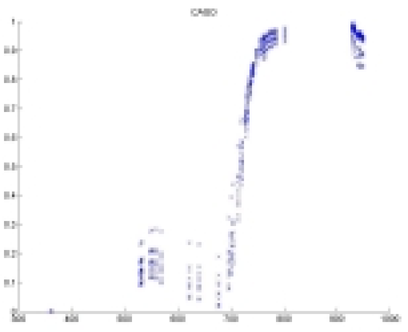
soil soil(4)	soil	soil	CASO CASO(1)	soyb	soyb	IPOH	IPOH IPOH(1)
soil	soil	CASO CASO(1)	CASO	soyb XANS(1)	soyb	soyb	IPOH IPOH(5)
CASO	CASO CASO(1)	CASO CASO(1) XANS(1)	CASO CASO(1)	soyb soyb(2)	soyb	soyb	soyb
CASO XANS(1)	CASO CASO(1) XANS(1)	CASO XANS(2)	CASO CASO(1)	soyb	soyb	soyb	soyb
IPOL IPOL(1)	CASO CASO(1)	CASO	CASO CASO(1)	soyb CASO(1) XANS(1) soyb(1)	soyb	IPOH	IPOH IPOH(2) IPOL(1)
IPOL IPOL(2)	IPOL IPOL(1)	CASO	XANS CASO(1)	XANS	XANS	IPOH	IPOH
IPOL IPOH(1) IPOL(1)	IPOL	IPOL IPOL(2)	XANS	XANS XANS(1)	XANS XANS(1)	IPOH	IPOH IPOH(2) IPOL(2)
IPOL IPOL(6) IPOH(1) soyb(1)	IPOL	XANS XANS(2)	XANS XANS(1)	XANS XANS(4)	XANS	IPOH	IPOH IPOH(4)

072500$_{f}$ull.txt

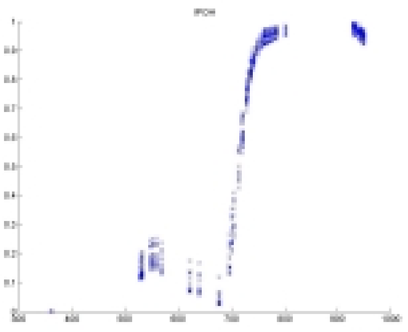


072500$_{f}$ull.txt

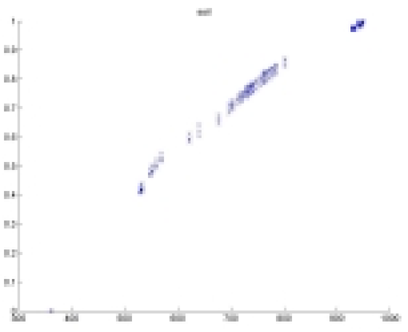
D.2 Trial 2



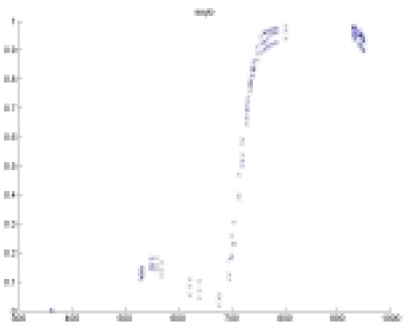
CASO



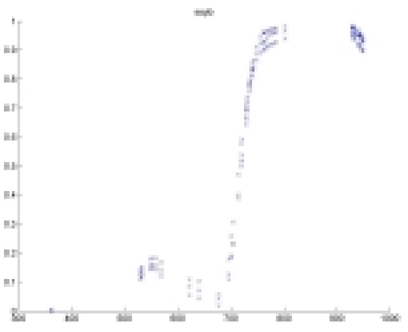
IPOH



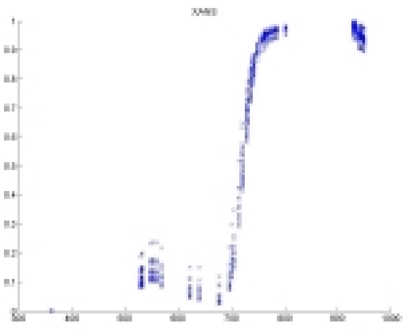
IPOL



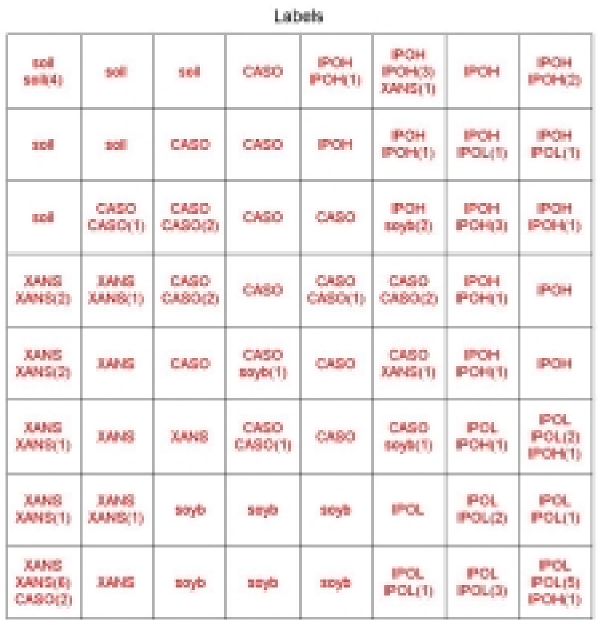
SOIL



SOYB

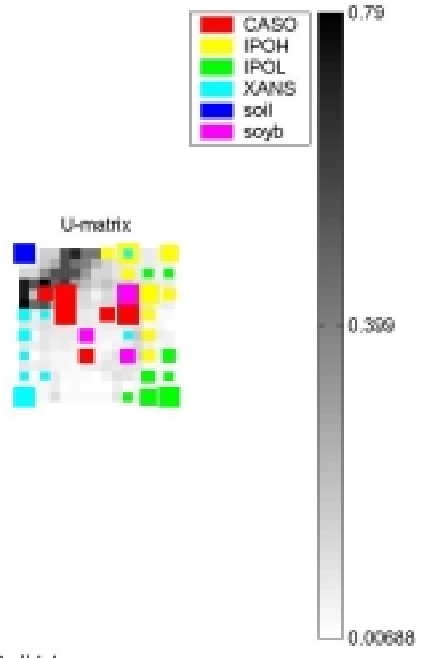


XANS


Labels

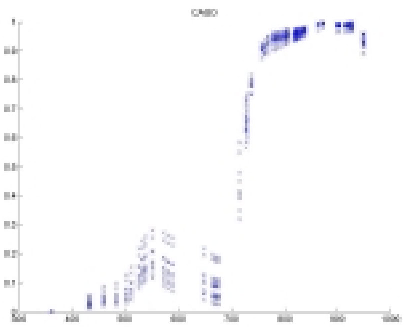
soil soil(4)	soil	soil	CASO	IPOH IPOH(1)	IPOH IPOH(3) XANS(1)	IPOH	IPOH IPOH(2)
soil	soil	CASO	CASO	IPOH	IPOH IPOH(1)	IPOH IPOL(1)	IPOH IPOL(1)
soil	CASO CASO(1)	CASO CASO(2)	CASO	CASO	IPOH soyb(2)	IPOH IPOH(3)	IPOH IPOH(1)
XANS XANS(2)	XANS XANS(1)	CASO CASO(2)	CASO	CASO CASO(1)	CASO CASO(2)	IPOH IPOH(1)	IPOH
XANS XANS(2)	XANS	CASO	CASO soyb(1)	CASO	CASO XANS(1)	IPOH IPOH(1)	IPOH
XANS XANS(1)	XANS	XANS	CASO CASO(1)	CASO	CASO soyb(1)	IPOL IPOH(1)	IPOL IPOL(2) IPOH(1)
XANS XANS(1)	XANS XANS(1)	soyb	soyb	soyb	IPOL	IPOL IPOL(2)	IPOL IPOL(1)
XANS XANS(6) CASO(2)	XANS	soyb	soyb	soyb	IPOL IPOL(1)	IPOL IPOL(3)	IPOL IPOL(5) IPOH(1)

072500$_f$ull.txt

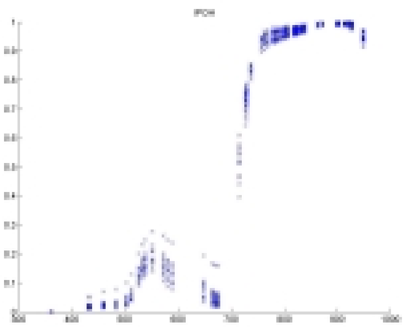


072500$_f$ull.txt

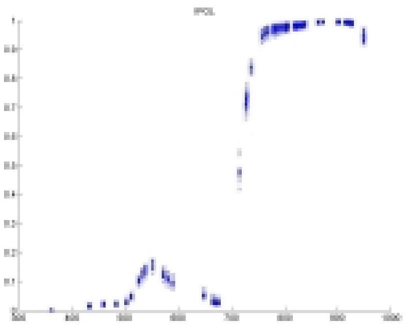
D.3 Trial 3



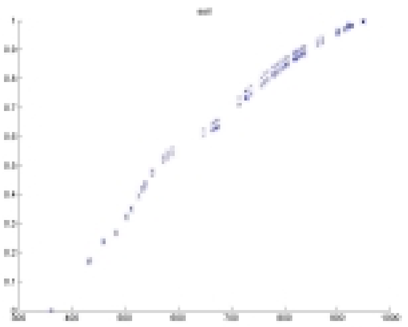
CASO



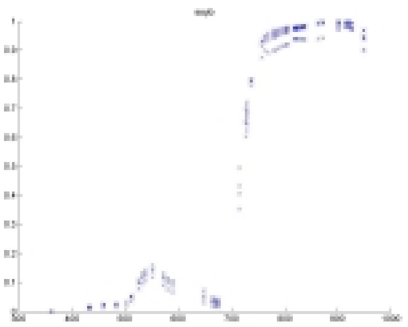
IPOH



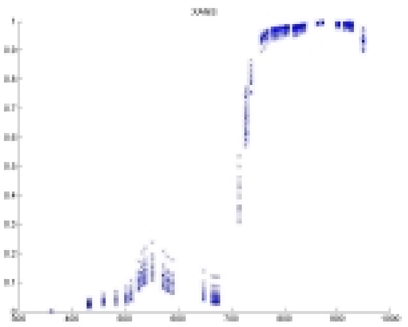
IPOL



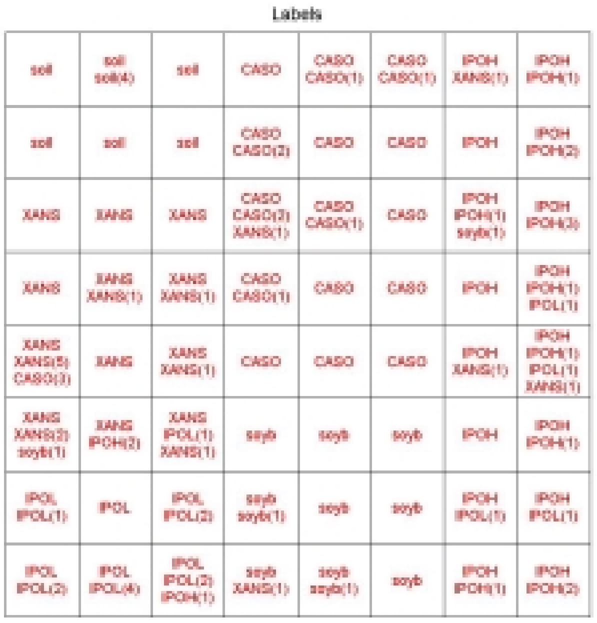
SOIL



SOYB



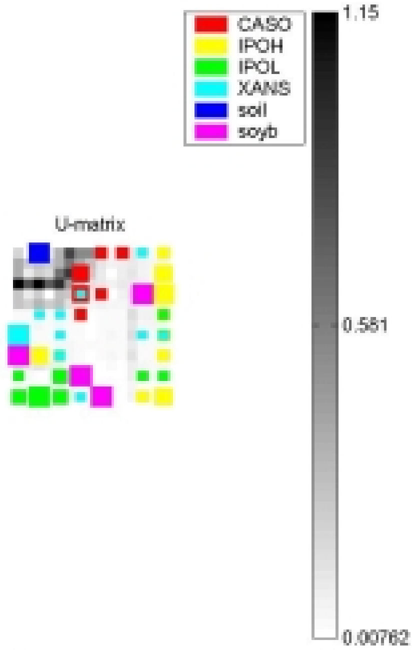
XANS



Labels

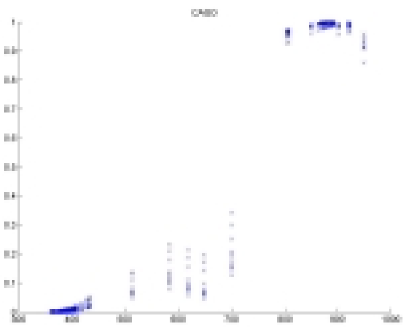
soil	soil soil(4)	soil	CASO	CASO CASO(1)	CASO CASO(1)	IPOH XANS(1)	IPOH IPOH(1)
soil	soil	soil	CASO CASO(2)	CASO	CASO	IPOH	IPOH IPOH(2)
XANS	XANS	XANS	CASO CASO(3) XANS(1)	CASO CASO(1)	CASO	IPOH IPOH(1) soyb(1)	IPOH IPOH(3)
XANS	XANS XANS(1)	XANS XANS(1)	CASO CASO(1)	CASO	CASO	IPOH	IPOH IPOH(1) IPOL(1)
XANS XANS(5) CASO(3)	XANS	XANS XANS(1)	CASO	CASO	CASO	IPOH XANS(1)	IPOH IPOH(1) IPOL(1) XANS(1)
XANS XANS(2) soyb(1)	XANS IPOH(2)	XANS IPOL(1) XANS(1)	soyb	soyb	soyb	IPOH	IPOH IPOH(1)
IPOL IPOL(1)	IPOL	IPOL IPOL(2)	soyb soyb(1)	soyb	soyb	IPOH IPOL(1)	IPOH IPOL(1)
IPOL IPOL(2)	IPOL IPOL(4)	IPOL IPOL(2) IPOH(1)	soyb XANS(1)	soyb soyb(1)	soyb	IPOH IPOH(1)	IPOH IPOH(2)

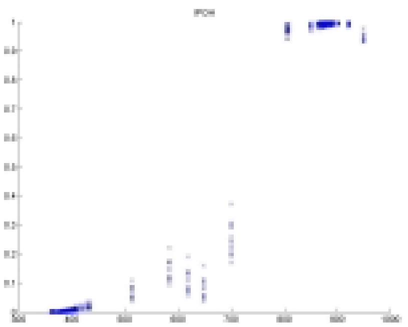
072500$_f$ull.txt

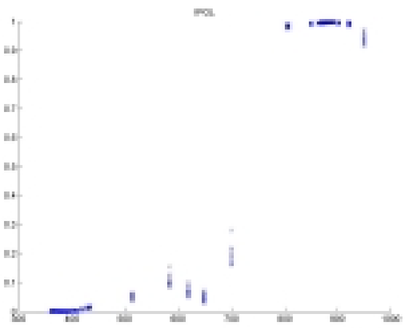


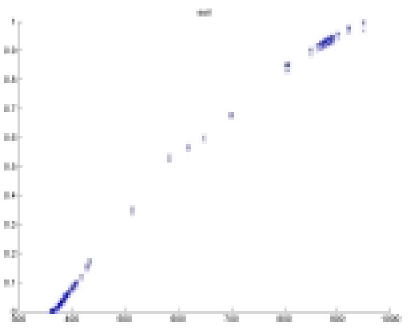
072500$_f$ull.txt

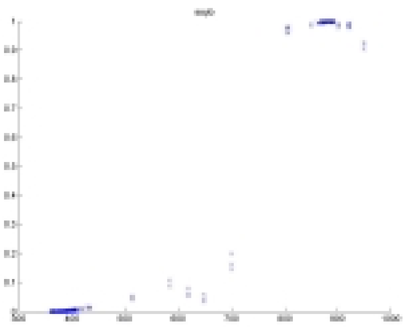
D.4 Trial 4

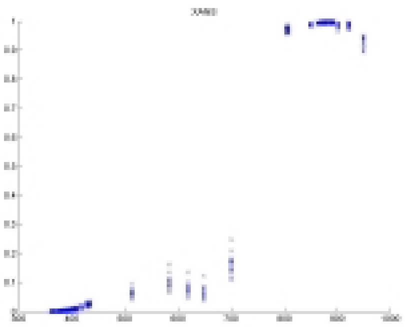

CASO

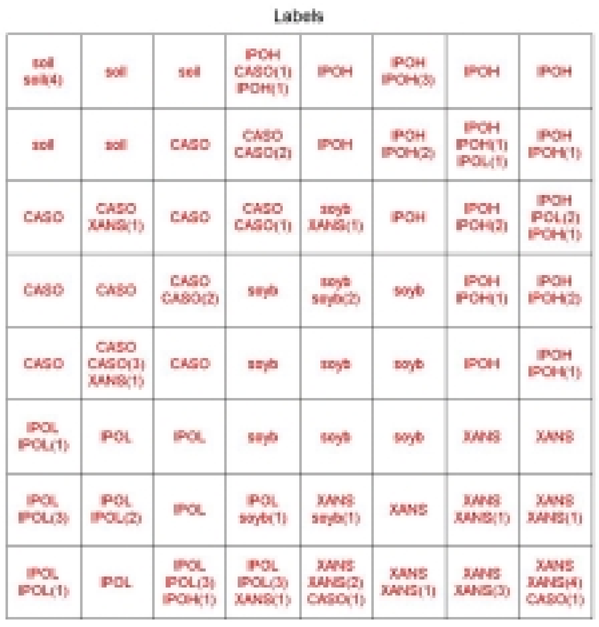

IPOH


IPOL


SOIL


SOYB

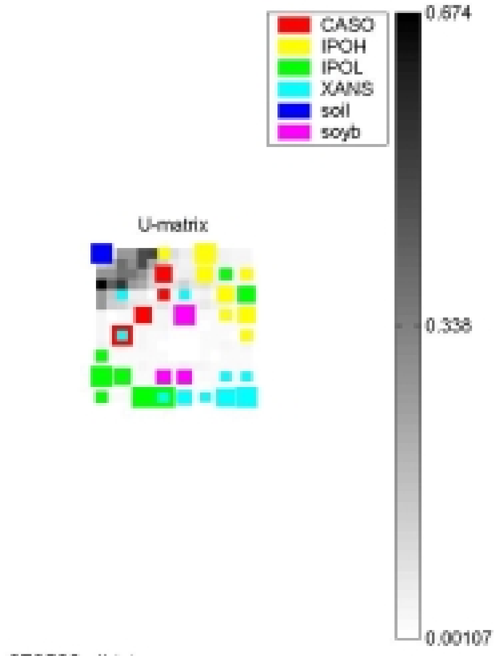

XANS



Labels

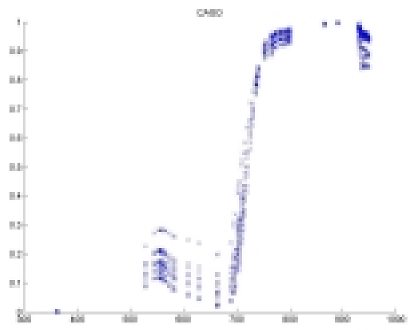
soil soil(4)	soil	soil	IPOH CASO(1) IPOH(1)	IPOH	IPOH IPOH(3)	IPOH	IPOH
soil	soil	CASO	CASO CASO(2)	IPOH	IPOH IPOH(2)	IPOH IPOH(1) IPOL(1)	IPOH IPOH(1)
CASO	CASO XANS(1)	CASO	CASO CASO(1)	soyb XANS(1)	IPOH	IPOH IPOH(2)	IPOH IPOL(2) IPOH(1)
CASO	CASO	CASO CASO(2)	soyb	soyb soyb(2)	soyb	IPOH IPOH(1)	IPOH IPOH(2)
CASO	CASO CASO(3) XANS(1)	CASO	soyb	soyb	soyb	IPOH	IPOH IPOH(1)
IPOL IPOL(1)	IPOL	IPOL	soyb	soyb	soyb	XANS	XANS
IPOL IPOL(3)	IPOL IPOL(2)	IPOL	IPOL soyb(1)	XANS soyb(1)	XANS	XANS XANS(1)	XANS XANS(1)
IPOL IPOL(1)	IPOL	IPOL IPOL(3) IPOH(1)	IPOL IPOL(3) XANS(1)	XANS XANS(2) CASO(1)	XANS XANS(1)	XANS XANS(3)	XANS XANS(4) CASO(1)

072500$_f$ull.txt

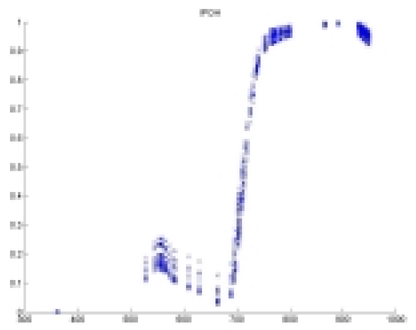


072500$_f$ull.txt

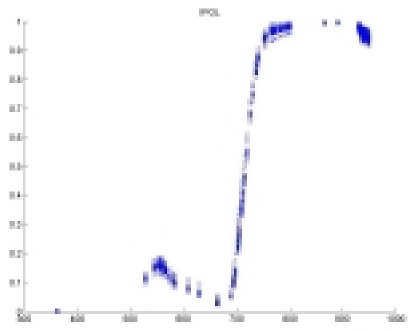
D.5 Trial 5



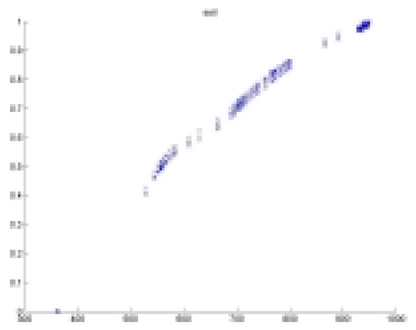
CASO



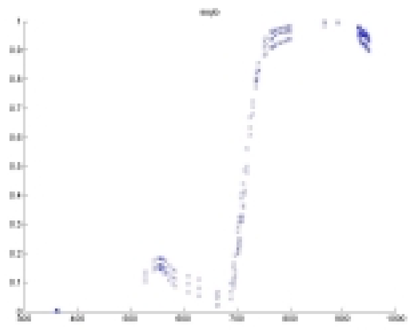
IPOH



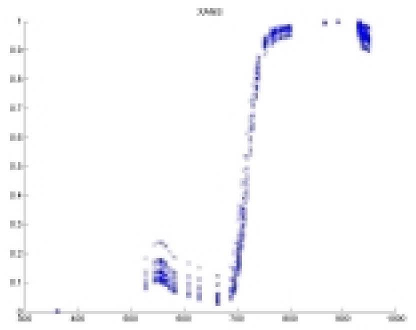
IPOL



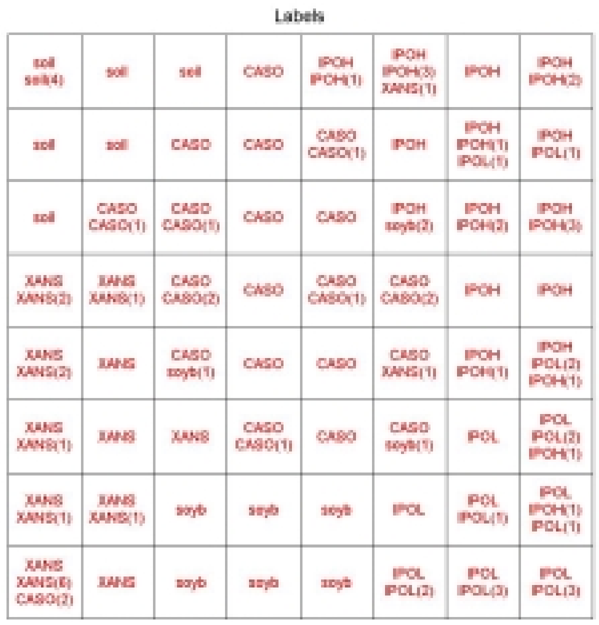
SOIL



SOYB



XANS


Labels

soil
soil(4)
	soil	soil	CASO	IPOH
				IPOH(1)
					IPOH
					IPOH(3)
					XANS(1)
						IPOH	IPOH
							IPOH(2)

soil	soil	CASO	CASO	CASO
				CASO(1)
					IPOH	IPOH
						IPOH(1)
						IPOL(1)
							IPOH
							IPOL(1)

soil	CASO CASO(1)	CASO CASO(1)	CASO	CASO	IPOH soyb(2)	IPOH IPOH(2)	IPOH IPOH(3)
XANS XANS(2)	XANS XANS(1)	CASO CASO(2)	CASO	CASO CASO(1)	CASO CASO(2)	IPOH	IPOH

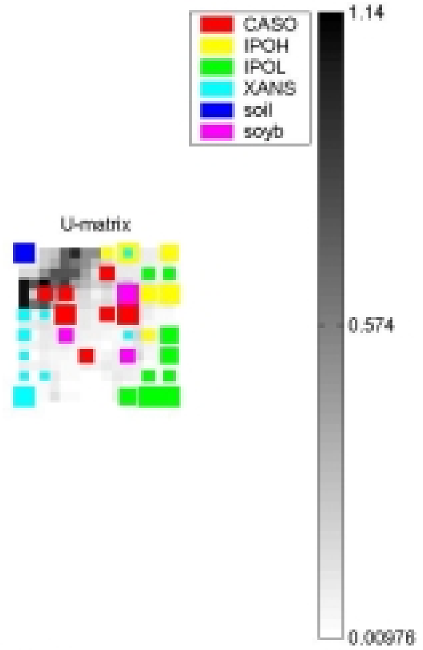
XANS
XANS(2)
	XANS	CASO
		soyb(1)
			CASO	CASO	CASO
					XANS(1)
						IPOH
						IPOH(1)
							IPOH
							IPOL(2)
							IPOH(1)

XANS
XANS(1)
	XANS	XANS	CASO
			CASO(1)
				CASO	CASO
					soyb(1)
						IPOL	IPOL
							IPOL(2)
							IPOH(1)

XANS
XANS(1)
	XANS
	XANS(1)
		soyb	soyb	soyb	IPOL	IPOL
						IPOL(1)
							IPOL
							IPOH(1)
							IPOL(1)

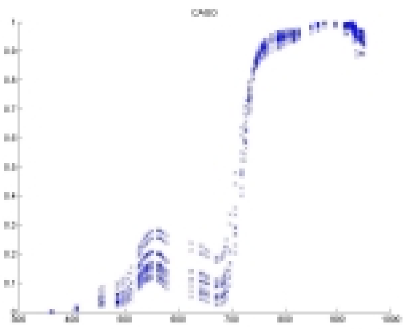
XANS
XANS(6)
CASO(2)
	XANS	soyb	soyb	soyb	IPOL
					IPOL(2)
						IPOL
						IPOL(3)
							IPOL
							IPOL(3)

072500$_{f}$ull.txt

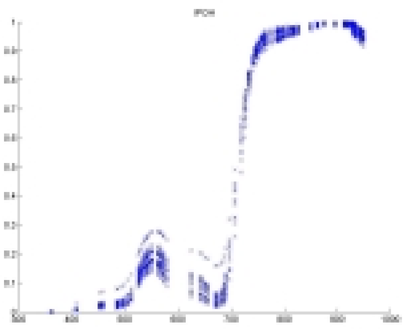


072500$_{f}$ull.txt

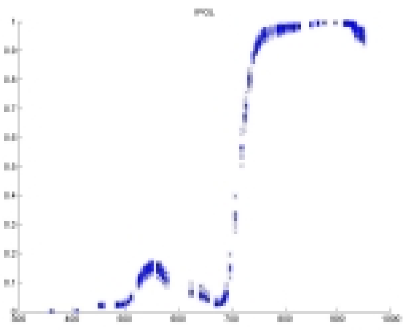
D.6 Trial 6



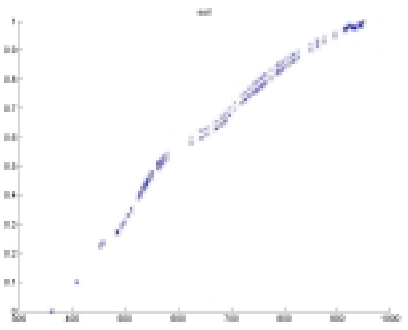
CASO



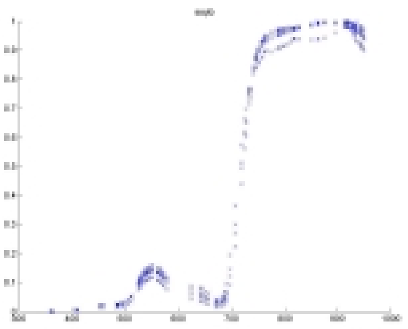
IPOH



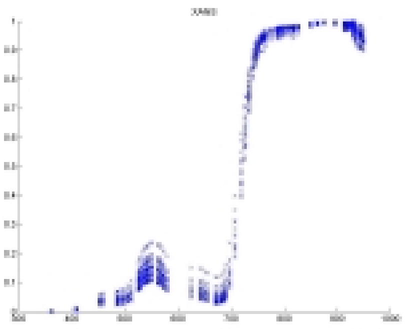
IPOL



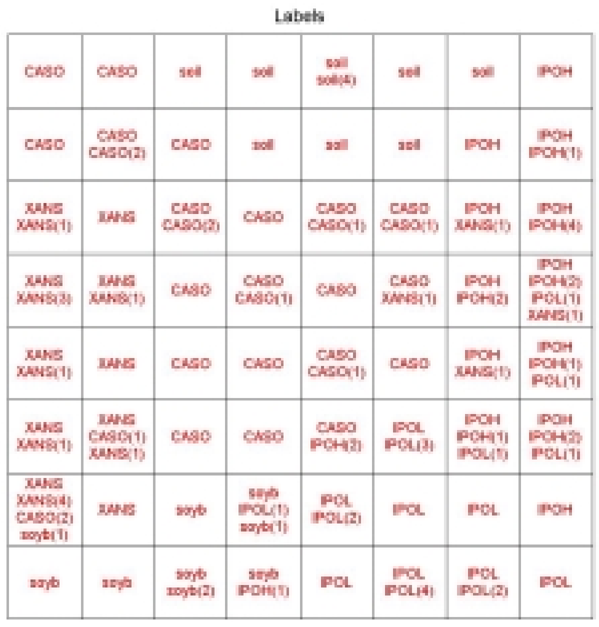
SOIL



SOYB



XANS



Labels

CASO	CASO	soil	soil	soil soil(4)	soil	soil	IPOH
CASO	CASO CASO(2)	CASO	soil	soil	soil	IPOH	IPOH IPOH(1)
XANS XANS(1)	XANS	CASO CASO(2)	CASO	CASO CASO(1)	CASO CASO(1)	IPOH XANS(1)	IPOH IPOH(4)

XANS
XANS(3)
	XANS
	XANS(1)
		CASO	CASO
			CASO(1)
				CASO	CASO
					XANS(1)
						IPOH
						IPOH(2)
							IPOH
							IPOH(2)
							IPOL(1)
							XANS(1)

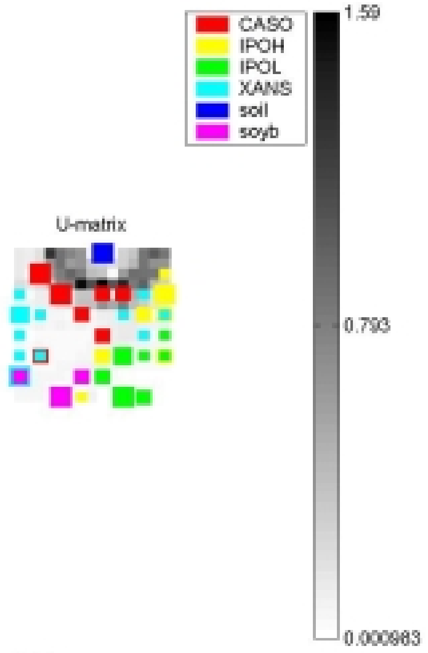
XANS
XANS(1)
	XANS	CASO	CASO	CASO
				CASO(1)
					CASO	IPOH
						XANS(1)
							IPOH
							IPOH(1)
							IPOL(1)

XANS
XANS(1)
	XANS
	CASO(1)
	XANS(1)
		CASO	CASO	CASO
				IPOH(2)
					IPOL
					IPOL(3)
						IPOH
						IPOH(1)
						IPOL(1)
							IPOH
							IPOH(2)
							IPOL(1)

XANS
XANS(4)
CASO(2)
soyb(1)
	XANS	soyb	soyb
			IPOL(1)
			soyb(1)
				IPOL
				IPOL(2)
					IPOL	IPOL	IPOH

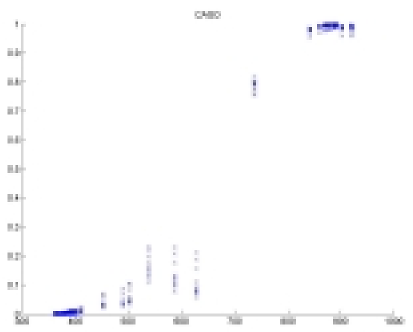
soyb	soyb	soyb soyb(2)	soyb IPOH(1)	IPOL	IPOL IPOL(4)	IPOL IPOL(2)	IPOL

072500$_{f}$ull.txt

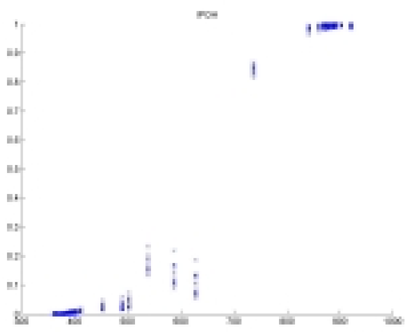


072500$_{f}$ull.txt

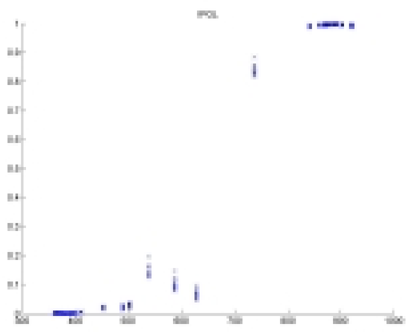
D.7 Trial 7



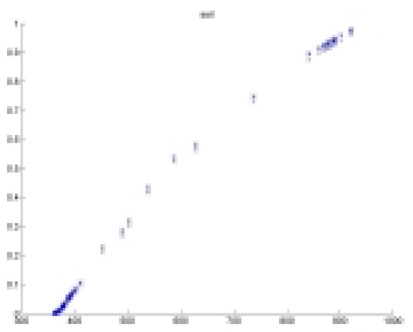
CASO



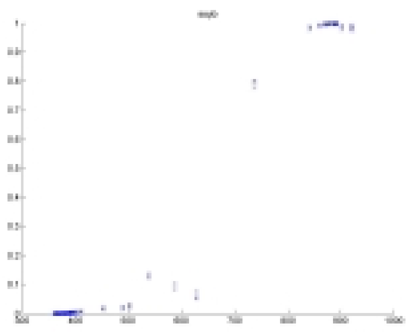
IPOH



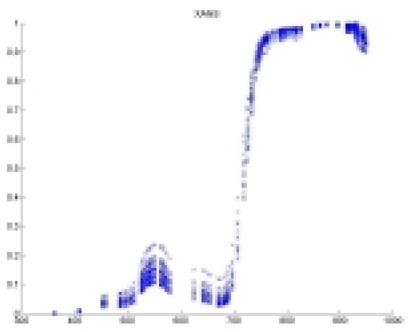
IPOL



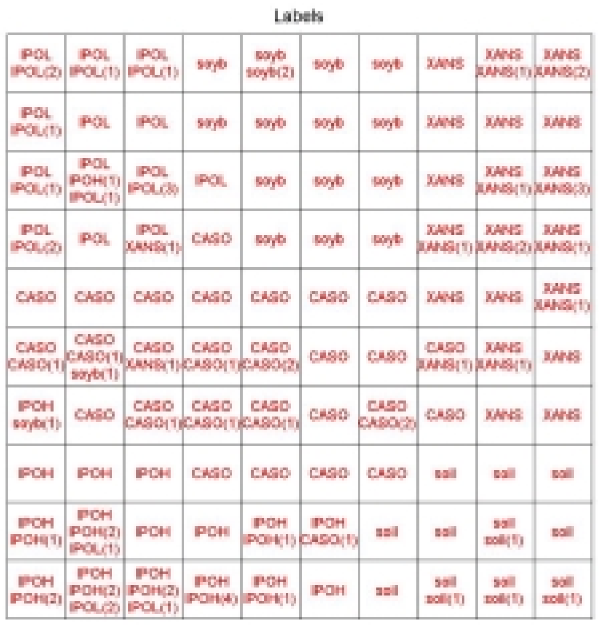
SOIL



SOYB



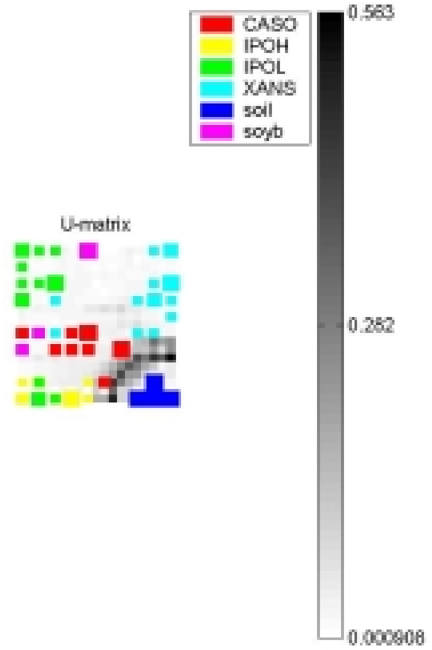
XANS



Labels

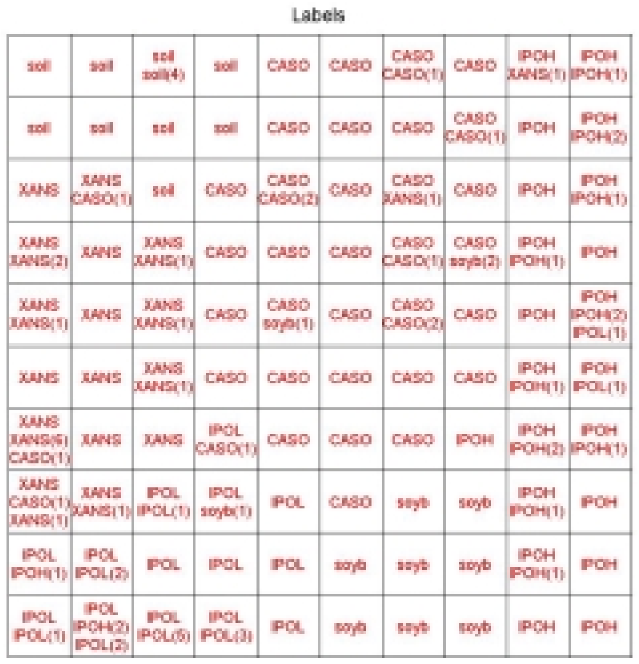
IPOL IPOL(2)	IPOL IPOL(1)	IPOL IPOL(1)	soyb	soyb soyb(2)	soyb	soyb	XANS	XANS XANS(1)	XANS XANS(2)
IPOL IPOL(1)	IPOL	IPOL	soyb	soyb	soyb	soyb	XANS	XANS	XANS
IPOL IPOL(1)	IPOL IPOH(1) IPOL(1)	IPOL IPOL(3)	IPOL	soyb	soyb	soyb	XANS	XANS XANS(1)	XANS XANS(3)
IPOL IPOL(2)	IPOL	IPOL XANS(1)	CASO	soyb	soyb	soyb	XANS XANS(1)	XANS XANS(2)	XANS XANS(1)
CASO	CASO	CASO	CASO	CASO	CASO	CASO	XANS	XANS	XANS XANS(1)
CASO CASO(1)	CASO CASO(1) soyb(1)	CASO XANS(1)	CASO CASO(1)	CASO CASO(2)	CASO	CASO	CASO XANS(1)	XANS XANS(1)	XANS
IPOH soyb(1)	CASO	CASO CASO(1)	CASO CASO(1)	CASO CASO(1)	CASO	CASO CASO(2)	CASO	XANS	XANS
IPOH	IPOH	IPOH	CASO	CASO	CASO	CASO	soil	soil	soil
IPOH IPOH(1)	IPOH IPOH(2) IPOL(1)	IPOH	IPOH	IPOH IPOH(1)	IPOH CASO(1)	soil	soil	soil soil(1)	soil
IPOH IPOH(2)	IPOH IPOH(2) IPOL(2)	IPOH IPOH(2) IPOL(1)	IPOH IPOH(4)	IPOH IPOH(1)	IPOH	soil	soil soil(1)	soil soil(1)	soil soil(1)

072500full.txt



072500full.txt

D.8 Trial 8



Labels

soil	soil	soil soil(4)	soil	CASO	CASO	CASO CASO(1)	CASO	IPOH XANS(1)	IPOH IPOH(1)
soil	soil	soil	soil	CASO	CASO	CASO	CASO CASO(1)	IPOH	IPOH IPOH(2)
XANS	XANS CASO(1)	soil	CASO	CASO CASO(2)	CASO	CASO XANS(1)	CASO	IPOH	IPOH IPOH(1)
XANS XANS(2)	XANS	XANS XANS(1)	CASO	CASO	CASO	CASO CASO(1)	CASO soyb(2)	IPOH IPOH(1)	IPOH

XANS
XANS(1)
	XANS	XANS
		XANS(1)
			CASO	CASO
				soyb(1)
					CASO	CASO
						CASO(2)
							CASO	IPOH	IPOH
									IPOH(2)
									IPOL(1)

XANS	XANS	XANS XANS(1)	CASO	CASO	CASO	CASO	CASO	IPOH IPOH(1)	IPOH IPOL(1)

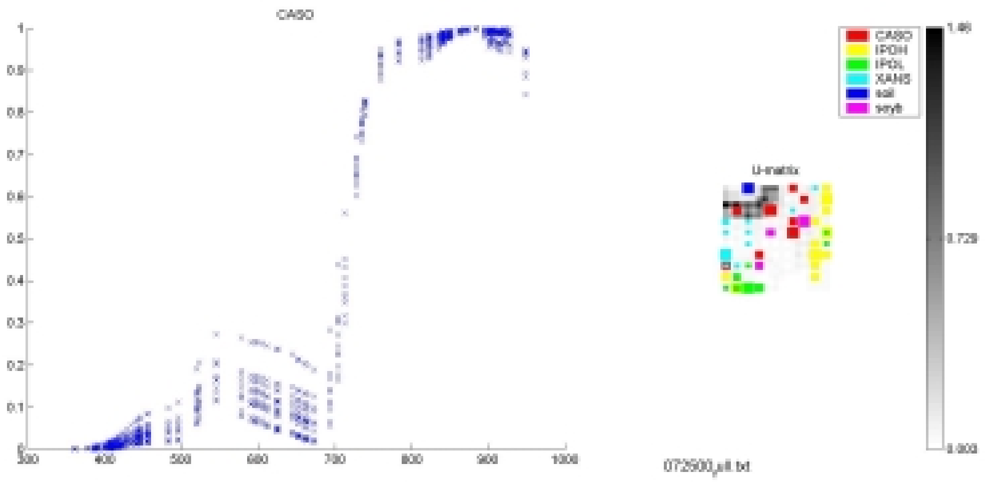
XANS
XANS(6)
CASO(1)
	XANS	XANS	IPOL
			CASO(1)
				CASO	CASO	CASO	IPOH	IPOH
								IPOH(2)
									IPOH
									IPOH(1)

XANS
CASO(1)
XANS(1)
	XANS
	XANS(1)
		IPOL
		IPOL(1)
			IPOL
			soyb(1)
				IPOL	CASO	soyb	soyb	IPOH
								IPOH(1)
									IPOH

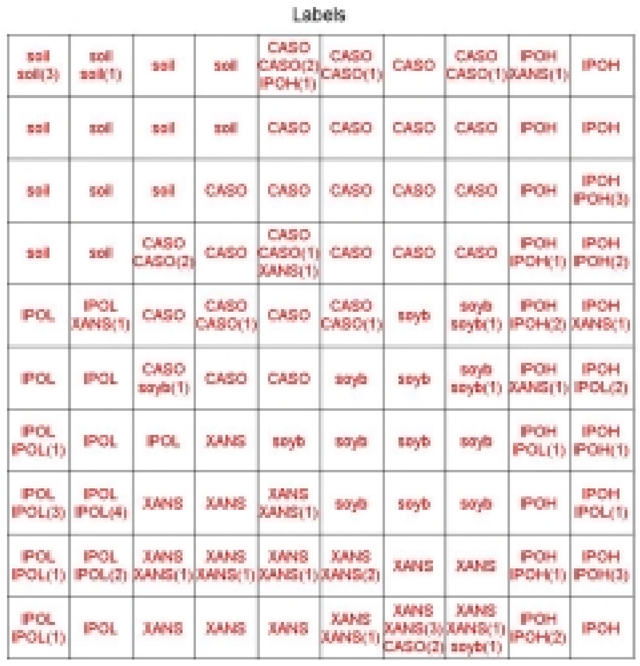
IPOL IPOH(1)	IPOL IPOL(2)	IPOL	IPOL	IPOL	soyb	soyb	soyb	IPOH IPOH(1)	IPOH

IPOL
IPOL(1)
	IPOL
	IPOH(2)
	IPOL(2)
		IPOL
		IPOL(5)
			IPOL
			IPOL(3)
				IPOL	soyb	soyb	soyb	IPOH	IPOH

072500$_f$ull.txt



D.9 Trial 9


Labels

soil
soil(3)
	soil
	soil(1)
		soil	soil	CASO
				CASO(2)
				IPOH(1)
					CASO
					CASO(1)
						CASO	CASO
							CASO(1)
								IPOH
								XANS(1)
									IPOH

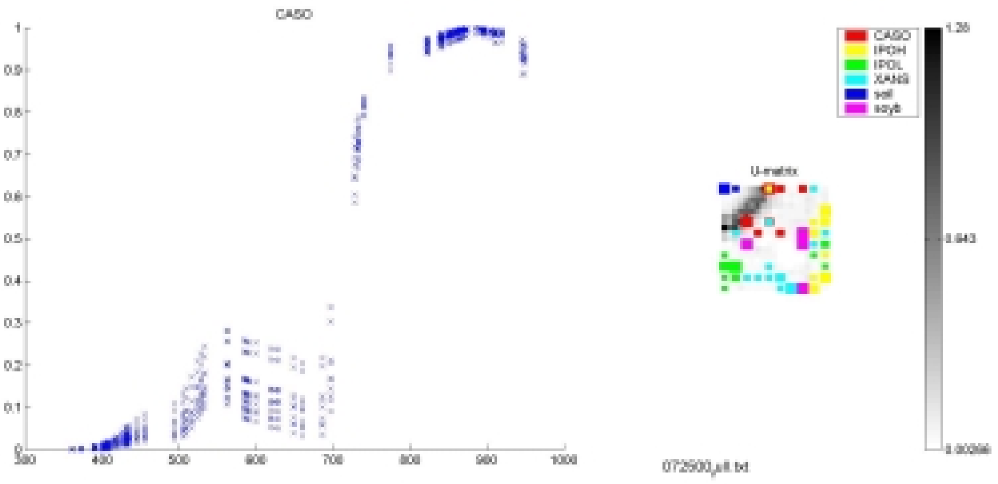
soil	soil	soil	soil	CASO	CASO	CASO	CASO	IPOH	IPOH
soil	soil	soil	CASO	CASO	CASO	CASO	CASO	IPOH	IPOH IPOH(3)

soil	soil	CASO
		CASO(2)
			CASO	CASO
				CASO(1)
				XANS(1)
					CASO	CASO	CASO	IPOH
								IPOH(1)
									IPOH
									IPOH(2)

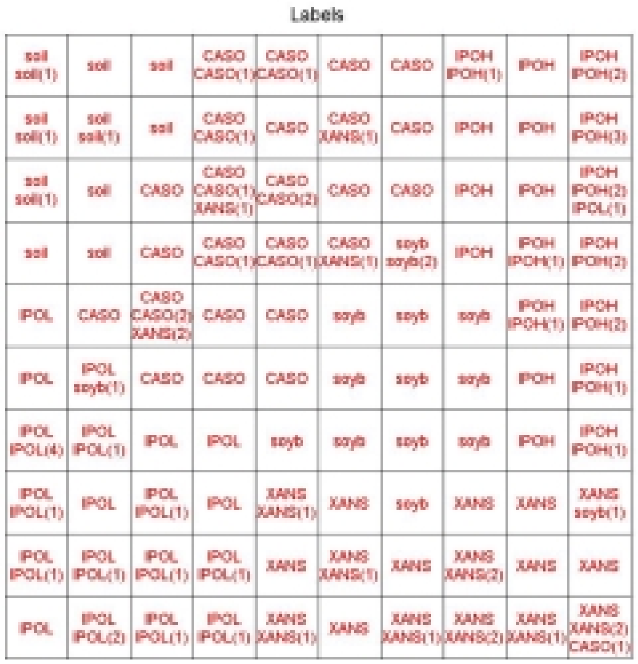
IPOL	IPOL XANS(1)	CASO	CASO CASO(1)	CASO	CASO CASO(1)	soyb	soyb soyb(1)	IPOH IPOH(2)	IPOH XANS(1)
IPOL	IPOL	CASO soyb(1)	CASO	CASO	soyb	soyb	soyb soyb(1)	IPOH XANS(1)	IPOH IPOL(2)
IPOL IPOL(1)	IPOL	IPOL	XANS	soyb	soyb	soyb	soyb	IPOH IPOL(1)	IPOH IPOH(1)
IPOL IPOL(3)	IPOL IPOL(4)	XANS	XANS	XANS XANS(1)	soyb	soyb	soyb	IPOH	IPOH IPOL(1)
IPOL IPOL(1)	IPOL IPOL(2)	XANS XANS(1)	XANS XANS(1)	XANS XANS(1)	XANS XANS(2)	XANS	XANS	IPOH IPOH(1)	IPOH IPOH(3)

IPOL
IPOL(1)
	IPOL	XANS	XANS	XANS	XANS
					XANS(1)
						XANS
						XANS(3)
						CASO(2)
							XANS
							XANS(1)
							soyb(1)
								IPOH
								IPOH(2)
									IPOH

072500ull.txt

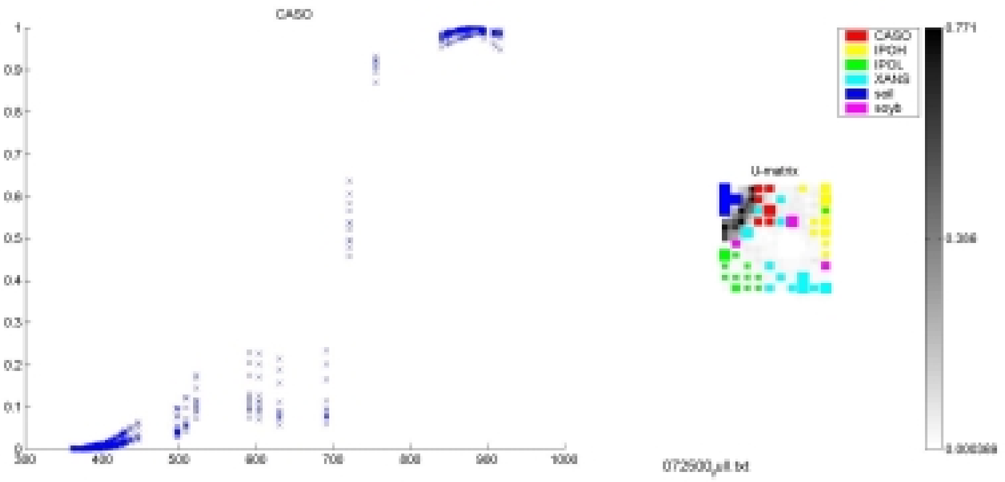


D.10 Trial 10

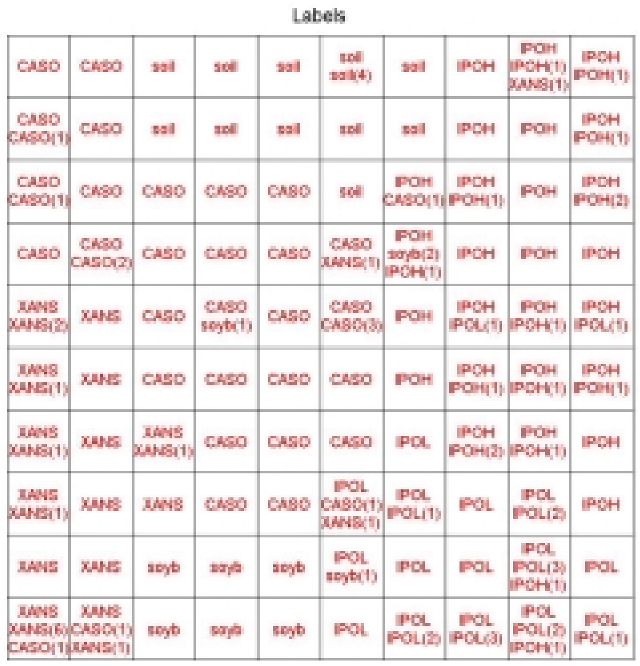

Labels

soil soil(1)	soil	soil	CASO CASO(1)	CASO CASO(1)	CASO	CASO	IPOH IPOH(1)	IPOH	IPOH IPOH(2)
soil soil(1)	soil soil(1)	soil	CASO CASO(1)	CASO	CASO XANS(1)	CASO	IPOH	IPOH	IPOH IPOH(3)
soil soil(1)	soil	CASO	CASO CASO(1) XANS(1)	CASO CASO(2)	CASO	CASO	IPOH	IPOH	IPOH IPOH(2) IPOL(1)
soil	soil	CASO	CASO CASO(1)	CASO CASO(1)	CASO XANS(1)	soyb soyb(2)	IPOH	IPOH IPOH(1)	IPOH IPOH(2)
IPOL	CASO	CASO CASO(2) XANS(2)	CASO	CASO	soyb	soyb	soyb	IPOH IPOH(1)	IPOH IPOH(2)
IPOL	IPOL soyb(1)	CASO	CASO	CASO	soyb	soyb	soyb	IPOH	IPOH IPOH(1)
IPOL IPOL(4)	IPOL IPOL(1)	IPOL	IPOL	soyb	soyb	soyb	soyb	IPOH	IPOH IPOH(1)
IPOL IPOL(1)	IPOL	IPOL IPOL(1)	IPOL	XANS XANS(1)	XANS	soyb	XANS	XANS	XANS soyb(1)
IPOL IPOL(1)	IPOL IPOL(1)	IPOL IPOL(1)	IPOL IPOL(1)	XANS	XANS XANS(1)	XANS	XANS XANS(2)	XANS	XANS
IPOL	IPOL IPOL(2)	IPOL IPOL(1)	IPOL IPOL(1)	XANS XANS(1)	XANS	XANS XANS(1)	XANS XANS(2)	XANS XANS(1)	XANS XANS(2) CASO(1)

072500$_u$ll.txt



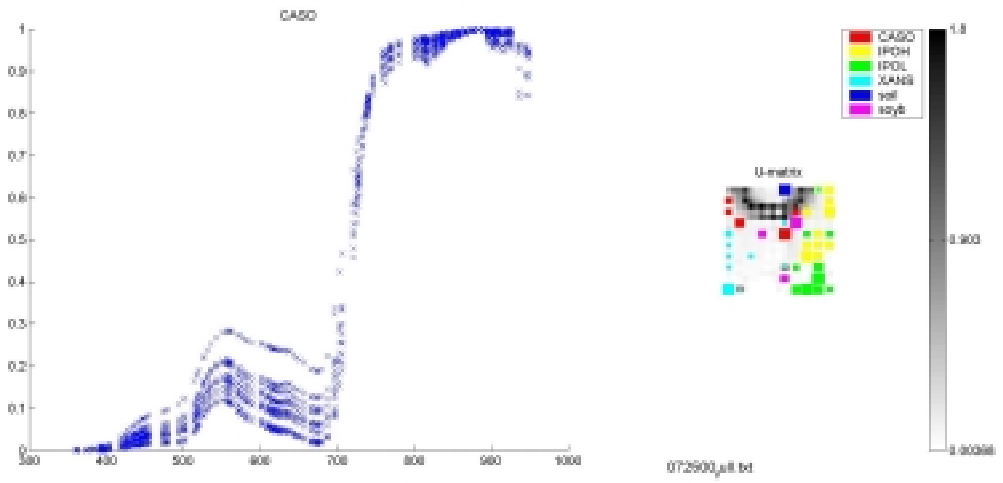
D.11 Trial 11



Labels

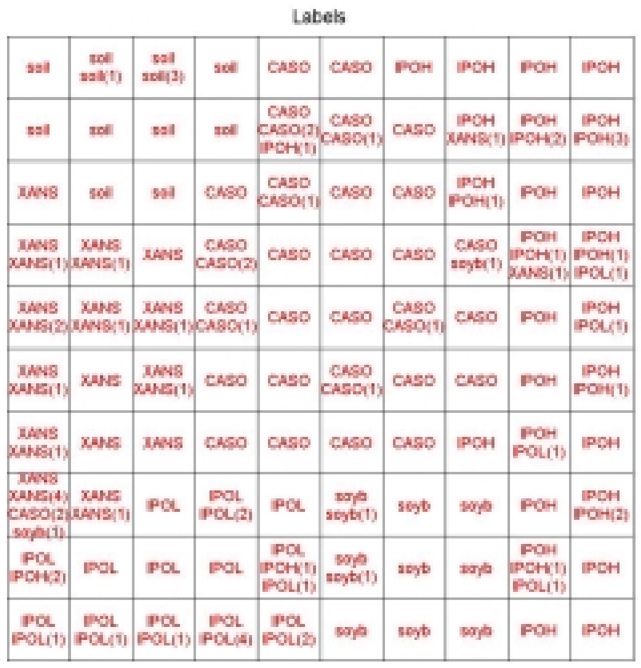
CASO	CASO	soil	soil	soil	soil soil(4)	soil	IPOH	IPOH IPOH(1) XANS(1)	IPOH IPOH(1)
CASO CASO(1)	CASO	soil	soil	soil	soil	soil	IPOH	IPOH	IPOH IPOH(1)
CASO CASO(1)	CASO	CASO	CASO	CASO	soil	IPOH CASO(1)	IPOH IPOH(1)	IPOH	IPOH IPOH(2)
CASO	CASO CASO(2)	CASO	CASO	CASO	CASO XANS(1)	IPOH soyb(2) IPOH(1)	IPOH	IPOH	IPOH
XANS XANS(2)	XANS	CASO	CASO soyb(1)	CASO	CASO CASO(3)	IPOH	IPOH IPOL(1)	IPOH IPOH(1)	IPOH IPOL(1)
XANS XANS(1)	XANS	CASO	CASO	CASO	CASO	IPOH	IPOH IPOH(1)	IPOH IPOH(1)	IPOH IPOH(1)
XANS XANS(1)	XANS	XANS XANS(1)	CASO	CASO	CASO	IPOL	IPOH IPOH(2)	IPOH IPOH(1)	IPOH
XANS XANS(1)	XANS	XANS	CASO	CASO	IPOL CASO(1) XANS(1)	IPOL IPOL(1)	IPOL	IPOL IPOL(2)	IPOH
XANS	XANS	soyb	soyb	soyb	IPOL soyb(1)	IPOL	IPOL	IPOL IPOL(3) IPOH(1)	IPOL
XANS XANS(6) CASO(1)	XANS CASO(1) XANS(1)	soyb	soyb	soyb	IPOL	IPOL IPOL(2)	IPOL IPOL(3)	IPOL IPOL(2) IPOH(1)	IPOL IPOL(1)

072500full.txt



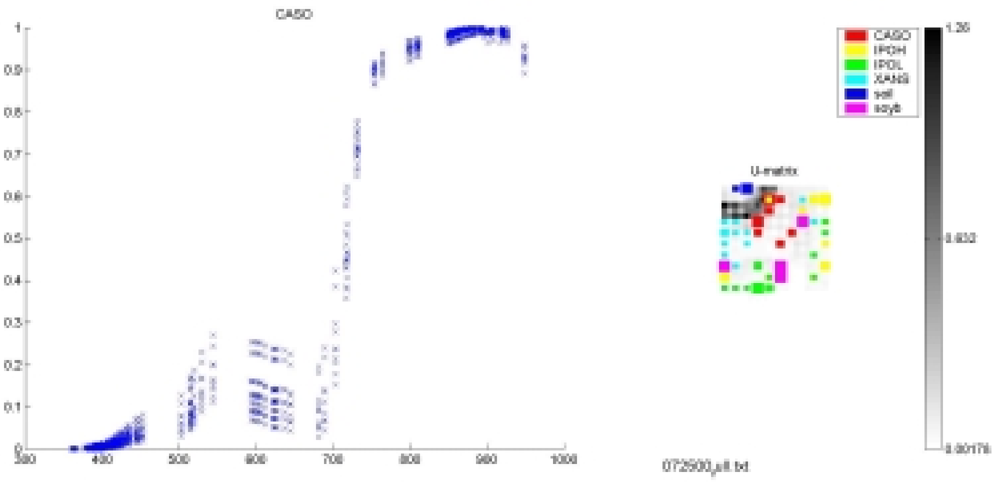
072500full.txt

D.12 Trial 12

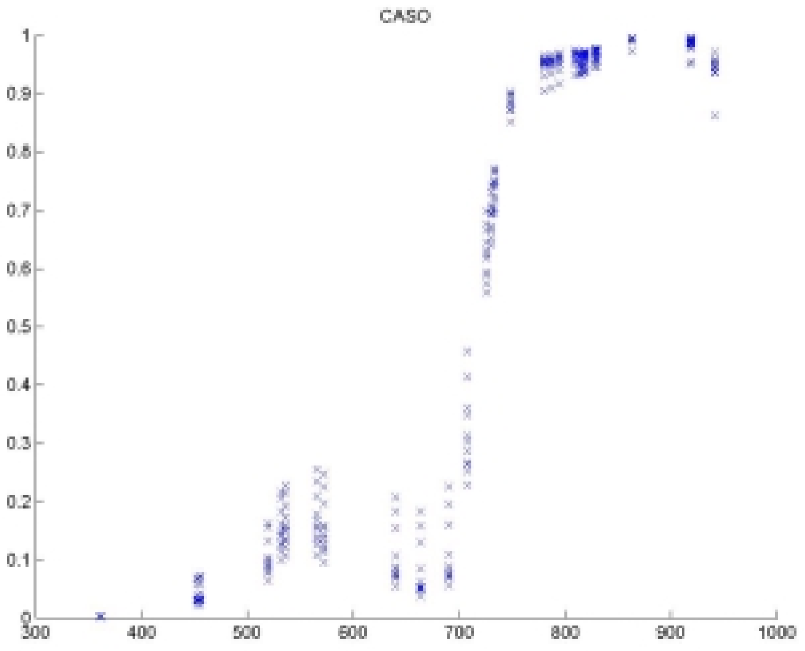

Labels

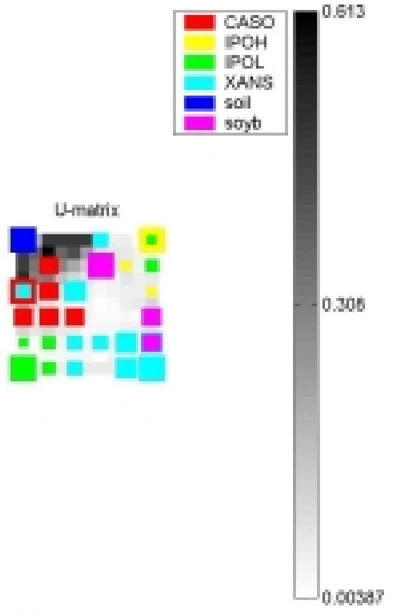
soil	soil soil(1)	soil soil(3)	soil	CASO	CASO	IPOH	IPOH	IPOH	IPOH
soil	soil	soil	soil	CASO CASO(2) IPOH(1)	CASO CASO(1)	CASO	IPOH XANS(1)	IPOH IPOH(2)	IPOH IPOH(3)
XANS	soil	soil	CASO	CASO CASO(1)	CASO	CASO	IPOH IPOH(1)	IPOH	IPOH
XANS XANS(1)	XANS XANS(1)	XANS	CASO CASO(2)	CASO	CASO	CASO	CASO soyb(1)	IPOH IPOH(1) XANS(1)	IPOH IPOH(1) IPOL(1)
XANS XANS(2)	XANS XANS(1)	XANS XANS(1)	CASO CASO(1)	CASO	CASO	CASO CASO(1)	CASO	IPOH	IPOH IPOL(1)
XANS XANS(1)	XANS	XANS XANS(1)	CASO	CASO	CASO CASO(1)	CASO	CASO	IPOH	IPOH IPOH(1)
XANS XANS(1)	XANS	XANS	CASO	CASO	CASO	CASO	IPOH	IPOH IPOL(1)	IPOH
XANS XANS(4) CASO(2) soyb(1)	XANS XANS(1)	IPOL	IPOL IPOL(2)	IPOL	soyb soyb(1)	soyb	soyb	IPOH	IPOH IPOH(2)
IPOL IPOH(2)	IPOL	IPOL	IPOL	IPOL IPOH(1) IPOL(1)	soyb soyb(1)	soyb	soyb	IPOH IPOH(1) IPOL(1)	IPOH
IPOL IPOL(1)	IPOL IPOL(1)	IPOL IPOL(1)	IPOL IPOL(4)	IPOL IPOL(2)	soyb	soyb	soyb	IPOH	IPOH

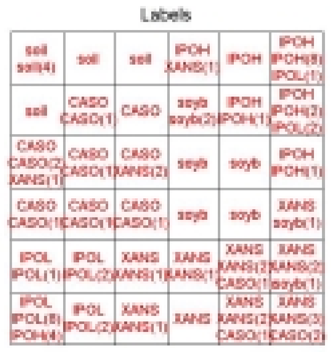
072500,ull.txt



D.13 Trial 13



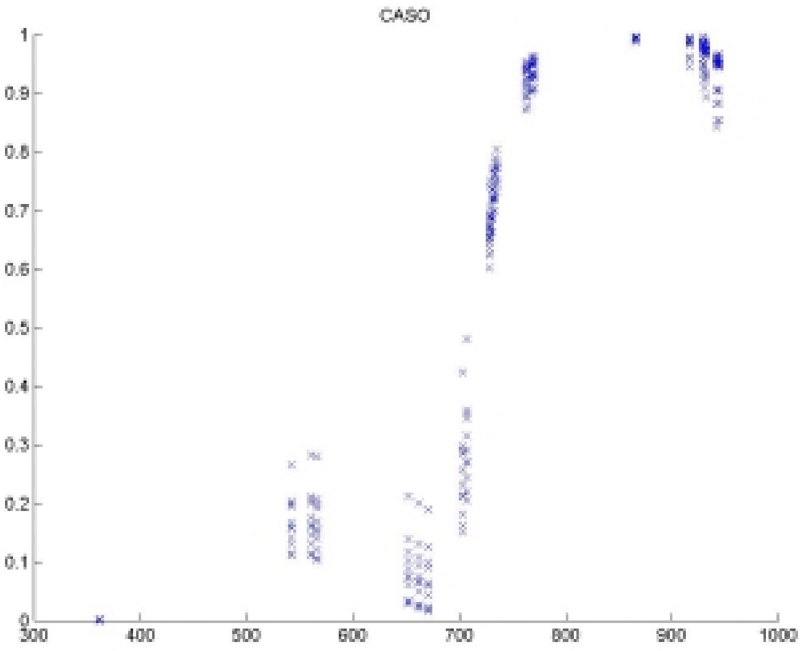


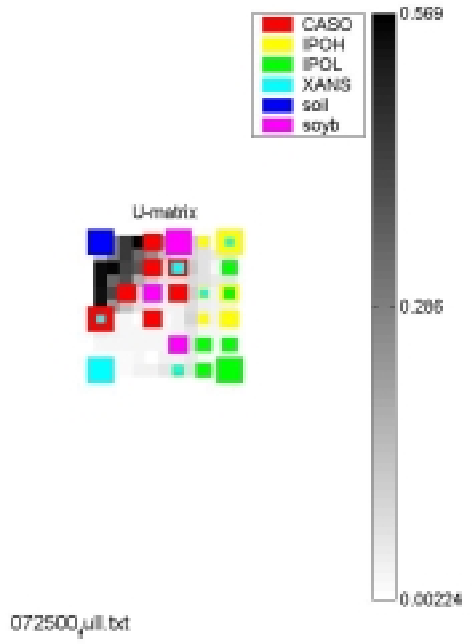

Labels

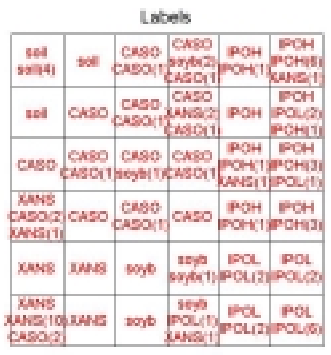
soil soil(4)	soil	soil	IPOH XANS(1)	IPOH	IPOH IPOH(8) IPOL(1)
soil	CASO CASO(1)	CASO	soyb soyb(2)	IPOH IPOH(1)	IPOH IPOH(3) IPOL(2)
CASO CASO(2 XANS(1)	CASO CASO(1	CASO XANS(2)	soyb	soyb	IPOH IPOH(1)
CASO CASO(1	CASO CASO(1	CASO CASO(1)	soyb	soyb	XANS soyb(1)
IPOL IPOL(1)	IPOL IPOL(2)	XANS XANS(1	XANS XANS(1	XANS XANS(2 CASO(1	XANS XANS(2) soyb(1)
IPOL IPOL(8) IPOH(4)	IPOL IPOL(2)	XANS XANS(1)	XANS	XANS XANS(2 CASO(1	XANS XANS(3) CASO(2)

072500_jull.txt

D.14 Trial 14

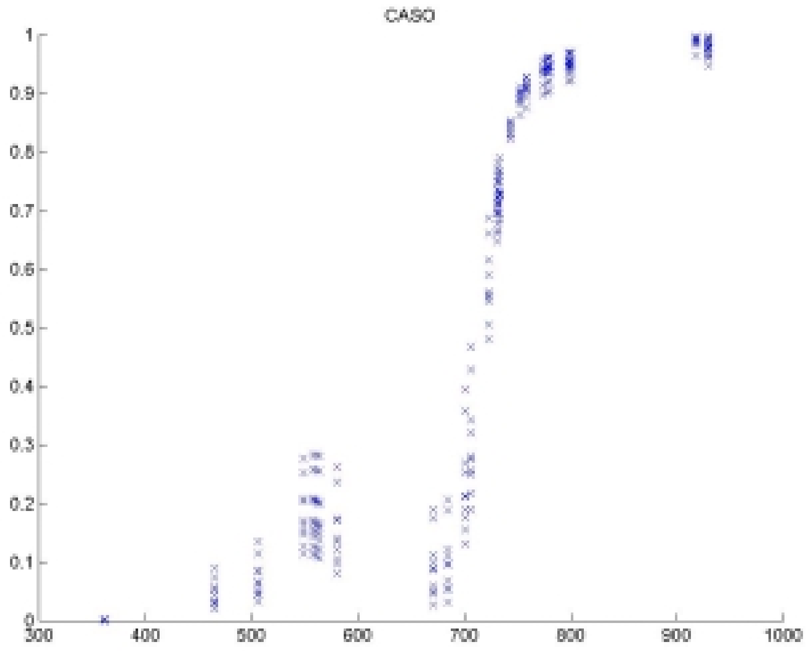


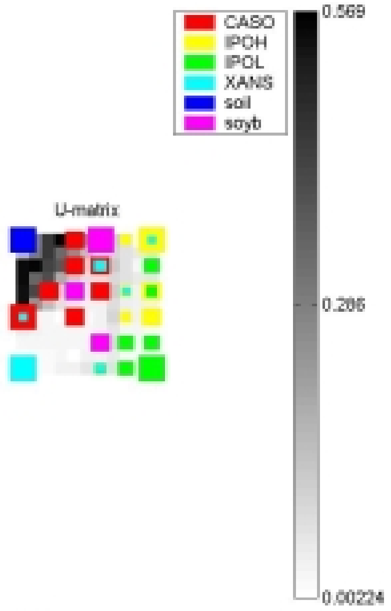


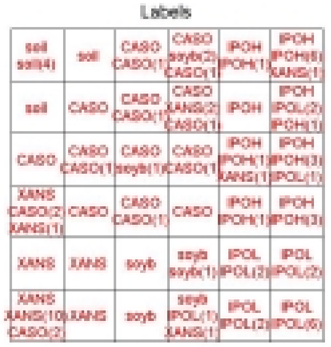

Labels

soil soil(4)	soil	CASO CASO(1)	CASO soyb(2) CASO(1)	IPOH IPOH(1)	IPOH IPOH(8) XANS(1)
soil	CASO	CASO CASO(1)	CASO XANS(2) CASO(1)	IPOH	IPOH IPOL(2) IPOH(1)
CASO	CASO CASO(1)	CASO soyb(1)	CASO CASO(1)	IPOH IPOH(1) XANS(1)	IPOH IPOH(3) IPOL(1)
XANS CASO(2) XANS(1)	CASO	CASO CASO(1)	CASO	IPOH IPOH(1)	IPOH IPOH(3)
XANS	XANS	soyb	soyb soyb(1)	IPOL IPOL(2)	IPOL IPOL(2)
XANS XANS(10) CASO(2)	XANS	soyb	soyb IPOL(1) XANS(1)	IPOL IPOL(2)	IPOL IPOL(6)

D.15 Trial 15



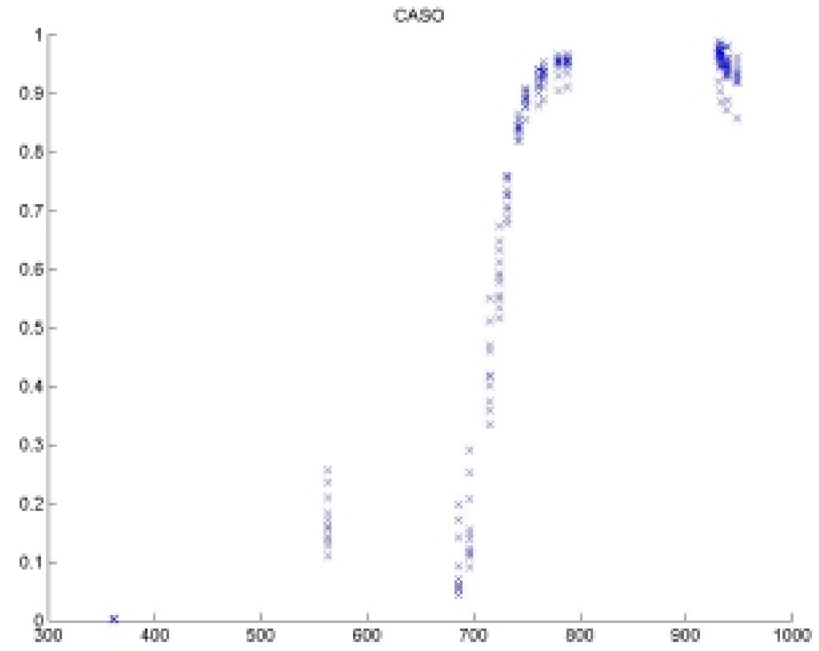


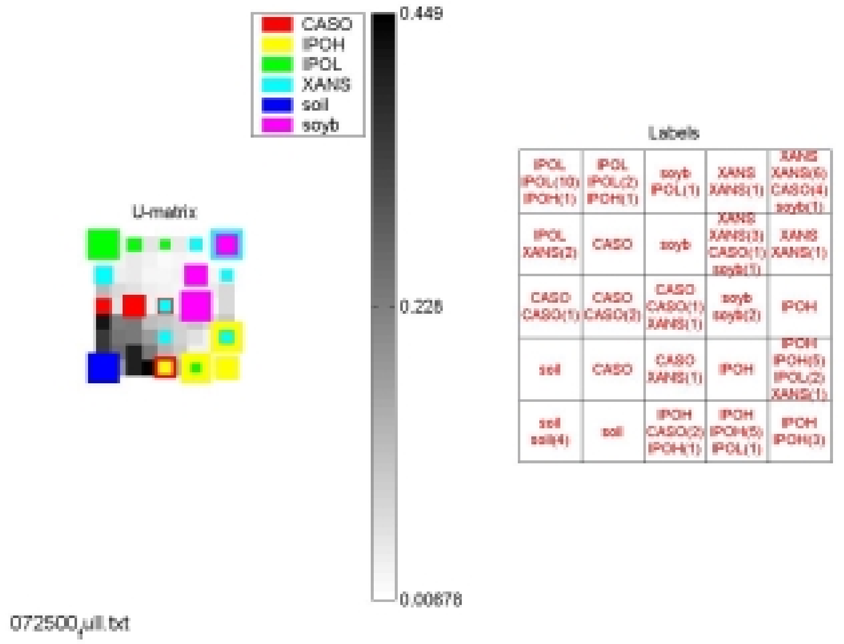

Labels

soil soil(4)	soil	CASO CASO(1)	CASO soyb(2) CASO(1)	IPOH IPOH(1)	IPOH IPOH(6) XANS(1)
soil	CASO	CASO CASO(1)	CASO XANS(2) CASO(1)	IPOH	IPOH IPOL(2) IPOH(1)
CASO	CASO CASO(1)	CASO soyb(1)	CASO CASO(1)	IPOH IPOH(1) XANS(1)	IPOH IPOH(3) IPOL(1)
XANS CASO(2) XANS(1)	CASO	CASO CASO(1)	CASO	IPOH IPOH(1)	IPOH IPOH(3)
XANS	XANS	soyb	soyb soyb(1)	IPOL IPOL(2)	IPOL IPOL(2)
XANS XANS(10) CASO(2)	XANS	soyb	soyb IPOL(1) XANS(1)	IPOL IPOL(2)	IPOL IPOL(6)

072500$_i$ull.txt

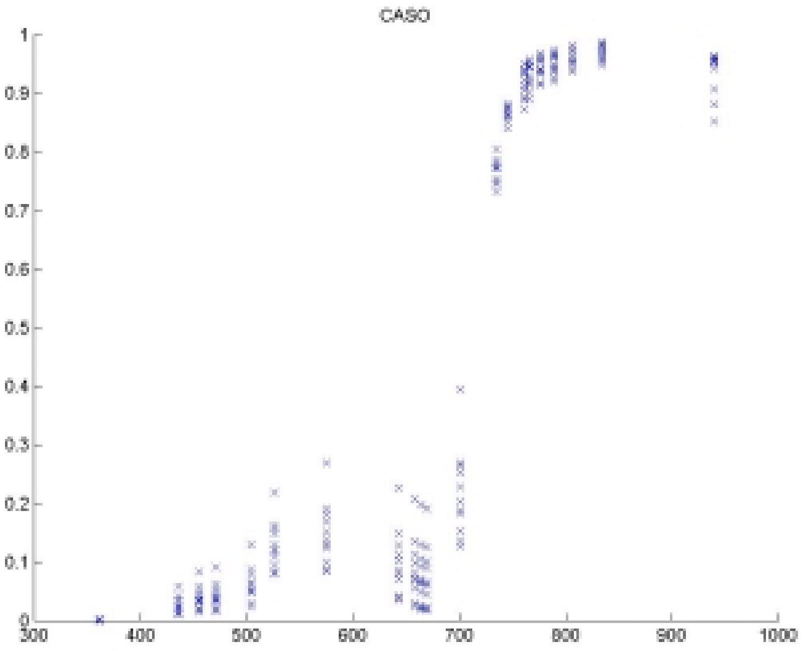
D.16 Trial 16

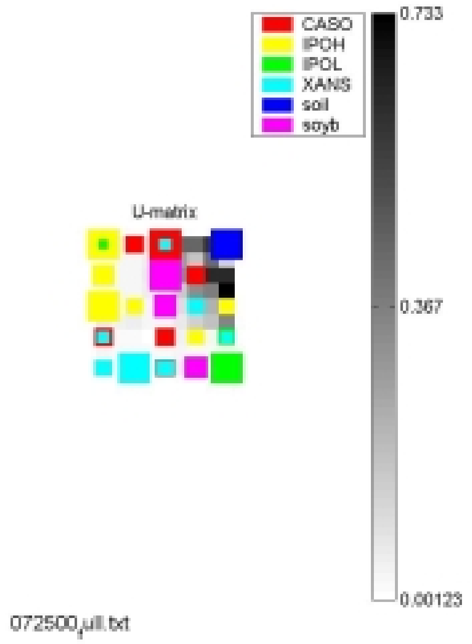


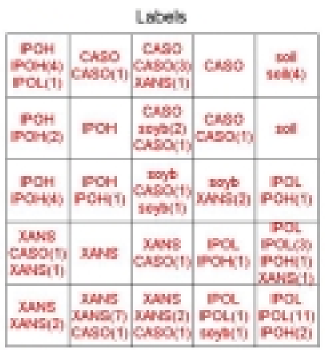


IPOL IPOL(10) IPOH(1)	IPOL IPOL(2) IPOH(1)	soyb IPOL(1)	XANS XANS(1)	XANS XANS(6) CASO(4) soyb(1)
IPOL XANS(2)	CASO	soyb	XANS XANS(3) CASO(1) soyb(1)	XANS XANS(1)
CASO CASO(1)	CASO CASO(2)	CASO CASO(1) XANS(1)	soyb soyb(2)	IPOH
soil	CASO	CASO XANS(1)	IPOH	IPOH IPOH(5) IPOL(2) XANS(1)
soil soil(4)	soil	IPOH CASO(2) IPOH(1)	IPOH IPOH(5) IPOL(1)	IPOH IPOH(3)

D.17 Trial 17

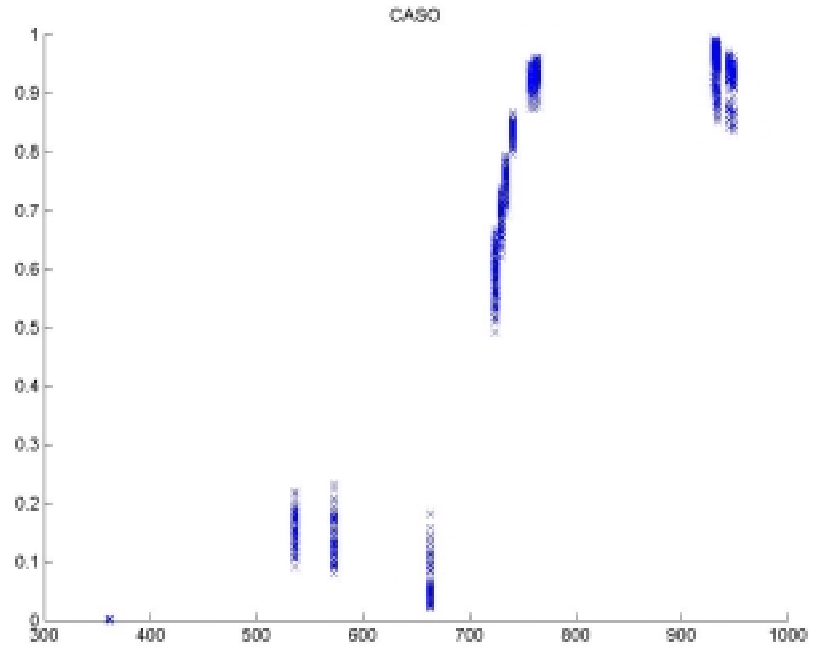


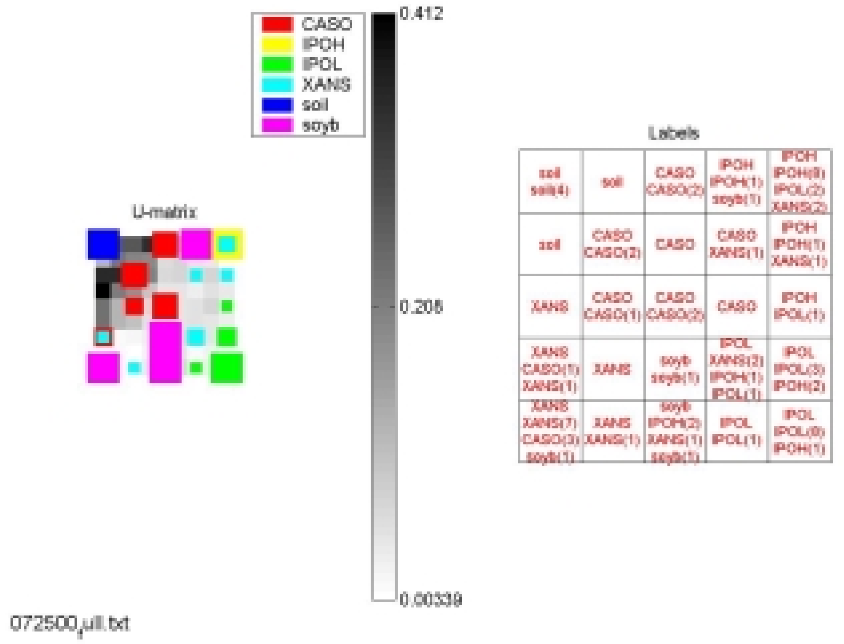



Labels

IPOH IPOH(4) IPOL(1)	CASO CASO(1)	CASO CASO(3) XANS(1)	CASO	soil soil(4)
IPOH IPOH(2)	IPOH	CASO soyb(2) CASO(1)	CASO CASO(1)	soil
IPOH IPOH(4)	IPOH IPOH(1)	soyb CASO(1) soyb(1)	soyb XANS(2)	IPOL IPOH(1)
XANS CASO(1) XANS(1)	XANS	XANS CASO(1)	IPOL IPOH(1)	IPOL IPOL(3) IPOH(1) XANS(1)
XANS XANS(2)	XANS XANS(7) CASO(1)	XANS XANS(2) CASO(1)	IPOL IPOL(1) soyb(1)	IPOL IPOL(11) IPOH(2)

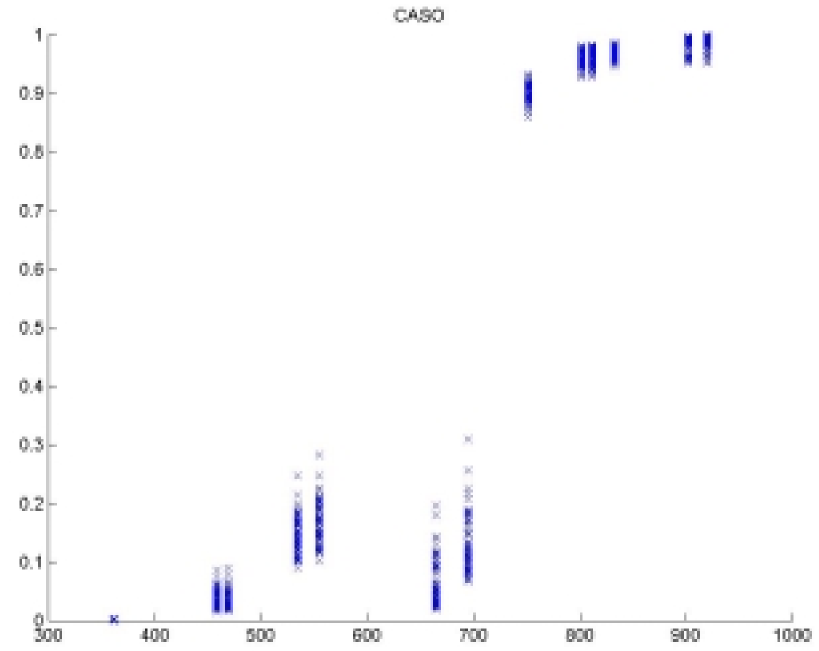
D.18 Trial 18

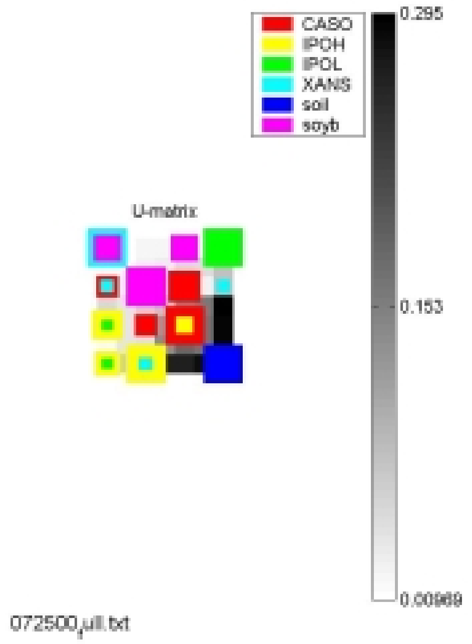


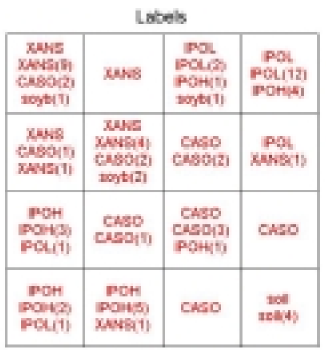


soil soil(4)	soil	CASO CASO(2)	IPOH IPOH(1) soyb(1)	IPOH IPOH(8) IPOL(2) XANS(2)
soil	CASO CASO(2)	CASO	CASO XANS(1)	IPOH IPOH(1) XANS(1)
XANS	CASO CASO(1)	CASO CASO(2)	CASO	IPOH IPOL(1)
XANS CASO(1) XANS(1)	XANS	soyb soyb(1)	IPOL XANS(2) IPOH(1) IPOL(1)	IPOL IPOL(3) IPOH(2)
XANS XANS(7) CASO(3) soyb(1)	XANS XANS(1)	soyb IPOH(2) XANS(1) soyb(1)	IPOL IPOL(1)	IPOL IPOL(8) IPOH(1)

D.19 Trial 19

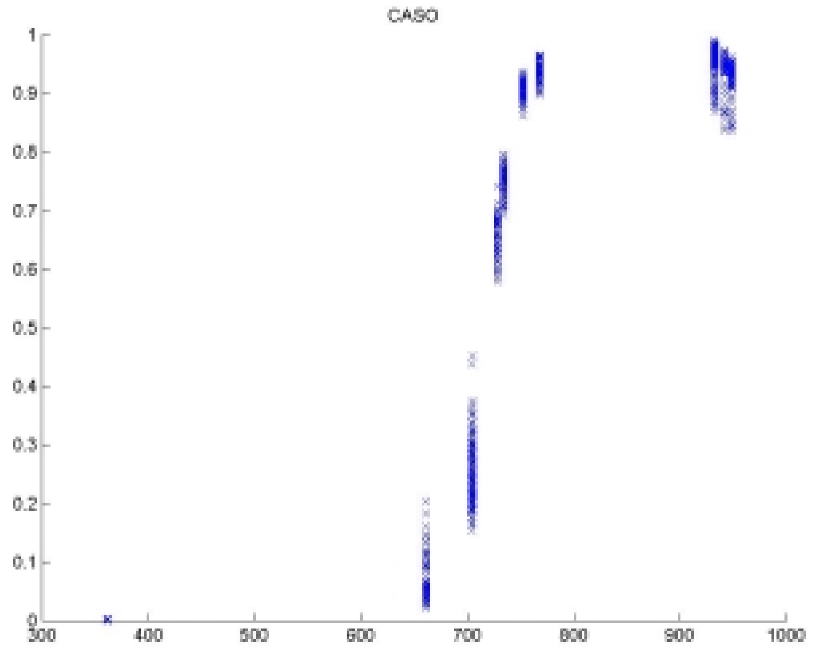


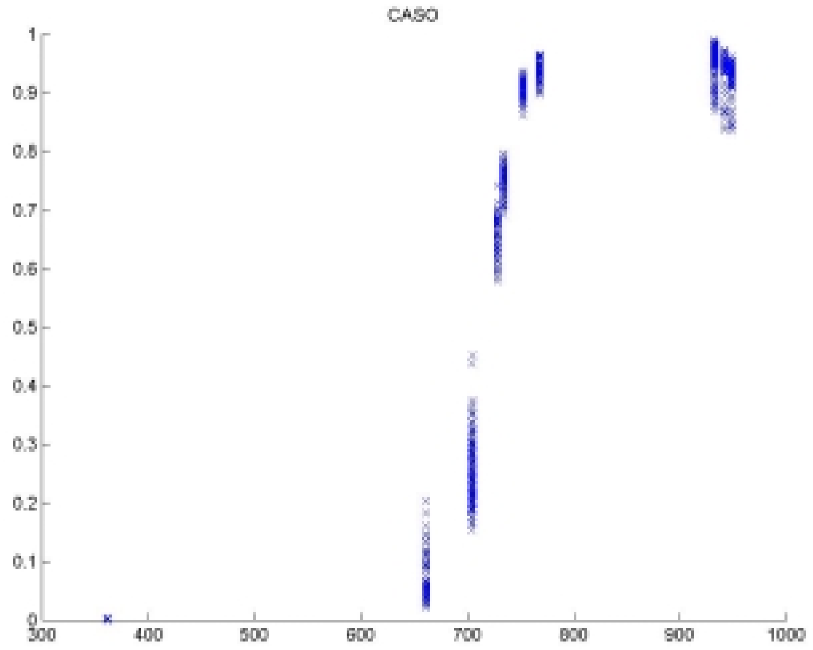



Labels

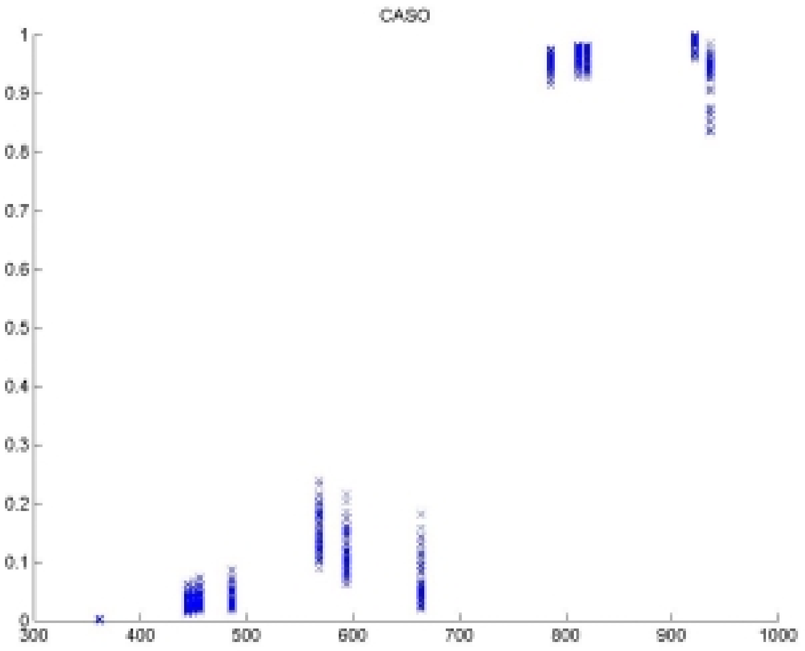
XANS XANS(9) CASO(2) soyb(1)	XANS	IPOL IPOL(2) IPOH(1) soyb(1)	IPOL IPOL(12) IPOH(4)
XANS CASO(1) XANS(1)	XANS XANS(4) CASO(2) soyb(2)	CASO CASO(2)	IPOL XANS(1)
IPOH IPOH(3) IPOL(1)	CASO CASO(1)	CASO CASO(3) IPOH(1)	CASO
IPOH IPOH(2) IPOL(1)	IPOH IPOH(5) XANS(1)	CASO	soil soil(4)

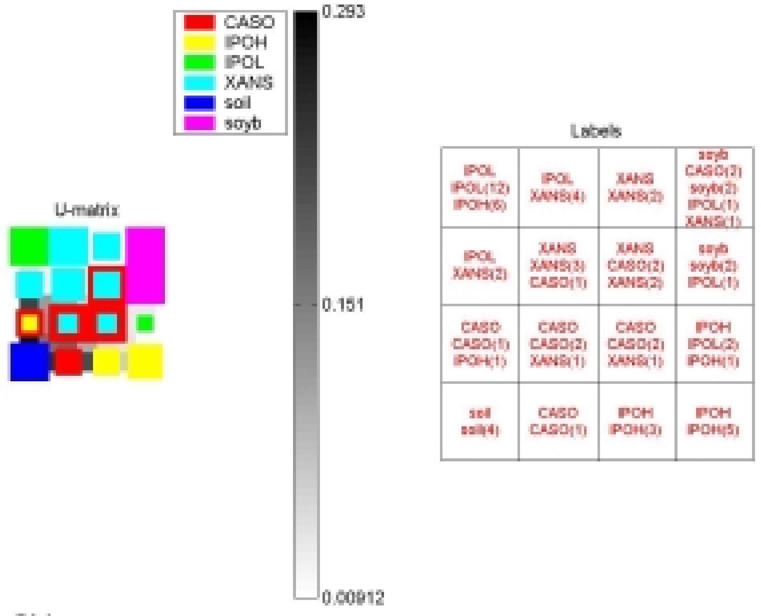
D.20 Trial 20


CASO
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
300
400
500
600
700
800
900
1000


CASO
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
300
400
500
600
700
800
900
1000

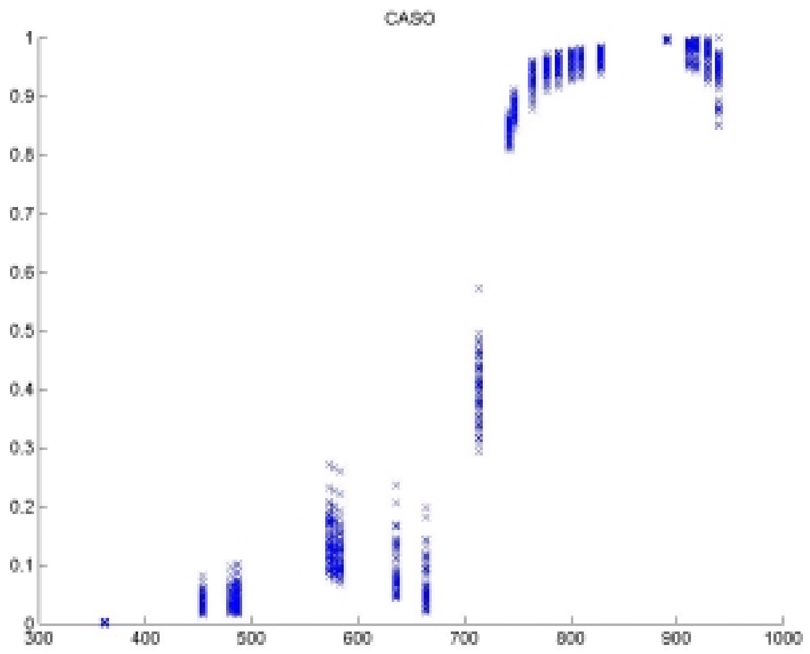
D.21 Trial 21

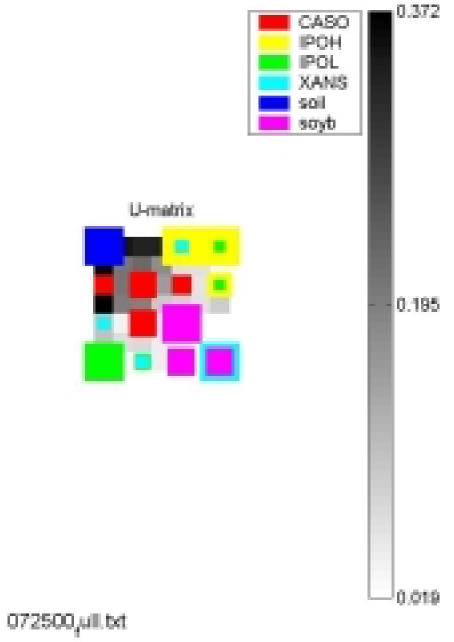


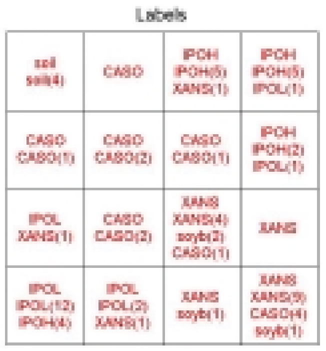


072500_ull.txt

D.22 Trial 22






Labels

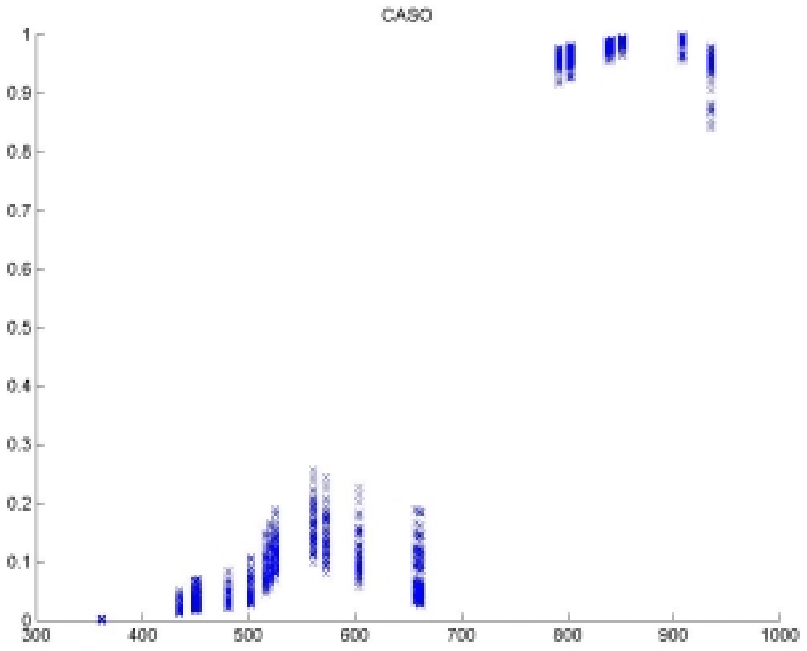
soil
soil(4)
	CASO	IPOH
		IPOH(5)
		XANS(1)
			IPOH
			IPOH(5)
			IPOL(1)

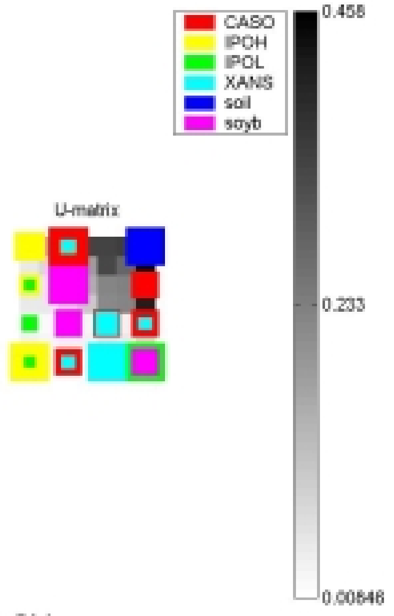
CASO
CASO(1)
	CASO
	CASO(2)
		CASO
		CASO(1)
			IPOH
			IPOH(2)
			IPOL(1)

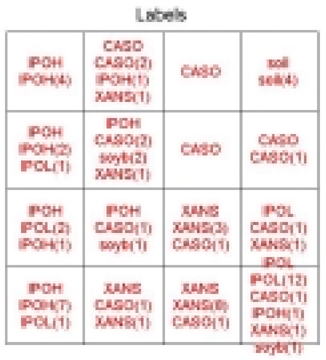
IPOL
XANS(1)
	CASO
	CASO(2)
		XANS
		XANS(4)
		soyb(2)
		CASO(1)
			XANS

IPOL
IPOL(12)
IPOH(4)
	IPOL
	IPOL(2)
	XANS(1)
		XANS
		soyb(1)
			XANS
			XANS(9)
			CASO(4)
			soyb(1)

D.23 Trial 23





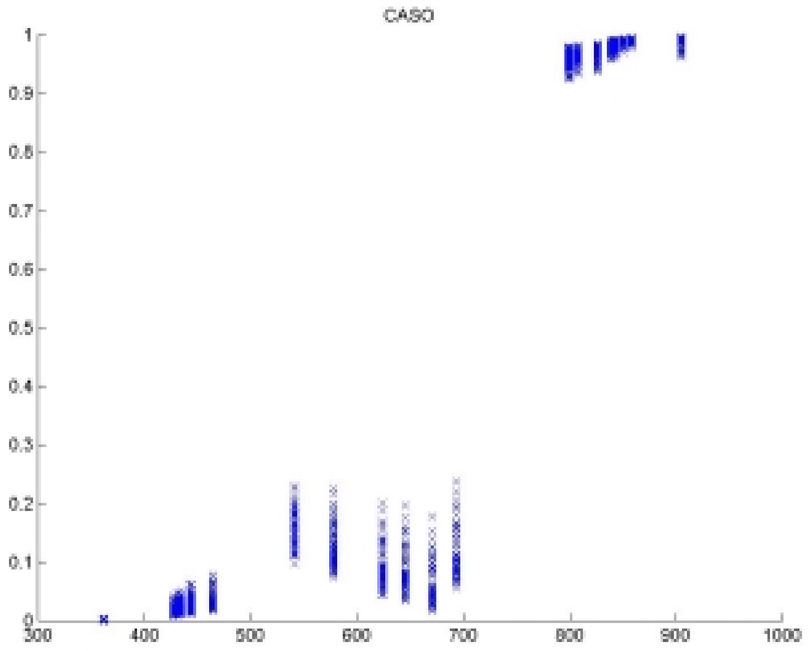


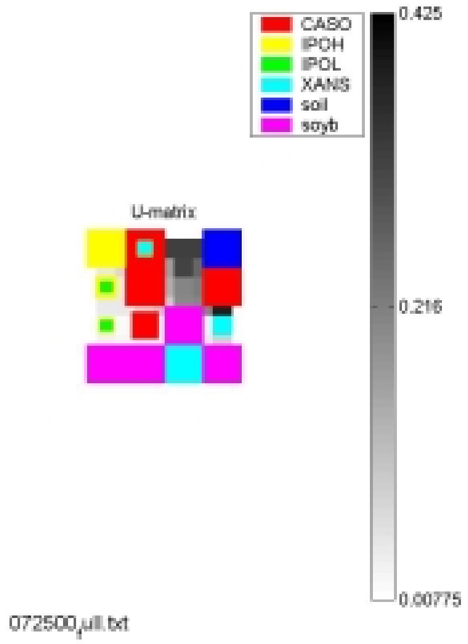
Labels

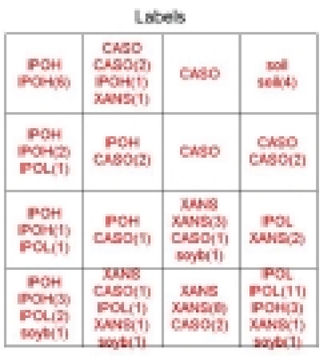
IPOH IPOH(4)	CASO CASO(2) IPOH(1) XANS(1)	CASO	soil soil(4)
IPOH IPOH(2) IPOL(1)	IPOH CASO(2) soyb(2) XANS(1)	CASO	CASO CASO(1)
IPOH IPOL(2) IPOH(1)	IPOH CASO(1) soyb(1)	XANS XANS(3) CASO(1)	IPOL CASO(1) XANS(1)
IPOH IPOH(7) IPOL(1)	XANS CASO(1) XANS(1)	XANS XANS(8) CASO(1)	IPOL IPOL(12) CASO(1) IPOH(1) XANS(1) soyb(1)

072500_ull.txt

D.24 Trial 24







Labels

IPOH
IPOH(6)
	CASO
	CASO(2)
	IPOH(1)
	XANS(1)
		CASO	soil
			soil(4)

IPOH
IPOH(2)
IPOL(1)
	IPOH
	CASO(2)
		CASO	CASO
			CASO(2)

IPOH
IPOH(1)
IPOL(1)
	IPOH
	CASO(1)
		XANS
		XANS(3)
		CASO(1)
		soyb(1)
			IPOL
			XANS(2)

IPOH
IPOH(3)
IPOL(2)
soyb(1)
	XANS
	CASO(1)
	IPOL(1)
	XANS(1)
	soyb(1)
		XANS
		XANS(8)
		CASO(2)
			IPOL
			IPOL(11)
			IPOH(3)
			XANS(1)
			soyb(1)