

Integrating Digitizing Pen Technology and Machine Learning with the Clock Drawing Test

by

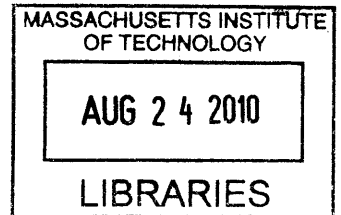
Kristen Felch

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

February 2010

2010 Massachusetts Institute of Technology
All rights reserved.

ARCHIVES



Author _____

Kristen Felch

Department of Electrical Engineering and Computer Science

February 2010

Certified By _____

Professor Randall Davis

Thesis Supervisor

Accepted By _____

Dr. Christopher J. Terman

Chairman, Department Committee on Graduate Theses

Integrating Digitizing Pen Technology and Machine Learning with the Clock Drawing Test

by

Kristen Felch

Submitted to the
Department of Electrical Engineering and Computer Science

February 2010

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The Clock Drawing Test (CDT) is a medical test for neurodegenerative diseases that has been proven to have high diagnostic value due to its ease of administration and accurate results. In order to standardize the administration process and utilize the most current machine learning tools for analysis of CDT results, the digitizing pen has been used to computerize this diagnostic test.

In order to successfully integrate digitizing pen technology with the CDT, a digit recognition algorithm was developed to reduce the need for manual classification of the data collected and maintain the ease of administration of the test. In addition, the Multitool Data Analysis Package was developed to aid in the exploratory data analysis stage of the CDT. This package combines several existing machine learning tools with two new algorithm implementations to provide an easy-to-use platform for discovering trends in CDT data.

Contents

1	Introduction	1
2	Background	3
2.1	The Clock Drawing Test	3
2.2	Digitizing Pen Technology	7
3	The Data Collection Process	9
3.1	Status Quo	9
3.2	A New Algorithm	12
3.2.1	The Initial Split	12
3.2.2	Method One: The ‘Standard’ Case	12
3.2.2.1	Simple Binning Classifier	12
3.2.2.2	Rotational Classifier	13
3.2.2.3	Choosing a Method	14
3.2.2.4	Digit Recognition	14
3.2.2.5	Finalizing the Classifications	16
3.2.3	Method Two: The ‘Nonstandard’ Case	16
3.3	Algorithm Performance	16
3.4	Visual to Numerical Transformation	19
3.5	The Final Data	21
4	The Multitool Data Analysis Package	23
4.1	Introduction	23
4.2	MDA Package	24
4.3	File Creation	25
4.4	Machine Learning Tools	29
4.4.1	Weka	29
4.4.2	DataBionics ESOM	30
4.4.3	Cluster 3.0	30

CONTENTS

4.4.4	BSVM	30
4.4.5	PCA	30
4.4.6	EPCA	31
5	Sample Analysis	33
5.1	Input Overview	33
5.2	Weka- The Decision Stump	34
5.3	ESOM	35
5.4	BSVM	35
5.5	Clustering	37
6	Contributions	41
7	Bibliography	43
8	Appendix	45

1

Introduction

Bringing technology to the medical field is a present-day challenge that attracts the efforts of both researchers and doctors. Recent innovations in software and hardware provide seemingly endless opportunities for the advancement of medical data collection and analysis, but progress is slowed by the necessary caution and regulation surrounding work in the medical field. This thesis addresses the movement of one particular technology into the medical domain, and describes the tools that have been built to support the application.

The technology in question is a digitizing ballpoint pen, which records its location on paper through time-stamped data points; the medical application is the Clock Drawing Test (CDT). The Clock Drawing Test has a long history of success in diagnosing patients with Alzheimer's, Parkinson's, and other neurodegenerative diseases, but is currently administered using paper, pen, and subjective rating scales. Is it possible to improve the accuracy of the CDT through standardization of the data collection and analysis procedures, using a digitizing ballpoint pen rather than the traditional pen?

Software has been developed that interacts with the new pen to record results of the CDT. The Clocksketch program is straightforward, in use requiring minimal training for doctors that use it to administer the CDT. However, in order to be truly embraced by the medical community, CDT administration with the digitizing pen should be easier than traditional methods. My first contribution outlined in this thesis was improving the algorithm that runs behind the scenes of the Clocksketch program, easing the process of data analysis.

A computerized version of the CDT must maintain the ease-of-use of its pen and paper predecessor, its primary advantage over more intrusive/complex diagnostic tests, while also creating opportunities for more rigorous data analysis and standardization of data collection. My second

1. INTRODUCTION

contribution is the creation of a data analysis package, the MultiTool Data Analysis Package (MDA), which was developed to ease exploratory analysis of CDT output data. MDA has evolved to handle any input data of a specified format, but its features and functionality were chosen by predicting the demands of a future analyst working with CDT data.

At this point, the data collected by digitizing pens from the CDT is small in size compared to the expected influx of data over the next few months. However, the successful processing of our limited data using MDA indicates that the tool will be indispensable once data arrives on a larger scale. MDA's chief functionality includes automatic creation of input files for machine learning tools and a common launching platform for these tools. With the most popular machine learning tools at his fingertips, an analyst will easily be able to provide more rigorous analysis of CDT data than that of a doctor prior to use of the digitizing pen.

The digitizing ballpoint pen and supporting software are making their way into the medical domain because of the improvements they can offer on the traditional Clock Drawing Test. Data collection is improved through use of classification algorithms dealing with output of the digitizing pen. Because of the digital nature of the test, the most modern machine learning techniques can be used to analyze output from the CDT and add rigor and standardization to the process. This thesis explains exactly how the data collection and analysis phases of the CDT are improved upon in the new Clock Drawing Test.

2

Background

To understand the advantages of incorporating the digitizing pen technology with the traditional Clock Drawing Test, we look at the history and development of the CDT. We then explore the most current technology for digitizing pens, and how they can be used in a medical context.

2.1 The Clock Drawing Test

The Clock Drawing Test is a cognitive screening tool that has been in use for over 50 years, diagnosing patients with various dementias and other neurological disorders. It is a fairly simple test, requiring the patient to draw two clocks showing a given time. One of these clocks is drawn without any visual references, while for the second the patient is given an image to copy. Previous research has shown that the CDT has high potential for differentially diagnosing Alzheimer's and other neurological disorders. However, the current implementation is a pen-on-paper sketch that is viewed by doctors and ranked based on widely-varying scales. Instructions given to the patient, implementation of the copied clock, and even the ranking scale for clock drawings differ depending on where the test is administered.

The success of the traditional CDT in recognizing neurological disorders even with nonstandardized procedures suggests that it could become a very powerful tool when executed using modern technology. In a 1986 study at the University of Toronto, researchers first attempted to collect data to legitimize the use of the CDT as part of cognitive screening. They compared the results of the CDT to results from two other well-established screens of cognitive function, and found that the CDT could provide a simple and practical method of screening for mental disorder. (Shulman 136)

For this Toronto study, seventy-five subjects were selected from the inpatient and outpatient populations at the University of Toronto. Subjects were over the age of 65, and included subjects with mental disorders and healthy subjects. Three tests were administered to the patients- the

2. BACKGROUND

Clock Drawing Test, the Mini-Mental State Exam, and the Short Mental Status Questionnaire. In addition, a test to assess the subject's level of depression was administered. The MMSE is one of the most widely-accepted cognitive function tests, and is often used as a standard for screening. The SMSQ is a short 10-answer questionnaire that classifies scores of 7 or above as normal. In administering the CDT, subjects were given the outline of a clock and asked to complete it and set the time to three o'clock. The clocks were then analyzed by two independent raters, who ranked the clock with a number 1-5 based on a hierarchical classification of errors. There were about 20 errors that the rankers were asked to look for when assigning a number to the clock. (Shulman 136)

The results of this study showed that the magnitude of error in the CDT is correlated to the error of the MMSE and the SMSQ. In other words, when a subject scores poorly on the CDT he or she is likely to score poorly on the other tests as well. The correlation coefficient between two tests is a measure of how the pattern in results of one test mimick the patterns seen in the other test. The correlation coefficient of the CDT and MMSE was -.65, and that of the CDT and SMSQ was -.66. Correlation is negative since high scores on the CDT indicate impaired cognition while high scores on the other two tests indicate a healthy subject. As a result of this study, the CDT was established as a useful addition to the cognitive tests already in use. Since it is simple to administer, it can provide a quick screening measure before the MMSE is administered. (Shulman 139)

Division of the Clock Test

One of the next advances in the CDT involved dividing the test into two separate parts. In a study at the University of Texas in 1998(Royall 589), it was shown that separate clock tests could be used to differentiate between alternative cognitive impairments. The first test, CLOX1, requires the test subject to draw a clock without any visual guidance. It is meant to reflect the subject's performance in a novel situation, something referred to as executive control, since he or she is presented with only a blank page and asked to draw a clock showing a particular time. Observations were made such as the order that the digits were drawn in and whether or not corrections were made along the way. CLOX2 involves giving the subject a clock to look at and asking them to copy it. This test is aimed more at examining the subject's visuospatial awareness than executive control. Correlation between these two separate CDTs was measured in comparison to MMSE and EXIT25, a common measure of executive control. (Royall 589)

This study determined that both MMSE and EXIT25 results were correlated with results from CLOX1, but only MMSE was correlated with CLOX2. This is as expected, since copying a clock

does not require the same cognitive function as drawing from memory. In a comparative analysis between both CLOX tests and MMSE performance, it was concluded that the CLOX tests were able to correctly identify 81:3% of subjects who had Alzheimer's disease. This is compared to the 89:9% that a combination of EXIT25 and MMSE achieve. When looking at differential diagnosis between Alzheimer's subgroups, CLOX2 scores alone were able to discriminate between the subgroups. Using a combination of CLOX1 and CLOX2, performance on differentiating Alzheimer's subgroups was 91:9% correct. This was a remarkable result, since a combination of EXIT25 and MMSE did not perform as well. The conclusion drawn from this study was that a combination of two clock drawings would be a practical and reliable cognition test to be used by clinicians. (Royall 591-3)

Rating Scales

One of the most notable differences across CDT usage is the scale or rating system that is used. Some tests rate out of 4, others 5 or 10, and some up to 30. Clearly comparisons between facilities, or even between doctors within one facility, would be very difficult with this variation.

The ranking methods for the two studies described above differed in nature and content. For the Toronto study, errors were classified into different levels and the clock was classified to match the highest error level found. The first level included small errors such as drawing the numbers outside rather than inside the circle, or drawing in helper lines to indicate spacing. The middle levels included things such as leaving out numbers or continuing numbering past 12. The fifth and worst level was the inability to make an attempt at drawing a clock. (Shulman 136) Thus, the clocks were ranked from 1 to 5.

The Texas study had rankings that ranged from 0 to 15. Rather than dividing errors into levels to classify the clock, a simple checklist of common errors was used. Each error that occurred was worth one point, totalling 15. Things such as the absence of all 12 numbers on the clockface, as well as the absence of a minute hand and an hour hand, were each worth one point. (Royall 592) In both of these ranking systems, low scores indicate a healthy patient.

Several papers have been written comparing alternate rating systems for the CDT. In one study performed in 2002, 63 clock drawings were rated using 5 different scales and the results compared:

- The Wolf-Klein scoring method assigns 10 points based on the spacing of the numbers.

2. BACKGROUND

- The Sunderland scoring method assigns a maximum of 10 points, but is based on the placement of the hour and minute hands.
- The Shulman method was the one used in the Toronto study, but was modified to have 6 categories of error rather than 5.
- The Clock Drawing Interpretation Scale (CDIS) awards a maximum of 20 points for the circle, hand placement, and presence and placement of numbers.
- The Glass method assigns a score of 0 if five main aspects are correct- circle formation, numbers aligned, all numbers included, hands aligned, and both hands identified. Half of a point is added if one of the above is partially correct, and one point if it is incorrect. (Richardson 170)

The conclusion of this study was that each of the scoring methods had significant correlation with the MMSE test, with the Sunderland scoring criteria having the highest correlation. More importantly, it was discovered that different classification methods worked better for diagnosing different diseases. With patients who had Alzheimer's disease, the Glass method had the highest correlation with MMSE. For patients with mixed dementia and multiinfarct dementia, the CDIS scoring criteria yielded the highest correlation with MMSE.

The study concluded that patients with mixed dementia are more likely to make time-setting mistakes than number-spacing errors. Therefore, they score better with the Glass and Shulman tests that do not depend on the clock hand placement.

These results suggest that a standardized method of classification that combines all of the measurements would help to differentially diagnose the different types of neurogenetic diseases. (Richardson 171-2)

A similar investigation conducted in Australia in 2001 assessed different scoring criteria by integrating the area under the ROC curve. The ROC curve is created by mapping the relationship between the number of true positives and false positives for a given scoring criteria. A high area, between .7 and .9, indicates a useful criteria because it means that a high percentage of positives can be correctly identified without incurring a large number of false positives. The scoring methods that were examined in this study were those proposed by Mendez, Shulman, Sunderland, Watson, and Wolf-Klein. The Mendez criterion has 20 frequent errors that are assigned points, similar to the CDIS. The Watson method scores out of 7 based on number placement. The study showed that

the Shulman and Mendez methods give the largest area under the ROC curve, and predict dementia more accurately than the Sunderland and Watson methods. The Schulman method had both high sensitivity and high specificity, while the Watson method had high sensitivity but low specificity. The researchers conclude that the Schulman method scores the best, but that CDTs should still not be used alone to diagnose dementia. Until a standardized scoring criteria is developed and much more testing is done, CDTs are not reliable enough to be used as stand-alone tests. (Storey)

One such method of standardizing the CDT is use of a digitizing ballpoint pen. If a standard procedure for collecting data can be established through distribution of a pen and supporting software, then CDT results across locations can be compared and analyzed. With more data available, analysts will have a better sense of the comparative diagnostic value of the CDT. The next section will describe the digitizing ballpoint pen that will be incorporated into the CDT.

2.2 Digitizing Pen Technology

In order to digitize the process of drawing, two components are necessary- the digitizing pen and special paper upon which to draw.

The Paper

Special paper is required in order for a digitizing pen to be able to store information about the image being drawn. This paper uses the Anoto pattern, which provides a simple but very powerful mechanism that identifies which physical page a pen is on as well as the writing and drawing movements made on the page. The pen would not be able to record anything without the pattern, and it is the pattern that makes it possible for the pen to know what you write and where you write.

The Anoto pattern consists of small dots (100 microns in diameter) arranged with a spacing of approximately 0.3 mm on a square grid. The dots are slightly displaced from the grid, each dot in one of four possible positions. The pattern is visible on plain paper, but faint enough so as to not interfere with drawing.

The Pen

We used the the Maxell DP-201 (R4.1) digitizing pen. This pen has a built-in camera, which takes digital snapshots of the pattern as the pen moves over the paper. The pen uses an infrared LED (light-emitting diode) to take the snapshots between 50 and 100 times per second. The camera

2. BACKGROUND

can determine its position only on paper with an Anoto pattern. The pen processes the snapshots by calculating its position on the page, and stores the snapshots in the form of pen stroke data (a set of time-stamped coordinates). (Anoto(A) 22)

With the combined capabilities of a digitizing pen and the Anoto patterned paper, it is possible for the CDT to be recorded in the form of time-stamped points. Armed with this capability, we are ready to see how the data analysis stage of the CDT has been revolutionized by the digitizing pen technology and supporting software.

3

The Data Collection Process

In order for the digitizing ballpoint pen to be successfully integrated into the Clock Drawing Test, it must provide advantages during administration of the test. If it is more difficult or time-consuming to administer the CDT using a digitizing pen, doctors will be hesitant to make any changes to their time-proven approach. For this reason, the current software that supports the pen must be improved to have higher digit recognition rate so that less input from doctors is required.

3.1 Status Quo

The digital Clock Drawing Test is administered in much the same way as the traditional CDT, by asking the patient to draw two separate clocks. The patient uses the Maxell digitizing pen, and draws the clocks on paper containing an Anoto pattern. A few changes have been made to the administration process to reflect use of the digitizing pen.

The test paper is divided into a right and a left half, and the left half is covered for the first portion of the test. The test administrator places a check mark in the upper right corner of the page to indicate that a new test is starting, then hands the pen to the patient, who draws the first clock on the (empty) right half of the page. The left half of the page is then uncovered to reveal an image of a clock. The patient copies the image on the left half of the page, and the administrator places a check mark in the bottom right corner of the page to indicate completion.

The software that supports use of the digitizing pen with the Clock Drawing Test is called the Clocksketch program. Once the patient has completed the test, the pen is returned to its base, which is connected to the computer. It is possible to run tests on several patients before the pen is connected to the computer and data is transferred. The maximum amount of data that the pen

3. THE DATA COLLECTION PROCESS

can store is that from approximately 500 patients.

Once the pen has been connected to the computer, the Clocksketch program will automatically launch. The first test will be loaded, and images of the two clocks will appear in the center panel. The doctor will enter information including patient name, date of birth, sex, MMSE score, doctor name, and scorer name. Once this information is entered, the doctor must save the test before it can be re-opened for analysis. If there is more than one set of data stored in the pen, the doctor needs to enter patient information and save each individually. Files are saved with a .csk extension. Figure 3.1 shows the appearance of the Clocksketch program when the pen is first connected to the computer.

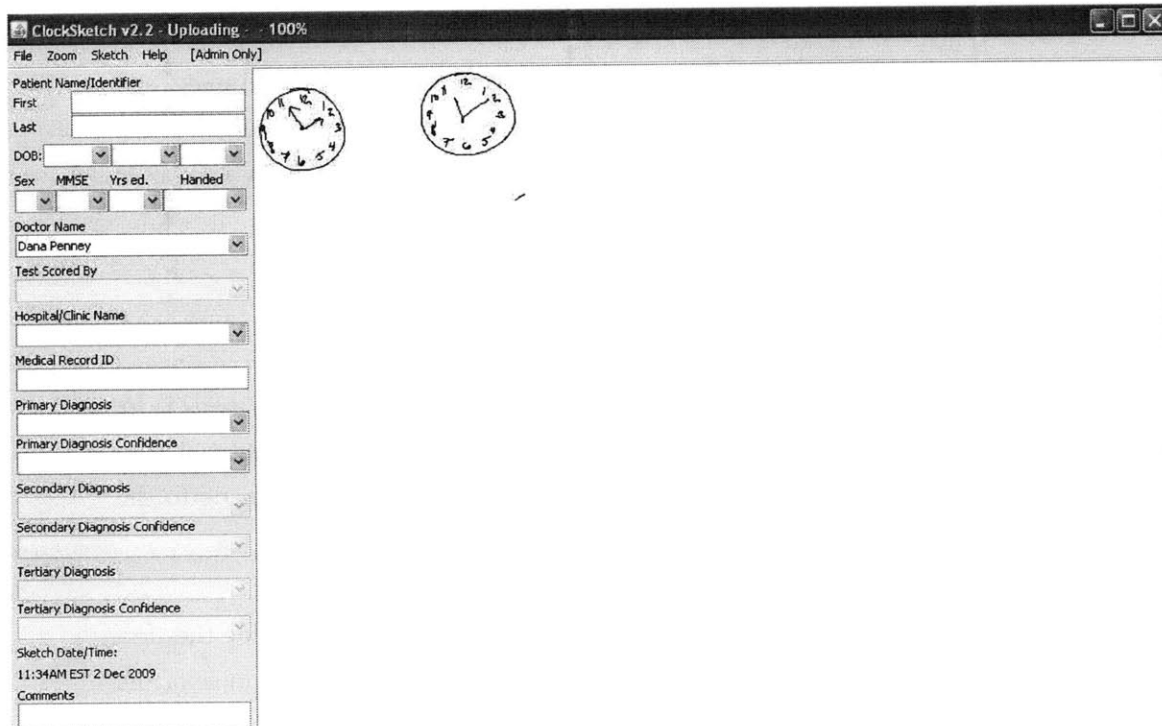


Figure 3.1: Appearance of Clocksketch Program Upon Startup;

After the doctor has saved the original data files, they can be reopened using the Clocksketch program for analysis. The .csk file contains information about each stroke drawn by the patient, but strokes have not been interpreted or assigned to the correct clock. There is no information about location of particular digits or hour/minute hands.

The Clocksketch program currently uses a simple method for classifying the different components

of the clock. First, the outline of the clock is found by looking for the largest continuous stroke drawn by the patient. Then, the location of the center of the clock is found by fitting a circle to the clock outline and finding the center of this circle. From here, the circle is divided into “bins,” resembling slices of a pie. There are 12 bins, and each is expected to contain one number on the clock. For example, the bin directly above the center of the clock is expected to contain the 12. Strokes are assigned to a particular number according to the bin that they fall into. Classified strokes are surrounded by tan boxes to indicate that they have been classified. After the numbers are found, the hour and minute hands are recognized as the strokes that come closest to the clock center.

Another feature of the Clocksketch software is the ability to play a movie of the drawing, showing how it was drawn (rather than just the final appearance). The movie can be played slower or faster in order to make it easier to observe different phenomena. The movie can be played back at any time, including long after the test was taken, giving a physician reviewing the medical record a chance to “look over the patients’ shoulder,” seeing what he/she would have seen when the test was being given. Hence all the hesitations, false starts, error corrections, perseverations, etc., are preservable along with the final appearance of the clocks. (Penney 4)

The current method for identifying the elements on the clock is highly dependent on their correct placement. Given the nature of the CDT, many of the clocks that doctors encounter will be far from perfect. If the patient begins numbering the clock in the wrong place, the program will classify each digit incorrectly. Similarly, the current method will give incorrect classifications if the numbers are clumped together rather than spread evenly along the circumference of the clock. If this situation arises, it is necessary for the doctor to correct the mistakes that the program made. This involves classifying each digit by hand by placing it in the correct folder, which can be a tedious process.

If the classification process is too difficult for clocks that are even slightly abnormal, the advantage of using the digitizing pen is lost. For this reason, a more accurate classification algorithm is needed. This algorithm should take into account not only spatial information about the strokes in the drawing, but also the temporal information that is available. It is also possible to measure the accuracy of digit classification by comparing the digits that are found to sample drawings of each number. With a better classification algorithm, doctors will need to make far fewer adjustments and the data collection process will be sped up immensely. The next section will describe a new algorithm that I have developed to achieve this goal.

3. THE DATA COLLECTION PROCESS

3.2 A New Algorithm

In order to improve classification of the elements on the face of a clock, spatial and temporal factors are used. The algorithm that I developed chooses from two different methods to classify the clock; the first is used if the layout of the digits is fairly circular; the second is used if the digits are not laid out as expected (clumped to one side, for example). In the first case, timing and angle information is used in a similar, but more complex, manner as the original algorithm. In the second case, the algorithm relies only on digit recognition to pick out numbers. Each stage of the algorithm is described below.

3.2.1 The Initial Split

In order to get a general sense of how the digits are laid out on the face of the clock, the algorithm looks at the shape that the strokes make. The Clockface stroke (the outer circle) is removed from consideration, and we similarly remove the hands by removing any strokes that come close to the clock center. We then extract the center of mass of each remaining stroke, combining these centers into one stroke, and determine the ellipticity of the shape that these points make. If the ellipticity falls below a certain threshold, indicating a fairly circular arrangement of digits, we will use the first method of classification below to classify the digits. High ellipticity indicates a non-standard arrangement of the digits, and thus the second classification method is used.

3.2.2 Method One: The ‘Standard’ Case

If the layout of the digits is judged to be fairly circular, then we proceed with a classification method based on angular location of the strokes and the relative times at which they were drawn. This method is actually a combination of two methods, several additional procedures that correct mistakes, and a final procedure that determines which of the methods achieves a better classification. The two methods are referred to as the Binning and Rotational Classifiers.

3.2.2.1 Simple Binning Classifier

This method closely resembles the method used in the original algorithm for finding digits on the face of the clock. It assumes that the digits will be drawn near their correct absolute locations on the face of the clock, and therefore looks at specific angles for each number. However, it is slightly more sophisticated than simply dividing the clock into pie-slice bins.

The first stage of the Binning Classifier divides the clock face into 12 bins, but only categorizes those strokes that fall very close to the center of the bin. For example, in order to be considered

part of the 12, a stroke will have to appear extremely close to directly above the center of the clock. In the original algorithm, each bin had a range of 30 degrees, to total the 360 degrees around the clock. With the new version, strokes must fall within 5 degrees of where a number should fall in order to be classified. This prevents numbers that are slightly out of place from immediately being placed in the incorrect bin.

After this initial assignment is complete, strokes that are touching classified strokes are given the same classification as the stroke they are touching. This allows for classification of strokes that are not within the narrow margin described above, but are very likely part of a classified number. A possible error could be made in this stage if several numbers overlap, but this is accounted for in the next stage.

Finally, adjustments are made based on the number of strokes that we expect for each digit. For example, we expect the 8 to contain 1-2 strokes. If the digit has been assigned more than the expected number of strokes, we remove the stroke that is farthest away from the center of mass of the bin of that digit. After these strokes have been removed from each digit, we add the now unclassified strokes to digits that have fewer than the expected number of strokes. In the end, we expect to have a reasonable number of strokes assigned to each digit and to have found those strokes closest to where each digit should be located.

This simple binning classifier is the best performer when there are digits missing from a clock face. Since it is based on relative location of the strokes to pre-determined bins, correct classification of one stroke is independent of classification of other strokes. However, this classifier is the weakest performer when a patient draws the 12 digits with a non-standard placement. For example, if they start by drawing the 12 in the 1's position and continue around the clock, each classification will be off by one number. For this reason, we have a second classifier called the Rotational Classifier.

3.2.2.2 Rotational Classifier

The Rotational Classifier takes a different approach to classifying the digits on a clock face. The insight that led to creation of this classifier is that patients may draw the numbers on the clock in the correct order, but not begin in the correct location. Or, they may start in the correct location but finish numbering before they've come full circle. In either of these cases, the Binning Classifier would get a majority of the classifications incorrect.

3. THE DATA COLLECTION PROCESS

The Rotational Classifier begins by creating a set of all of the strokes that are not touching the center of the clock. This is to avoid classifying the hour and minute hands as digits. Then, the set of strokes is sorted by the angle of the center of mass of each stroke. Once the strokes are in a sorted list, the next step is to combine nearby strokes that are likely to be part of the same digit. This is done by repeatedly looping through the list and combining the two strokes that are nearest to each other, until there are 12 remaining strokes.

Once the list is of length 12, we use nearest neighbor classification to determine which stroke belongs to the number 8, based on a database of 60 samples of each number. This technique is described in further detail below, but we chose to look for the number 8 because it is the most easily differentiable from the other numbers. Once we have found the 8, we assign the numbers 1-12 based on the location of the 8, based on the assumption that the patient drew the numbers in order around the clock.

The obvious disadvantage to using this method is that it will not work if the patient drew the numbers on the clock out of order. If there are any numbers missing, the Rotational Classifier will get only the numbers between the 8 and the first missing number correct. However, this method performs better than the Binning Classifier when the patient has all of the correct numbers on the clock, but not exactly aligned with the expected angles. Since we don't know ahead of time whether the Binning or Rotational Classifier will perform better, we use digit recognition to choose between them.

3.2.2.3 Choosing a Method

After both the Rotational and Simple Binning Classifiers have been run on the clock data, we score the results of each classifier using nearest-neighbor matching on each of the digits. We then count the number of digits whose classification matches the bin in which it was placed, and choose the classifier that gets the higher score.

3.2.2.4 Digit Recognition

In order to determine whether a stroke has been placed in the correct bin, we need a method for assigning a digit to an unclassified stroke. This is commonly known as the digit recognition problem, and there are many methods for solving it. Nearest-neighbor classification, support vector machines, and neural networks are the most common methods found in digit classification literature. Several studies have compared performance of different machine learning algorithms

on the digit recognition problem.

One study has been particularly useful to me in beginning work on digit recognition, done in 2005 by Levent Kara of Carnegie Mellon University and Thomas Stahovich of the University of California. They developed a method of classification that combines four separate template-matching classifiers and uses these results to classify the unknown. All four classifiers are based on a down-sampled version of the image, called the template.

The unclassified image goes through 4 stages: pre-processing, template matching, rotation handling, and classifier combination. In the preprocessing stage, the image is down-sampled to a 48 by 48 square grid. In the template-matching stage, four different distance formulas are used to determine the similarity between the test sample and each of the training samples. The first is the Hausdorff distance, which finds the largest minimum distance between a point in the test sample and one in the training sample. This number marks the furthest distance between the two samples. The second formula is a modified version of Hausdorff distance, which finds the average of minimum distances rather than the maximum. The third formula involves counting the number of black pixels in each of the samples and their overlap. The Tanimoto coefficient is defined as the number of black pixels that overlap between the two images, divided by the sum of the number of black pixels in each image minus the overlapping ones. Since this is based solely on black pixels, a complementary coefficient can be used to compare white pixels. Finally, the Yule coefficient takes into account overlap between both the black and the white pixels.

The last two methods for classification are subject to large error if the digits are rotated even slightly. Therefore, one stage of the classification involves taking rotational invariance into account. In this study, researchers developed a method of rotating the square grids by varying degrees and finding the closest match at each rotation. In order to combine these four techniques, the distances are normalized. Then, the classification found that corresponds to the smallest distance is taken as the test sample's classification. The use of the down-sample square grid, as well as the rotational technique developed here, were helpful in my own development of a digit recognition system. (Kara)

I developed a method of digit recognition that combines the preprocessing steps in this study with a simpler nearest neighbor classifier. I used a downsampled 15 by 15 square grid as a feature vector, similar to the method used above. I thickened the data that was received from the pen, and rotated the symbols by small increments in each direction to adjust for digits that are not drawn perfectly straight up and down. I did a study of which methods worked best on my data,

3. THE DATA COLLECTION PROCESS

and discovered that an SVM using an RBF kernel and the nearest neighbor method both work rather well. When calculating the error rate for each method, I used 10 fold cross-validation for the SVMs and leave-one-out cross-validation for testing the nearest neighbor method. Since the nearest neighbor method is quicker and simpler, I chose this method for use with the clock drawing test. With a large enough dataset, error rates drop to about 3%.

3.2.2.5 Finalizing the Classifications

The final stage in classifying the digits on the clock in method one is using digit recognition to choose between the Rotational and the Simple Binning Classifier. The combination of these two classifiers works well for clocks whose digits are drawn in a relatively circular fashion. However, a much different approach must be taken when this is not the case.

3.2.3 Method Two: The ‘Nonstandard’ Case

When the digits are not drawn on the clock face in a circular pattern, it does not make sense to use the Binning or Rotational Classifiers. The classifier for this case is much simpler, and depends on the relative timing and location of each of the strokes. There is a threshold for both time and space, and if two strokes are close enough to each other in both domains they are combined together. Once all combinations are made, the nearest neighbor digit recognition technique is used to assign a number to each set of strokes.

There are both benefits and drawbacks to this method. The first is that we could end up with more than one of a given digit. This could also be seen as an advantage, because in some cases the patient will actually draw more than one of the same digit. Another drawback is that this method does not take advantage of the relative locations of the numbers. It does not assume that the 4 will be closer to the 3 than the 9, because if the numbers are not drawn in a circle we do not make any assumptions about their actual layout. This method was shown to perform much better on poorly drawn clocks than the Rotational and Binning Classifiers.

3.3 Algorithm Performance

The new algorithm for digit classification is much more accurate than the existing one. The new algorithm also performs slower, due to the nearest neighbor matching. The timing of the new algorithm is about 15 seconds for the first clock, and then 5 for any subsequent clocks that are classified while the program is still running. This difference occurs because the sample digit data for nearest neighbor classification must be read in and preprocessed. Below are some comparisons

of the performance of the two algorithms, so that the improvement in classification can be seen.

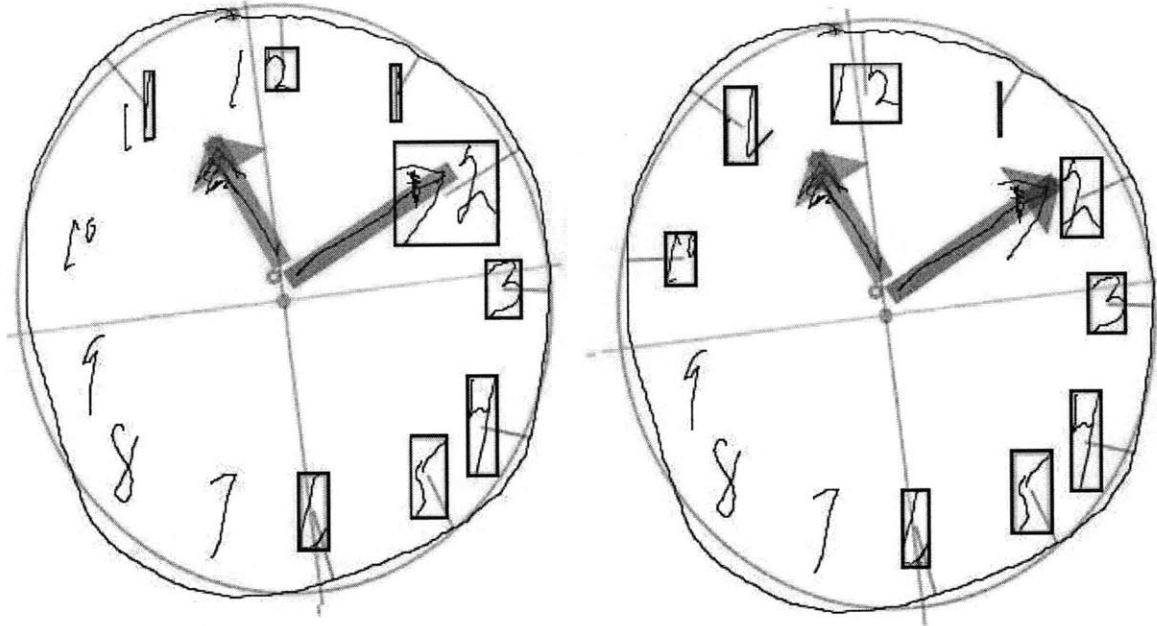


Figure 3.2: Sample One;

In the first sample, neither the old(left) or new(right) algorithm gets all of the numbers correct. However, the new algorithm classifies the 10 correctly where the old method did not find the 10. In addition, the old algorithm found only one digit of both the 11 and the 12. The new algorithm found both digits, because it looked for more strokes when it did not initially find the minimum number expected (in the case of the 11 and the 12, the minimum is 2). Finally, the arrow on the minute hand was classified as part of the 2 in the old algorithm. This mistake was not made with the new algorithm.

In sample two, the old(left) algorithm classifies only half of the numbers correctly. The new(right) algorithm does much better, missing only 3 of the numbers. The Rotational Classifier was able to recognize the sequence of numbers from 5 to 12, even though the individual numbers are not drawn in the correct locations. The three errors are a result of the 2 being classified as part of the minute hand, because it overlaps with the arrowhead. For this reason, the 3 is classified as the 2, the 4 as the 3, and the top of the 5 is called the 4. The hour hands are not found in either case, but the new algorithm is able to correctly identify the minute hand.

As in the previous samples, the old(left) algorithm does not find all of the numbers on the clock in sample three. However, in this case the new(right) algorithm correctly classifies all of the dig-

3. THE DATA COLLECTION PROCESS

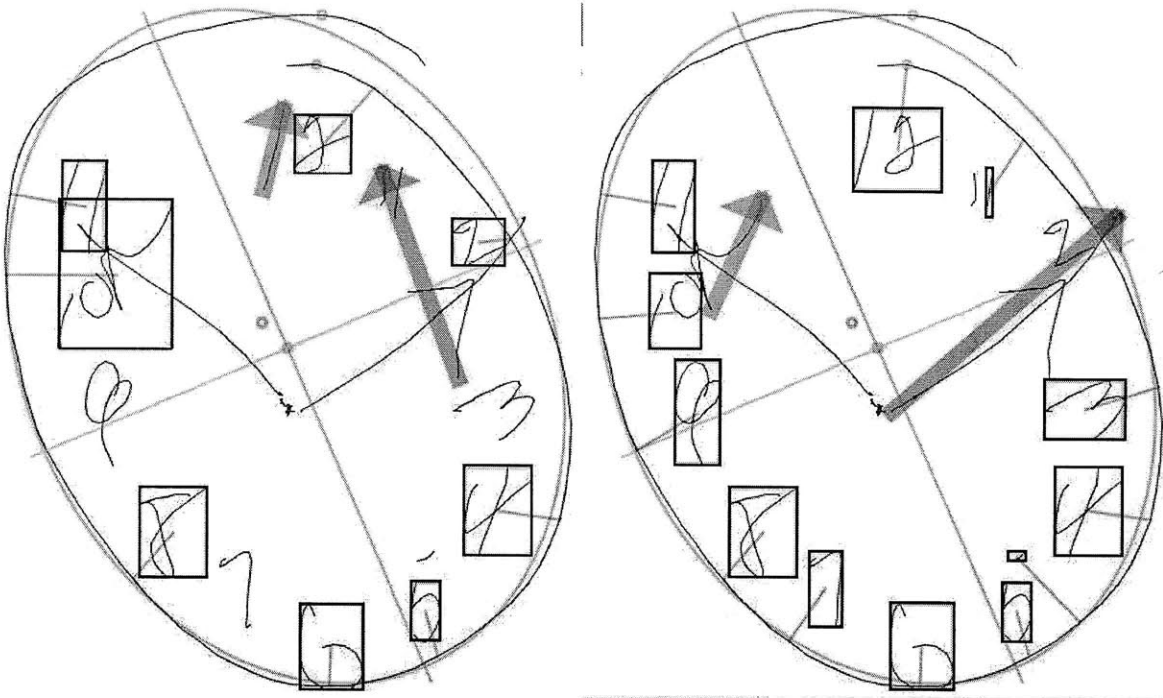


Figure 3.3: Sample Two;

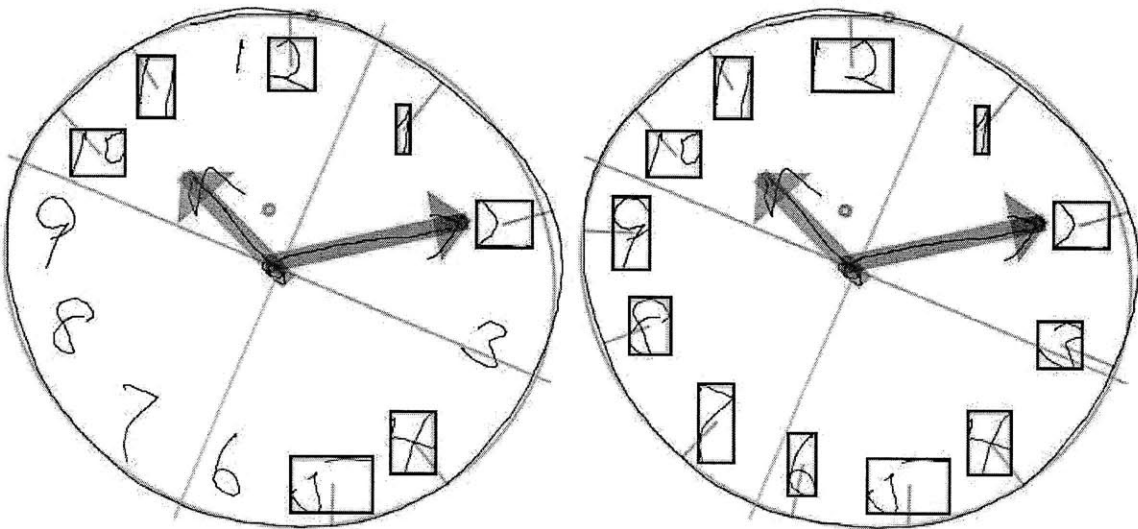


Figure 3.4: Sample Three;

its. Despite the fact that numbers are shifted slightly from their expected angles, the Rotational Classifier can still recognize them because they do not overlap and are drawn clearly.

Over all, digit classification is improved tremendously. Administration of the Clock Drawing Test should now only require a few small fixes by doctors, rather than classifying each digit by hand in the worst cases. We now move on to discuss the transformation of this visual data to spreadsheet format.

3.4 Visual to Numerical Transformation

After the elements on the clock face have been classified correctly, as approved by the doctor, the file is saved in a .csk format. The next step is to export the data to an .xls file, which will record all of the clock data in numerical format. The Clocksketch program has the ability to make certain measurements once the digits are all identified, including measures of time, distance, and angle. The doctors and researchers leading the CDT project have decided upon a list of features that should be recorded, agreeing that these features have potential diagnostic value. The data analysis stage will attempt to pick out specific valuable features, but in the data collection stage our goal is to maintain all of the potentially important data without losing any information. The information recorded is divided into 10 categories.

Patient Information

This section contains information identifying the patient. It includes name, facility, medical record number, diagnosis, gender, handedness, and other facts collected from the patient not related to the CDT. Some of these factors, such as gender, may turn out to be critical pieces of information for diagnosis. However, for the most part, this data is mainly used for identification purposes.

Drawing

The Drawing category contains only 3 measurements. The first is Drawing Type, which is either “command” or “copy”. For each patient, there will be one page of the .xls spreadsheet dedicated to information about the command clock and another to the copy clock. This feature identifies which clock the data represents. The other two features are number of strokes and total time, both measured for the entire clock. Although these are simple measurements, we expect that they are critical to differentiate our healthy patients.

Clockface

This section records all data related to the Clockface stroke, the outer circle of the clock. Measurements include number of strokes, total time, and speed. For the speed measurement, the clockface

3. THE DATA COLLECTION PROCESS

is divided up into quartiles and four different speeds are recorded. The goal of this approach is to determine if the patient slows down or speeds up during certain parts of the circle. Other features include measurements of symmetry, horizontal and vertical diameter, and location of the center. “Closedness” measurements indicate whether or not the patient was able to complete the circle, and how close their ending point is to their starting point.

Hands

The next two sections are dedicated to the minute hand and the hour hand. Each records total length, time and number of strokes. In addition, the center of mass is recorded as well as distance from the center of the clock. Direction of drawing is recorded for both the hand and the (potential) arrow on the end of the hand, both of which are either “outward” or “inward.” Finally, the angle of the hand is recorded in addition to the difference of this angle from the ideal angle of the hand.

Hand Pairs

The Hand Pairs section contains information about the relationship between the minute hand and the hour hand. Measurements include x and y intersection, as well as the angle between the two hands.

Numbers

There are two parts to the Numbers section- an overall numbers analysis and specific details about each number. The Numbers Analysis section contains average width and height of the digits as well as the number of digits missing. The next section records information such as the number of strokes, total time, and length of each individual digit. The center of mass of each digit is recorded as an x and y coordinate, along with the height and width of the digit. An OutsideClockface feature records if the digit was drawn inside or outside of the Clockface Stroke, because some patients will be unable to place digits correctly. Finally, the absolute angle of the digit, the difference from ideal angle of the given digit, and the distance from the circumference of the circle are recorded. We predict that comparing the placement of specific digits on the face of the clock will be a key factor in differential diagnosis.

Hooklets

Hooklets are abrupt changes in direction at the end of a stroke. When drawing the numbers on a clock, it is a common phenomenon for subjects to leave a “tail” on one digit that points in the direction of the next digit to be drawn. This is because the pen has not been fully lifted off of the

paper before the hand moves towards the next digit. An example of a hooklet can be seen in Fig 3.5.



Figure 3.5: Sample Hooklet;

Information such as the length, slope, and speed of each hooklet are recorded, as well as which digit they are part of and the distance to the following digit. It is predicted that the presence of hooklets will indicate a foresight or planning capability in the patient, that of planning the next digit while they are still drawing the current one, that will be absent in patients who have certain mental deficiencies.

Non-Standard Symbols

This section contains information about symbols that are only present in some patients' drawings. These include a center dot for the clock, tick marks along the circumference, text that has been written on the page, and any digits over 12 that are written. The presence or absence of certain extra features may be indicative of specific mental deficiencies.

Latencies

The final category of features that is recorded for each clock is Latencies, or lag time between different parts of the sketch. Latencies are recorded between different parts of the clock, such as the Clockface and the hands or the hands and the center dot. The inter-digit latency is also recorded, which is the average time that the patient takes between digits. Large latencies indicates a long pause between each part of the drawing, which could be indicative of different mental conditions.

3.5 The Final Data

A third sheet in the .xls file contains a long list of features for the patient. All of these features come directly from the measurements above, and some are derivative of these measurements. For example, this final list has space allotted for 9 hand pairs. If information about only one hand

3. THE DATA COLLECTION PROCESS

pair was recorded above, then the remainder of the entries will be filled with zeroes.

The final list of features was agreed upon in a joint effort between doctors and researchers. The goal is to include all features that could be potentially useful in diagnosis, losing as little information as possible between the sketch and the database. Currently there are 1017 features that are recorded for each clock, making a feature vector of about 2000 for each patient. The next step is to analyze our data and discover which of these features can be used to differentially diagnose neurodegenerative diseases. How do we deal with data so large, and figure out which of these features actually have diagnostic value? This is the question that will be addressed in the next section.

4

The Multitool Data Analysis Package

4.1 Introduction

After the data collection stage, we are left with feature vectors approximately 2000 features in length for each patient. These vectors contain spatial and temporal information about both the command clock and the copy clock, as well as other information such as patient gender and age. How do we take this large set of data and find common trends? Is there, for instance, a group of features that take on extreme values in patients with Alzheimer's disease as opposed to patients with Parkinson's?

This problem calls for the use of data mining and machine learning tools such as clustering, dimensionality reduction, and self-organizing maps. A large number of tools are available today that provide the functionality of machine learning algorithms. However, when the domain of data is relatively new, such as data collected from patients in the CDT, it is difficult to know in advance which of these techniques will be most useful in uncovering trends or patterns. Since each individual tool requires a unique data input format, the conversion can be a tedious process, particularly when data mining is in the exploratory phase.

In order to solve this problem for CDT data analysis, I developed the Multitool Data Analysis (MDA) Package. This tool unifies different approaches to data mining on a single, easy-to-use platform. The MDA Package is designed to provide several options for analysis techniques, to provide file conversion for ease of use, and to facilitate easy launching of the programs from a single interface.

While many excellent tools are available in the machine learning community, each has drawbacks that make exclusive use undesirable. For example, Cluster 3.0, the clustering tool chosen for

4. THE MULTITOOLO DATA ANALYSIS PACKAGE

the MDA Package, allows for easy and direct modification of clustering parameters, but has no method for visualization of results. WEKA is probably the most common machine learning tool, as it provides many different functionalities, but it lacks the customization capability that more focused tools provide.

With the ability that the MDA Package provides to easily launch several tools, analysts will be able to more easily determine which techniques are most useful for their domain. One difficulty that arises when trying to use several analysis tools is that each requires its own idiosyncratic input file format, and some require additional template or header files. The MDA Package provides functionality that converts data in the well known .csv format into formats usable by each of the six selected tools. Rather than requiring the user to create input files for each program by hand, the package generates the necessary files from a .csv data file.

The MDA Analysis Package currently launches six machine learning tools:

- Weka: A multi-faceted tool providing clustering, dimensionality reduction, and supervised classification tools.
- Cluster 3.0: A tool that allows the user to directly edit parameters in the k-means clustering algorithm, the self-organizing map algorithm, and in principle components analysis.
- Databionics ESM: A tool that allows the user to see a 2D visualization of dimensionality reduction by SOMs.
- BSVM: A tool that trains a support vector machine with a focus on multi-class classification, allowing the user to edit parameters.
- PCA and EPCA, both for dimensionality reduction.

These six tools were chosen for the collective depth and breadth that they provide to an analyst, and to include tools for both supervised and unsupervised learning methods.

4.2 MDA Package

The MDA Analysis Tool can be run either as a command-line application or from a graphical user interface. The graphical interface is customized to work with CDT data, so analysis of other data must be done through the command-line option. The interface is simple to use, with an upper

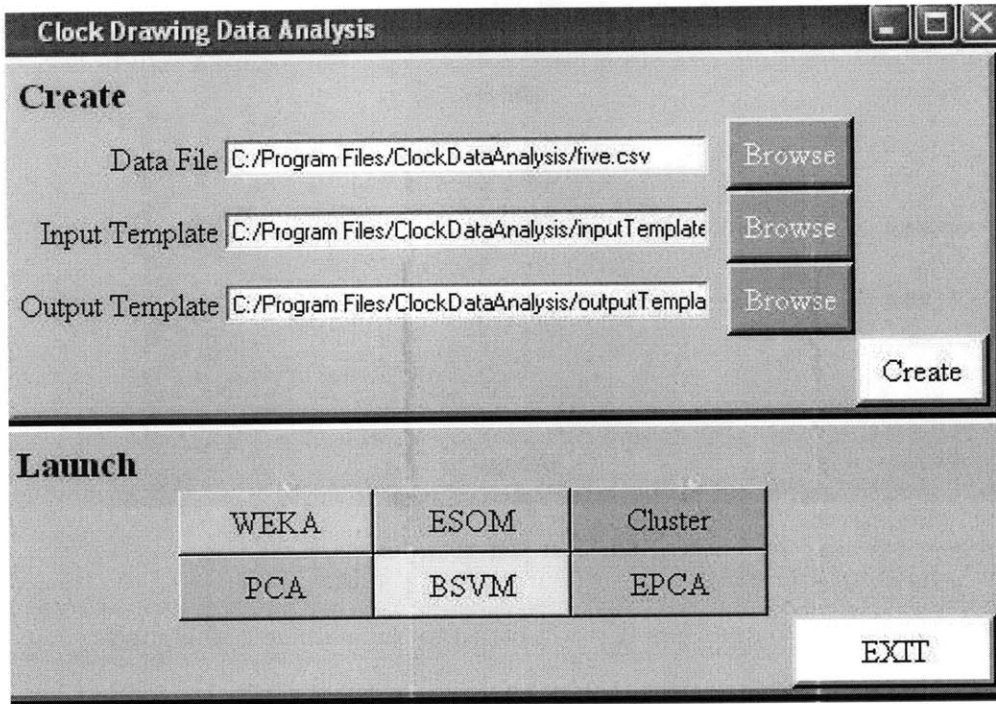


Figure 4.1: User Interface for CDT Data;

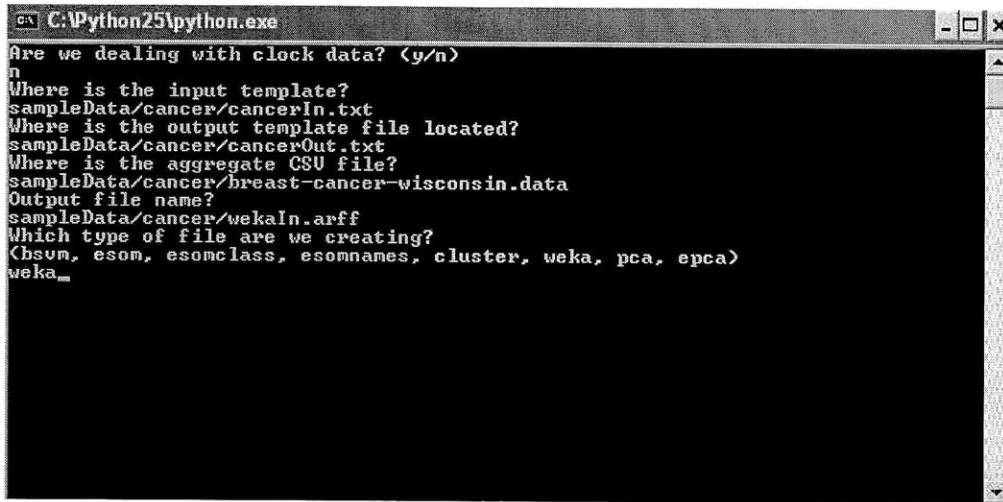
panel for file creation and a lower panel for launching the six available machine learning tools.

Data that is not derived from the Clock Drawing Test can be formatted for machine learning tools using the command line version of this package. The user will be prompted with a series of questions, asking for the location of all of the input files as well as the machine learning tool that will be used. Files can be created for each program individually, rather than in batch form as with the CDT data. We show a sample use of the command line interface in Figure 4.2.

4.3 File Creation

Three input files are used to guide the transformation of .csv files into a format usable by our selection of machine learning tools. One is the raw data file, and the other two are referred to as template files, which contain information on the type of input data and desired output data. The format for these template files was developed to be as simple as possible for the user, but still contain enough information about the input in order to correctly transform the input .csv file. I chose to use separate input and output template files so that different analyses can be performed on the same input data. By simply changing the output template file, the analyst can decide

4. THE MULTITool DATA ANALYSIS PACKAGE



```
C:\Python25\python.exe
Are we dealing with clock data? (y/n)
n
Where is the input template?
sampleData/cancer/cancerIn.txt
Where is the output template file located?
sampleData/cancer/cancerOut.txt
Where is the aggregate CSV file?
sampleData/cancer/breast-cancer-wisconsin.data
Output file name?
sampleData/cancer/wekaIn.arff
Which type of file are we creating?
(hsvm, esom, esomclass, esomnames, cluster, weka, pca, epca)
weka_
```

Figure 4.2: Text UI;

which features are included in the output files that will be fed to the machine learning tools.

ClockSketch Data Files

The Clocksketch software creates a .csv file in a common format, with the first line containing uniquely named features taken from the clock drawings, followed by two lines of data for each patient.

Here are the first few lines from a sample .csv file.

```
Type,PtNameF,PtNameL,MedicalRec,DocName,DwgTotStrokes,DwgTotTime
CS,"CIN1875169520","86","","Dana Penney",22,25.38900
CS,"CIN1875169520","86","","Dana Penney",25,25.58200
CS,"CIN1525846837","91","","Dana Penney",82,158.74800
CS,"CIN1525846837","91","","Dana Penney",30,63.19800
```

This is a standard .csv format, with values separated by commas and no extra spaces. Missing string data is indicated by "", and missing numerical data is indicated by two commas in sequence. The first line names seven features, and each name is unique. The next two lines have the same PtNameF and PtNameL features, indicating that the data is from the same patient.

Input Template File

The purpose of the input template file is to specify types for each of the features contained in the .csv file. This file will be used to ensure that the data we are analyzing follows an expected format- the analyst should be made aware of any unexpected values that appear in the data before they cause strange results in the analysis stage. The input template file will contain a list of all of

the features that are expected in the data file; our current feature vectors for the Clock Drawing Test are about 2000 in length. In addition to the name of each feature, the type of the feature will be included so that type checking can be performed before file creation. Specifications for the input template file can be found in Appendix A.

Here are a few lines from a sample input template file. To keep the example short, ellipses have been used to indicate text that is left out. Further explanation of the example can also be found in Appendix A.

```
Type NA
PtNameF NA
PtNameL NA
MedicalRec NA
DocName NA
FacilName NA
PtDiagnosis1 {ADD,ADHD,Alcohol Related,...}
...
DwgType NA
DwgTotStrokes int
DwgTotTime real
ClockFace1TotStrokes int
ClockFace1TotTime real
```

Mapping Input to Output

In order to facilitate flexible data analysis, MDA allows the analyst to define the list of output features in the output template file. Some of these output features will be functions of the input features, rather than a direct mapping of the input value to the output file. For example, the analyst might want to look at trends in the time it takes to draw the 1-3, and therefore want to add together digit and latency times.

To accomodate any derived features that the analyst wants to study, the file creation component of MDA contains a section in which “transform functions” can be defined. These functions are written in Python, and therefore some familiarity with Python is required of the analyst. The analyst needs to define any transform functions that are needed to compute new features from those in the .csv file. The transforms.py file holds these functions. Each function takes two lists as input: a list of input features and a list of additional parameters. Here is a sample function, Pure, as seen in the above example of an output template file.

4. THE MULTITOOL DATA ANALYSIS PACKAGE

```
def Pure(x, args):  
    return x[0]
```

See Appendix A.2 for transform function specifications and a more complex example of a transform function.

Output Template File

The output template file indicates the mapping between data in the .csv file and the desired output data we want to analyze. For example, we may want to look at only a few features of the input data. Or, the features that we want to analyze may be functions of several of the input features. See Appendix A.3 for specifications of the output template file.

Here are the first few lines from a sample output template file.

```
ClockFace1TotStrokesCOMMAND Pure {ClockFace1TotStrokesCOMMAND} {} int  
ClockFace1TotTimeCOMMAND Pure {ClockFace1TotTimeCOMMAND} {} real  
ClockFace1TotLenCOMMAND Pure {ClockFace1TotLenCOMMAND} {} real  
ClockFace1PnSpdQ1COMMAND Pure {ClockFace1PnSpdQ1COMMAND} {} real  
ClockFace1PnSpdQ2COMMAND Pure {ClockFace1PnSpdQ2COMMAND} {} real
```

Names in the first column (the output features) can be chosen by the analyst. The rows in this example each use the Pure function, which means that the value of the feature contained in column three will be passed on to the output. If there aren't any extra parameters for the function, as in this case, column four contains empty brackets. As in the input template file, the final column is used for type-checking on the values that will be printed to the output files. An error message will indicate if any output values are calculated that do not match the expected output value type. It is important for correct file creation that the output feature to be used as the class is the last row in the output file. See Appendix A.3 for another example of an output template file.

Output Specifications

The file creation functionality of MDA will provide a directory of files for use with the six provided machine learning tools. The directory will be named according to the time of its creation, and subdirectories will contain files for each of the tools. These files should be opened with the appropriate tool for further analysis. See Appendix B for an example of the complete process of file creation.

4.4 Machine Learning Tools

The MDA Tool can launch six machine learning tools; this section explores use of these tools using a sample cancer dataset. Four of these tools are stand-alone tools that could be used outside of MDA, while two were developed specifically for the MDA Package. More details on use of each of the tools can be found in Appendix C.

4.4.1 Weka

Weka is a tool that is well known in the machine learning community for the number of algorithm implementations it provides and its easy-to-use interface.

One valuable feature of Weka is its ability to easily preprocess data. Certain features can be selected/deselected, and then removed from the dataset. This functionality is useful if the analyst wants to observe the results of classification based on a subset of the features recorded. A side panel provides a bar graph of the distribution of the samples for each feature selected; for each possible value of the feature, the graph indicates how many of the samples have this feature value. This is useful because it allows the analyst to quickly and easily recognize the distribution across each feature.

Weka provides both supervised and unsupervised learning functionality. For example, k-means clustering is an unsupervised learning tool and the decision tree algorithm is a supervised classification method, both available in Weka. Once preprocessing is completed, these algorithms can be run and the output will be displayed on the right-hand panel of the interface. The interface is text based, so there is no visualization functionality for algorithms such as clustering or linear separators.

In addition, Weka provides association, attribute selection, and visualization capabilities. The Associate functionality will generate rules that partition the data into classes. The Select Attributes functionality includes algorithms such as Principles Components Analysis, which reduces the dimensionality of the data by choosing to keep only features that account for high variance in the data. Finally, the Visualization capability shows graphs of pairwise comparisons between features of the data. This is useful for determining if any features are correlated, or just visualizing the relationship between any two features.

Weka provides a wide array of tools for the user to begin analyzing a dataset. More detailed information about Weka, as well as sample usage, is available in Appendix C.1.

4. THE MULTITOOL DATA ANALYSIS PACKAGE

4.4.2 DataBionics ESOM

The ESOM (emergent self-organizing map) tool provided in the MDA Package has a more focused functionality than the WEKA tool. An ESOM is a projection of high-dimensional space onto a two-dimensional “map”, with distances in the higher-dimensional space indicated by differing colors on the map. The major advantage to using this tool for data analysis is that it provides a detailed map of the data and a visual representation of any clustering that occurs in the data. The program requires the user to load three files, all of which are provided by the file creation functionality of the MDA Tool. (Appendix C.2)

4.4.3 Cluster 3.0

The clustering tool provided in the MDA Package was originally developed for clustering of DNA microarray data, and therefore has the capabilities required to handle large datasets. Although Cluster 3.0 contains PCA and SOM functionality, better implementations of these algorithms can be found in the PCA and ESOM tools. However, the clustering functionality of Cluster 3.0 allows for tight control over parameters by the analyst. Data can be centered and normalized before clustering is performed, and eight different distance metrics can be chosen for the k-means clustering algorithm. The number of clusters and iterations of the algorithm must also be chosen before run time. Clustering output is a text file which contains a list of each of the samples in the data and the cluster to which they have been assigned. (Appendix C.3)

4.4.4 BSVM

The BSVM tool provided is a tool for training SVMs on the data, allowing users to select desired kernel type, loss function, tolerance, and many other parameters. BSVM allows the user to choose from four different algorithms for multi class classification: bound-constrained multi-class support vector classification (SVC), multi-class SVC from solving a bound-constrained problem, multi-class SVC from Crammer and Singer, and bound-constrained support vector regression. Users can edit type of SVM, kernel type, and values for 12 other parameters. Once a model has been trained, it can be used to classify new unlabeled data. (Appendix C.4)

4.4.5 PCA

The PCA tool provided performs principle components analysis on the data and returns a new dataset in which the number of features has been reduced. Researchers perform PCA on data in order to get rid of features that do not account for significant variance between samples of different classes, and bring the data down to a more manageable size. The new features that are created

will be combinations of original features, mathematically creating the best matrix decomposition of the original data, but often having little physical interpretation.

This Java implementation of PCA was specifically developed for the MDA Package, to give users more control over the parameters of the algorithm. When PCA is performed, the number of features in the data will be reduced. It can be reduced to a set number of features, or reduced until the set of features accounts for a certain percentage of the variance in a dataset. Most PCA implementations only allow for the latter approach, but sometimes it will be necessary for analysts to choose a specific number of features to keep. Therefore, I have created a Java implementation of PCA that lets the user decide whether to reduce data to a certain size or to account for a specified variance. (Appendix C.5)

4.4.6 EPCA

The EPCA (Exponential PCA) tool provides almost the same functionality as the PCA tool, but has a different underlying algorithm. The goal of EPCA is dimensionality reduction, but this algorithm takes into account the differing feature types in the data. Therefore, the new features are chosen by a more complex process than simply choosing those that account for the highest variance across samples. The EPCA algorithm requires the number of output features desired as a parameter, and therefore the user must make this decision before running the program.

This python EPCA implementation was developed specifically for the MDA Package. It is based on a MATLAB implementation by Peter Gehler of the EPCA algorithm in the Collins et al. paper. The MATLAB implementation accounts for data that is completely binary or integer value. My implementation allows for different types within a dataset, and uses different loss functions for the different features. See Appendix C.6 for more information.

4. THE MULTITool DATA ANALYSIS PACKAGE

5

Sample Analysis

This chapter will walk through analysis of a sample CDT dataset, highlighting the results from each of the 6 machine learning tools that seem most promising. The intention of this chapter is to introduce methods rather than show results, since our sample dataset remains far too small for results to be significant.

5.1 Input Overview

The .csv file output from the CDT contains approximately 2000 features and 44 patients. However, not all of these features are useful for classification and some were not included in the output template file. Among the unused features are Type, PtNameF, PtNameL, MedicalRec, DocName, and FacilName; these features have NA as their data type in the input template file and are not present in the output template file. You can choose which features to exclude, and we have selected the few that shouldn't have an effect on classification. For example, PtNameF is the first name of the patient, and we do not expect there to be a pattern linking first name with diagnosis.

There are also features that we wanted to combine to create new output features. The order-ToAbsoluteDigit function described in Appendix A.2 was used to take all 60 possible digit entries and save only one for each digit. This function reorders the data so that we are always comparing like digits rather than digits drawn in the same order sequentially. In this example, rearranging how the digits are stored was the only function performed on input features. PtDiagnosis1 is the last feature listed on the output template file, since this is the feature chosen to indicate the class of a sample.

5. SAMPLE ANALYSIS

5.2 Weka- The Decision Stump

The Decision Stump algorithm is a good first step in creating a decision tree for the data. Decision trees are easy to understand, because they involve asking a series of questions about features that lead to classification. The Decision Stump chooses the first feature that should be examined, the one that can correctly classify the most samples with just one threshold value.

For this dataset, PtAgeAtTestCOMMAND (which would be the same value as PtAgeAtTestCOPY, since it is the same patient) was chosen as the most important feature. If the patient is over 69.337 years of age, the patient is classified as having DementiaSDAT, while anyone under this age is classified as a Healthy Control. The choice of this feature makes sense, because the Healthy Controls that we had are much younger in age than the other patients. In addition, DementiaSDAT and Healthy Control have the largest number of samples in our data, so choosing these two classifications naturally leads to the highest rate of correct classification. This phenomenon will be fixed when the dataset becomes much larger and all diagnoses are represented. The output from the Decision Stump algorithm is:

```
=== Run information ===

Scheme:      weka.classifiers.trees.DecisionStump
Relation:    clocks
Instances:   89
Attributes:  867
              [list of attributes omitted]
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

Decision Stump

Classifications

PtAgeAtTestCOMMAND <= 69.33717999999999 : HealthyControl
PtAgeAtTestCOMMAND > 69.33717999999999 : DementiaSDAT
PtAgeAtTestCOMMAND is missing : HealthyControl

Class distributions
...
```

Time taken to build model: 0.09 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	31	34.8315 %
Incorrectly Classified Instances	58	65.1685 %
Kappa statistic	0.1435	
Mean absolute error	0.0227	
Root mean squared error	0.1091	
Relative absolute error	88.9191 %	
Root relative squared error	98.1911 %	
Total Number of Instances	89	

This decision only correctly classified 34.8% of the samples, but this makes sense since we are only using one feature and two possible classes. The other classification methods, in particular those in the Tree category, perform much better by relying on more of the features. However, this means that they are more complicated and require more time and space resources.

5.3 ESOM

The ESOM tool provides a good visualization of our data. First, the three files created by the MDA tool were loaded, and then a self-organizing map was trained using the default program parameters. Figure 5.1 shows the resulting map.

Although the dataset is small, it is possible to see a bit of clustering for some of the classes. In Figure 5.1, the Dementia-SDAT samples are navy blue squares that are in bold. Since green represents low distance, the three samples near the top are close together, as are the two slightly lower. This is by no means amazing performance, but the small size of the dataset is a limiting factor. Once we have a large number of samples with each classification, we can hope to see more definitive clustering.

5.4 BSVM

The BSVM tool is a useful supervised classification method for finding separators in the data. For example, if the linear kernel option is chosen, BSVM will find linear separators that most accurately classify each sample. It is very easy to create a model using BSVM, and then test the

5. SAMPLE ANALYSIS

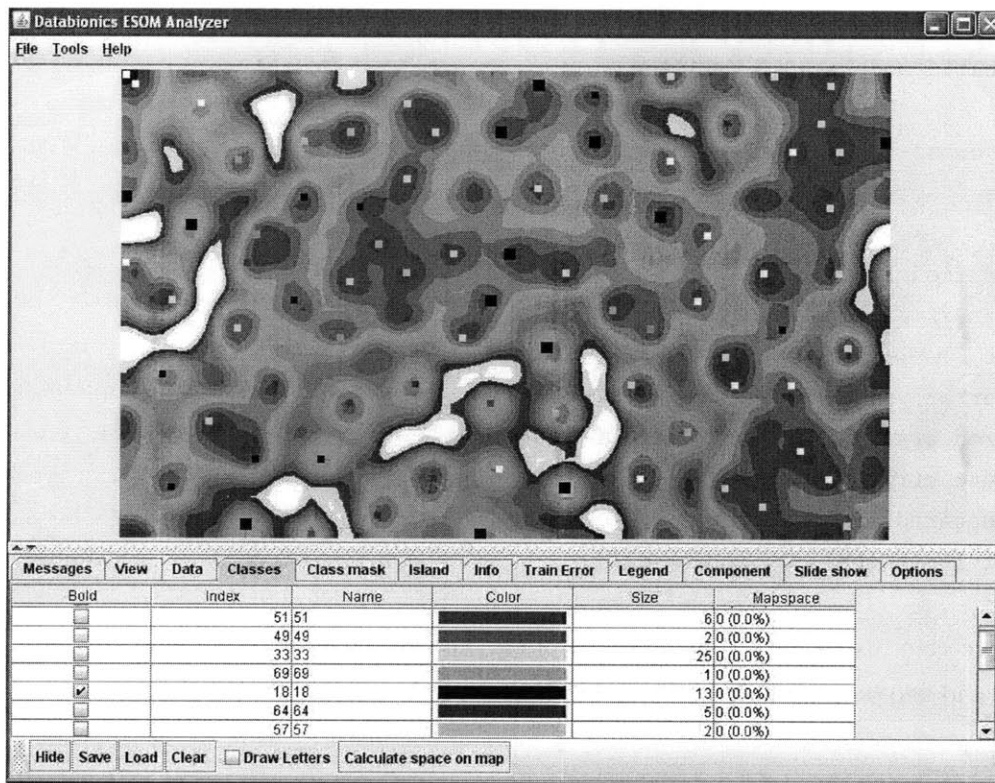


Figure 5.1: ESOM- CDT Sample;

model on a reserved set of test samples.

The file that MDA provides contains all of the samples from the original .csv file with the classification as the first feature in each row. An easy option to get started is to train an SVM on the entire dataset. However, in order to provide the model with new data for testing, we first set aside some of the samples and placed them in a new file called bsvmTest.txt. To do this, seven of the rows were deleted from the file created by MDA and copied into bsvmTest.txt. Once the dataset is large enough, these test samples will be chosen at random out of the dataset, or cross-validation will be used. However, since our dataset is so small, we chose samples for testing that come from classes that are highly represented in the data. Class 0 is unknown diagnosis and class 33 is Healthy Control. The results of classifying seven samples by a model built on the remaining 81 samples are:

Actual Class	Classification
0	33
33	33
33	33

33	33
33	33
0	0
0	0

Of the seven samples, six are classified correctly. This is a good start, and the simplicity of the BSVM tool makes it a promising option for when more data is available.

5.5 Clustering

In this section, we will examine the results of clustering on the sample dataset. We will first cluster the original data, and then perform clustering on dimensionality-reduced data output from the PCA and EPCA tools.

Original Data

To cluster the data, it was loaded into Cluster 3.0 and the k-means tab was selected. Under the “Genes” panel, “Organize Genes” was selected and we chose to use 20 clusters. Upon clicking “Execute”, the k-means algorithm produced an output file and a new data file in the same directory as the input file. The output file, with the .kcg extension, lists the samples that were clustered together in cluster order. To see how well the clustering performed, we look at one of the clusters and the actual classifications of the samples that were classed together. The first column below shows the sample number, the second column the cluster number, and the actual classification has been added as a third column.

```

11 0 PD-DBS Pre-Surgical
13 0 Healthy Control
29 0 0
34 0 0
36 0 Epilepsy - Right hemisphere foci
42 0 Healthy Control
46 0 0
57 0 Dementia - SDAT
61 0 0
80 0 0
84 0 Stroke - right hemisphere: PCA

```

In this cluster of samples, there are a wide range of classifications. The most common class is 0, which seems to indicate that the samples of unknown diagnosis group together. Overall, our data size is too small to get accurate clustering. However, we can compare these results with the results of performing dimensionality reduction first and then clustering.

5. SAMPLE ANALYSIS

PCA Reduced

When PCA is performed on the data with 95% variance accounted for, then new feature vectors of length 35 are assigned to each sample. One of the groups found by clustering this new data is:

```
13 11 Healthy Control
27 11 0
29 11 0
34 11 0
36 11 Epilepsy - Right hemisphere foci
42 11 Healthy Control
43 11 Healthy Control
59 11 Stroke - left hemisphere: MCA
61 11 0
80 11 0
82 11 Healthy Control
```

The first observation to be made is that performing PCA first allows clustering to happen much quicker. Clustering went from taking about 30 seconds on the original data to only 2 seconds on the dimensionality-reduced data. Secondly, we can see that most the samples remain in the same group. Samples 13, 29, 34, 36, 42, 61, and 80 cluster together in both the original data and the new data, while only four of the samples in the original cluster are no longer present in the new cluster. Both clusters are dominated by the 0 class, and the data that went through PCA first also has 4 Healthy Control samples in this cluster. With this small sample size, the most notable trend is that reducing the dimensionality of the data before clustering gives different but similar clustering results.

EPCA Reduced

Finally, we perform the same steps using EPCA. We reduce the data down to 35 dimensions, and the following are two clusters found in the new data:

```
12 7 PD-DBS OFF Post-Surgical
13 7 Healthy Control
34 7 0
36 7 Epilepsy - Right hemisphere foci
61 7 0
73 7 Epilepsy - Left hemisphere foci
74 7 Healthy Control
87 7 Stroke - right hemisphere: MCA
```

```
3 17 PD-DBS ON Post-Surgical
20 17 0
27 17 0
29 17 0
39 17 Normal Aging
57 17 Dementia - SDAT
63 17 Stroke - left hemisphere: MCA
77 17 Epilepsy - Indeterminate foci
80 17 0
```

Using the data reduced by EPCA, we get a much different clustering structure than with PCA. The samples from the one cluster that we followed from the original data and the PCA-reduced data are split up between two clusters after reduction by the EPCA algorithm. We can see that samples 13, 34, 36, and 61 are clustered together in each of the three methods, as are samples 29 and 80.

In conclusion, PCA and EPCA dimensionality-reductions create a dataset that is of a much more manageable size, but they change the clustering output by leaving out some features. This process of comparing the results of the three clustering methods should be repeated on a larger dataset, to determine if dimensionality reduction improves clustering.

5. SAMPLE ANALYSIS

6

Contributions

I made several contributions to the Clock Drawing Test. To improve the stroke classification stage of data analysis, I collected samples to train a model for nearest neighbor classification of digits. I used stroke thickening and rotation invariance to improve cross-validation results. I then developed an algorithm that uses temporal and spatial features of strokes, as well as digit recognition, to classify the digits on the face of a clock.

I developed the Multitool Data Analysis Package to ease exploratory data analysis by providing file conversion and tool launching functionality. MDA includes new implementations of Principles Components Analysis(PCA) and Exponential PCA, an extension of the PCA algorithm to handle binary and integer valued data through use of Bernoulli, Poisson, and Minimum Mean Squared Error Loss functions. Each of these contributions helps maintain the ease of administration of the CDT and subsequent data analysis.

6. CONTRIBUTIONS

7

Bibliography

Anoto(A), "Development Guide for Services Enabled by Anoto Functionality." Anoto AB: 2006.

Anoto(B), "Paper Interface Guidelines." Anoto AB: 2006.

Collins, Michael et al. "A Generalization of Principle Component Analysis to the Exponential Family." AT&T Labs.

Penney, Dana. "A Clock Drawing Test scoring application for dementia, vascular disease and Parkinson's Disease: A Pilot Study."

Richardson, Hayley et al. "A Comparison of Scoring Protocols on the Clock Drawing Test in Relation to Ease of Use, Diagnostic Group, and Correlations with Mini-Mental State Examination." *Journal of the American Geriatrics Society*. Jan 2002 50:169-173.

Royall, Donald et al. "CLOX: An Executive Clock Drawing Test." *Journal of Neurology, Neurosurgery, and Psychiatry*. 1998 64:588-594.

Shulman, Kenneth et al. "The Challenge of Time:Clock-Drawing and Cognitive Function in the Elderly." *International Journal of Geriatric Psychiatry*. 1986 1:135-140.

7. BIBLIOGRAPHY

8

Appendix

A. File Formats

A.1 Input Template File

- This is a tab-separated file, with two columns and a number of rows equal to the number of features in the .csv file.
- The first column of the input template file contains the feature names and hence should exactly match the first row of the .csv file.
- The second column lists the data type of values for each feature. Possibilities include: 1cm
 - bin : 0 or 1.
 - real : real number
 - int : integer values
 - {x,y,z} : This notation provides an enumeration of specific values possible for this column. The list can contain strings and/or numbers.
 - NA : The data will not be used, and hence there is no need to specify a type.

```
Type NA
PtNameF NA
PtNameL NA
MedicalRec NA
DocName NA
FacilName NA
PtDiagnosis1 {ADD,ADHD,Alcohol Related,...}
```

8. APPENDIX

```
...
DwgType NA
DwgTotStrokes int
DwgTotTime real
ClockFace1TotStrokes int
ClockFace1TotTime real
```

Notice in the sample input template file that there are two tab-separated columns for each row. There are no tabs or spaces following the second column. The first column matches the first row of the sample .csv file (Type,PtNameF). If there are discrepancies between the header line of the .csv file and the first column of the input template file, an error message will appear indicating the specific feature that needs to be fixed. The second column of the input template file will be used for type-checking on the input data. For example, the feature Type is not restricted in value but DwgTotStrokes must take on an integer value. PtDiagnosis1 is an example of an enumeration; this list must contain every possible value for the data type it describes. If a value is found that does not match the input template specifications, the user will be notified via an error message indicating which feature is incorrect.

A.2 Transform Functions

Transform functions are written to convert input features to output features. Each transform function must take two arguments, the first is a list of input features and the second is a list of extra parameters. The Pure function is the simplest transform function, simply passing the input value through to the output. The argument x is a list of input features and args is the list of extra parameters. For this function, x will be of length one, and the first term of x will be returned. Since no extra parameters are needed, args will be empty.

```
def Pure(x, args):
    return x[0]
```

orderToAbsoluteDigit is a slightly more complicated example.

```
def orderToAbsoluteDigit(l, num):
    num = num[0]
    index0f = None
    for i in range(60):
        if l[i] == num:
```

```

        index0f = i
    else:
        #since the digit 6 can be stored as 6 or "6", we check here to make
        #sure we recognize the string form of a digit.
        try:
            if int(l[i]) == num:
                index0f1 = i
        except ValueError:
            pass

    if index0f == None:
        return 0
    else:
        if l[index0f+60] == " ":
            print 'error'
        return l[index0f+60]

```

The goal of this function is to take in as `l` the list of digits in the order they were drawn, appended to a list of corresponding feature values for these digits (ie: length), and return a feature value for a particular digit. This is necessary because the CDT records digits in the order in which they are drawn, not numerical order. So if we want to compare the length of the 3 from two different clocks, in each case we have to look up when the 3 was drawn. For example, `orderToAbsoluteDigit(12,6,3,9,...,1.2,1.6,1.8,1.9,..., 6)` would find when the 6 was drawn by its position in the first half of the list. The first half of `l` (the first four numbers shown plus missing entries indicated by ellipses) contains the digits in the order they were drawn (12,6,3,9); the second half contains their values. For example, the 12 was of length 1.2 and the 6 was of length 1.6. Once we've found when the 6 was drawn, we can select the correct length for the 6. These lists will normally be of length 120, because there are 60 possible digits and 60 corresponding values. (There are 60 possible digits because some cognitive deficits cause patients to repeat digits, and the program has room for up to 5 such repetitions of each digit.) The second argument to the `orderToAbsoluteDigit` function is a list containing one number- the number that we want to find.

A.3 Output Template File

The output file contains 5 tab-separated columns:

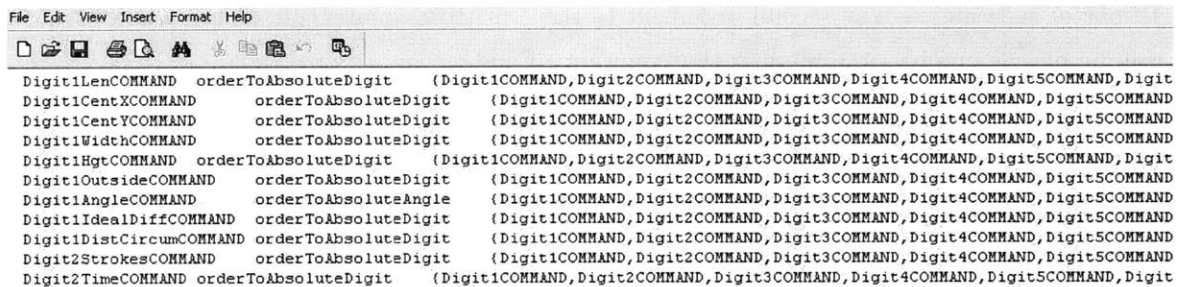
- The first column contains names for the output features, which must be unique.

8. APPENDIX

- The second column contains the name of the function that will be invoked on the input data to produce the output data. Possible functions will be defined in the `transforms.py` file
- The third column contains the names of features input to the function in column two, contained between brackets, with no spaces between features. Each feature must exactly match one of the feature names in the input template file with either “COMMAND” or “COPY” appended to the end.
- The fourth column contains any additional parameters that the function in column two requires, contained within brackets without any spaces.
- The fifth column contains the type of the data that will be output from the function and stored in the output file. Possible types are the same as those for the input template file.

Here is an example of several lines of an output template file. Ellipses indicate missing information, to keep the example short.

```
Digit1StrokesCOMMAND orderToAbsoluteDigit {Digit1COMMAND,...,Digit60COMMAND,
Digit1StrokesCOMMAND,...,Digit60StrokesCOMMAND} {1} int
Digit1TimeCOMMAND orderToAbsoluteDigit {Digit1COMMAND,...,Digit60COMMAND,
Digit1TimeCOMMAND,...,Digit60TimeCOMMAND} {1} real
Digit1LenCOMMAND orderToAbsoluteDigit {Digit1COMMAND,...,Digit60COMMAND,
Digit1LenCOMMAND,...,Digit60LenCOMMAND} {1} real
```



```
File Edit View Insert Format Help
Digit1LenCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND, Digit
Digit1CentXCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit1CentYCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit1WidthCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit1HgtCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND, Digit
Digit1OutsideCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit1AngleCOMMAND orderToAbsoluteAngle {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit1IdealDiffCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit1DistCircumCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit2StrokesCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND
Digit2TimeCOMMAND orderToAbsoluteDigit {Digit1COMMAND, Digit2COMMAND, Digit3COMMAND, Digit4COMMAND, Digit5COMMAND, Digit
```

Figure 8.1: Snapshot of the Output Template File

The sample output template file shows a function called `orderToAbsoluteDigit`, where inputs are not passed directly to the output template file. As described in Appendix A.2, this lookup function takes the list of digits in the order that they were drawn and the value of a particular feature of that digit (length, time, etc), and returns the value corresponding to a particular digit.

B. Sample File Creation

This chapter walks through the process of creating files for a sample dataset. Since this is not a CDT dataset, the command line option for running the program will be used. However, all of the input files used follow the same rules.

Cancer Dataset

The cancer dataset contains information from 699 tumor samples. There are ten attributes in addition to the class attribute which labels the tumor as benign or malignant. Machine learning algorithms are used on this dataset to discover which combinations of attributes indicate that a tumor is malignant. Below are excerpts from the data file, input template file and output template file for the cancer dataset.

Data file

```
id,clump thickness,size uniformity,shape uniformity,...
1000025,5,1,1,1,2,1,3,1,1,2
1002945,5,4,4,5,7,10,3,2,1,2
1015425,3,1,1,1,2,2,3,1,1,2
1016277,6,8,8,1,3,4,3,7,1,2
1017023,4,1,1,3,2,1,3,1,1,2
1017122,8,10,10,8,7,10,9,7,1,4
1018099,1,1,1,1,2,10,3,1,1,2
```

Just as with output from the CDT, acceptable data files are `.csv` files, with the first line of the file containing names of each field, separated by commas. The remaining lines each represent one subject/patient/sample, and entries on each line are separated by commas. There should be no spaces in the text lines or at the end of the lines.

8. APPENDIX

Input and Output Template File

Here is the input template file for the cancer data set in its entirety.

```
id int
clump thickness {1,2,3,4,5,6,7,8,9,10,?}
size uniformity {1,2,3,4,5,6,7,8,9,10,?}
shape uniformity {1,2,3,4,5,6,7,8,9,10,?}
marginal adhesion {1,2,3,4,5,6,7,8,9,10,?}
single cell size {1,2,3,4,5,6,7,8,9,10,?}
bare nuclei {1,2,3,4,5,6,7,8,9,10,?}
bland chromatin {1,2,3,4,5,6,7,8,9,10,?}
normal nucleoli {1,2,3,4,5,6,7,8,9,10,?}
mitoses {1,2,3,4,5,6,7,8,9,10,?}
class {2,4,?}class {2,4,?}
```

The input template file is tab-separated, with the first column containing the name of each attribute in the input file, which must match the corresponding entry in the first row of the data file. The second column records the type of value that each attribute can take. For example the first row, the id attribute, is an integer. In fact, in this dataset all of the attributes are integers. However, if values are limited, we can be more specific about their possible values by creating an enumeration. For example, clump thickness can only take on values 1-10 or ?.

Here is an output template file for the cancer dataset.

```
clump thickness Pure {clump thickness} {} {1,2,3,4,5,6,7,8,9,10,?}
size uniformity Pure {size uniformity} {} {1,2,3,4,5,6,7,8,9,10,?}
shape uniformity Pure {shape uniformity} {} {1,2,3,4,5,6,7,8,9,10,?}
marginal adhesion Pure {marginal adhesion} {} {1,2,3,4,5,6,7,8,9,10,?}
single cell size Pure {single cell size} {} {1,2,3,4,5,6,7,8,9,10,?}
bare nuclei Pure {bare nuclei} {} {1,2,3,4,5,6,7,8,9,10,?}
bland chromatin Pure {bland chromatin} {} {1,2,3,4,5,6,7,8,9,10,?}
normal nucleoli Pure {normal nucleoli} {} {1,2,3,4,5,6,7,8,9,10,?}
mitoses Pure {mitoses} {} {1,2,3,4,5,6,7,8,9,10,?}
```

The output template file is very similar to the input template file. The first column contains the names of desired output attributes. The user has the freedom to chose names for these attributes, as long as there are no duplicates. The second column contains the name of the function that will be performed in order to create this attribute. In this case, all of

the functions are “Pure” because we are simply passing input values to output. The third column contains a list of the input attributes that will be passed to the function, and the fourth column contains a list of constants that will be passed to the function. Finally, the last column contains the type of values that the new output attribute can take on.

Appendix C explores the different machine learning tools provided in the MDA Package using the cancer dataset.

C. Machine Learning Tools

C.1 Weka

Preprocess

Once the explorer is open, click on “Open File” to load your data file. You will then need to locate the file that was created in the previous stage. It will be contained in folder similar to `ClockData_234545612.23/weka`, located in the same directory as the `gui.py` file. The file that you load should have an `.arff` extension. Once data is loaded, the screen will appear as below. (Fig 8.2)

When the data is loaded, the “Preprocess” tab will be visible. Working in this tab, the user is able to filter the data by any of the features. Certain features can be selected/deselected, and then removed from the dataset. This functionality is useful if you want to observe the results of classification based on a subset of the features recorded. The right hand panel provides a bar graph of the distribution of the samples for each feature selected; for each possible value of the feature, the graph indicates how many of the samples have this feature value.

Each remaining tab at the top of the screen is a possible analyses technique that can be used on the data. However, some of the tools contained in Weka are supervised learning methods and some are unsupervised methods. For example, k-means clustering is an unsupervised learning tool available through weka and the decision tree algorithm is a supervised classification method. For supervised methods, the user needs to ensure that the classification attribute is present in the Attributes panel of the Preprocess tab. For unsupervised methods, the user needs to ensure that classification is not used as one of the features, by removing or deselecting.

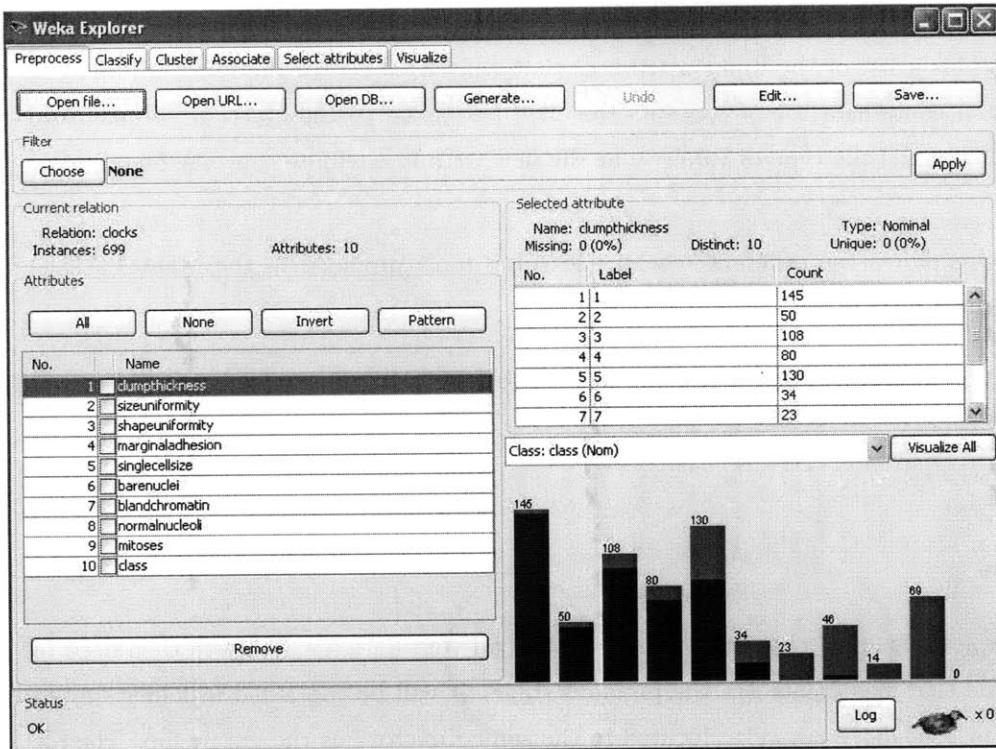


Figure 8.2: Weka with Data Loaded;

Classify

Supervised classification algorithms are found under the “Classify” tab. A “Choose” button will allow the user to select which classification method to use. (Figure 8.3)

In this case, choosing “Decision Stump” under the Trees category is one of the simpler classifiers to examine. This algorithm will determine which single attribute is the best at classifying the data. In this case, sizeUniformity of the cell is chosen as the attribute that is the highest indicator of class. Below is the output that will appear in the right-hand panel of the Weka interface when the decision stump algorithm is run, with comments added after hash marks.

```
=== Classifier model (full training set) ===
```

```
Decision Stump
```

```
Classifications
```

```
#If sizeuniformity is 1 or missing, then classify the sample as class 2.
```

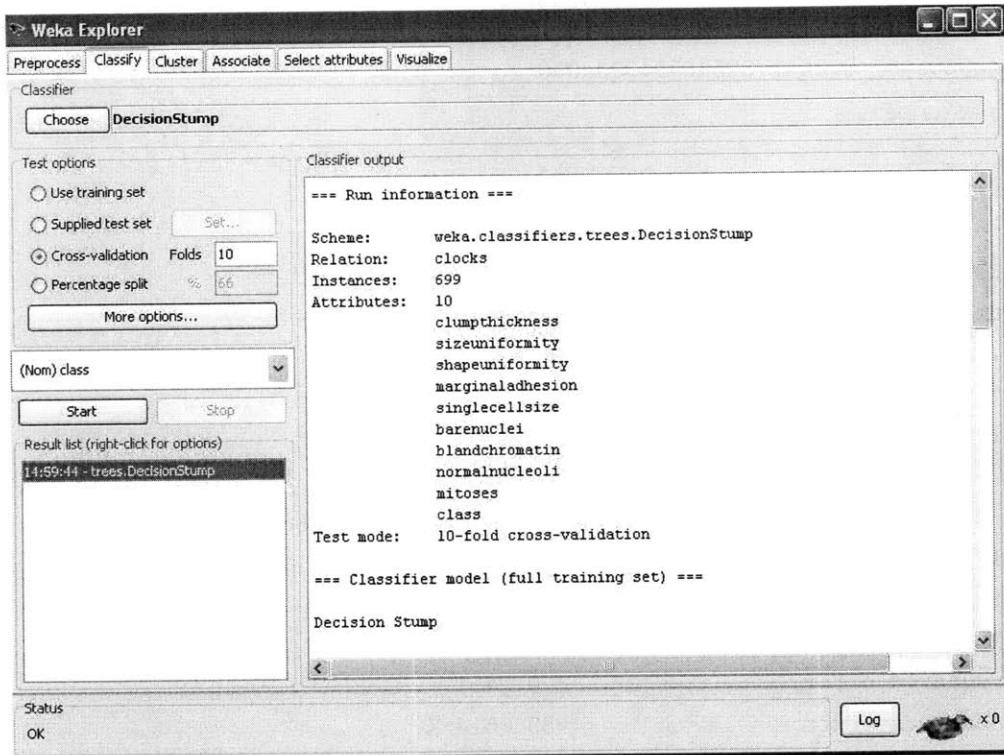



Figure 8.3: Weka-Classify;

```
#Otherwise classify the sample as class 4.
```

```
sizeuniformity = 1 : 2
sizeuniformity != 1 : 4
sizeuniformity is missing : 2
```

```
#Here is the breakdown of results of using this attribute.
#For example, .98 of the samples with sizeUniformity of 1 are
#correctly placed in class 2.
```

```
Class distributions
```

```
sizeuniformity = 1
2 4 ?
0.9895833333333334 0.010416666666666666 0.0
sizeuniformity != 1
2 4 ?
0.24761904761904763 0.7523809523809524 0.0
sizeuniformity is missing
```

8. APPENDIX

```
2 4 ?
0.6552217453505007 0.3447782546494993 0.0
```

Time taken to build model: 0.03 seconds

#Statistics, indicating how well the classifier performed.
#Root mean squared error is a common measure of accuracy.

```
=== Stratified cross-validation ===
=== Summary ===
```

Correctly Classified Instances	617	88.269 %
Incorrectly Classified Instances	82	11.731 %
Kappa statistic	0.758	
Mean absolute error	0.1197	
Root mean squared error	0.2451	
Relative absolute error	39.6614 %	
Root relative squared error	63.1639 %	
Total Number of Instances	699	

#This is a breakdown of how well the classifier did by class.
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.83	0.017	0.99	0.83	0.903	0.884	2
	0.983	0.17	0.752	0.983	0.853	0.884	4
	0	0	0	0	0	?	?
Weighted Avg.	0.883	0.07	0.908	0.883	0.885	0.884	

#The confusion matrix is a common output of classification algorithms, showing
#how many samples in each class were given each possible classification. The
#numbers appearing on the diagonals are correct classifications.

```
=== Confusion Matrix ===
```

```
  a   b   c  <-- classified as
380  78   0 |  a = 2
  4 237   0 |  b = 4
  0   0   0 |  c = ?
```

The confusion matrix at the bottom indicates the number of cells that were classified correctly using only this feature.

Cluster

When the Cluster tab is selected, a “Choose” button near the top of the screen allows for selection of a particular clustering algorithm to be used. (Fig 8.4)

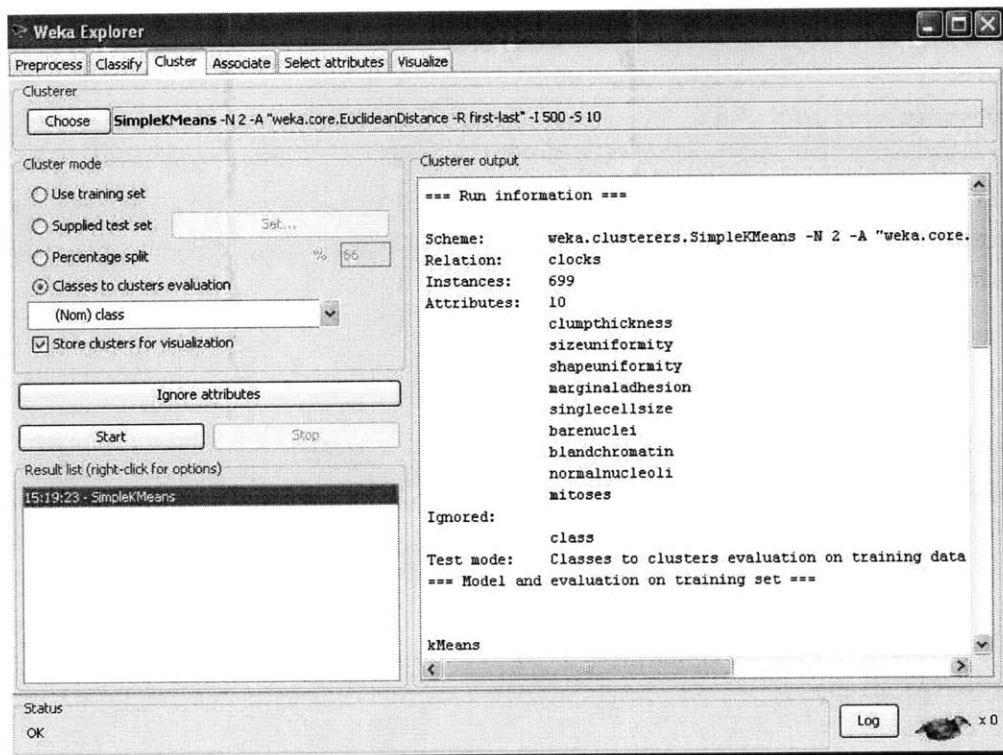


Figure 8.4: Weka-Cluster;

K-means is one of the simplest, and finds cluster centers that minimize distance from each data point to the nearest cluster center. Notice that “classes to cluster evaluation” is selected, meaning that the “class” attribute will be ignored as a feature and instead used to determine the number of cluster centers to look for. The output of clustering is displayed in the panel on the right.

kMeans

=====

8. APPENDIX

Number of iterations: 3

Within cluster sum of squared errors: 2544.0

Missing values globally replaced with mean/mode

Cluster centroids:

Attribute	Full Data (699)	Cluster#	
		0 (228)	1 (471)
=====			
clumpthickness	1	10	1
sizeuniformity	1	10	1
shapeuniformity	1	10	1
marginaladhesion	1	10	1
singlecellsize	2	3	2
barenuclei	1	10	1
blandchromatin	2	7	2
normalnucleoli	1	10	1
mitoses	1	1	1

Clustered Instances

0 228 (33%)

1 471 (67%)

Class attribute: class

Classes to Clusters:

```
0 1 <-- assigned to cluster
14 444 | 2
214 27 | 4
0 0 | ?
```

Cluster 0 <-- 4

Cluster 1 <-- 2

Incorrectly clustered instances : 41.0 5.8655 %

First, we can see a chart that indicates the coordinates of each cluster center. In this case, the cluster centers have 9 dimensions (rows). Then, smaller charts indicate the percentage of the data points that fell in each cluster, as well as the number of data points from each actual classification that were placed in each cluster. From these numbers, a percentage error is calculated.

Associate

When the “Associate” tab is selected, several algorithms for creating rules based on the data are available. (Fig 8.5)

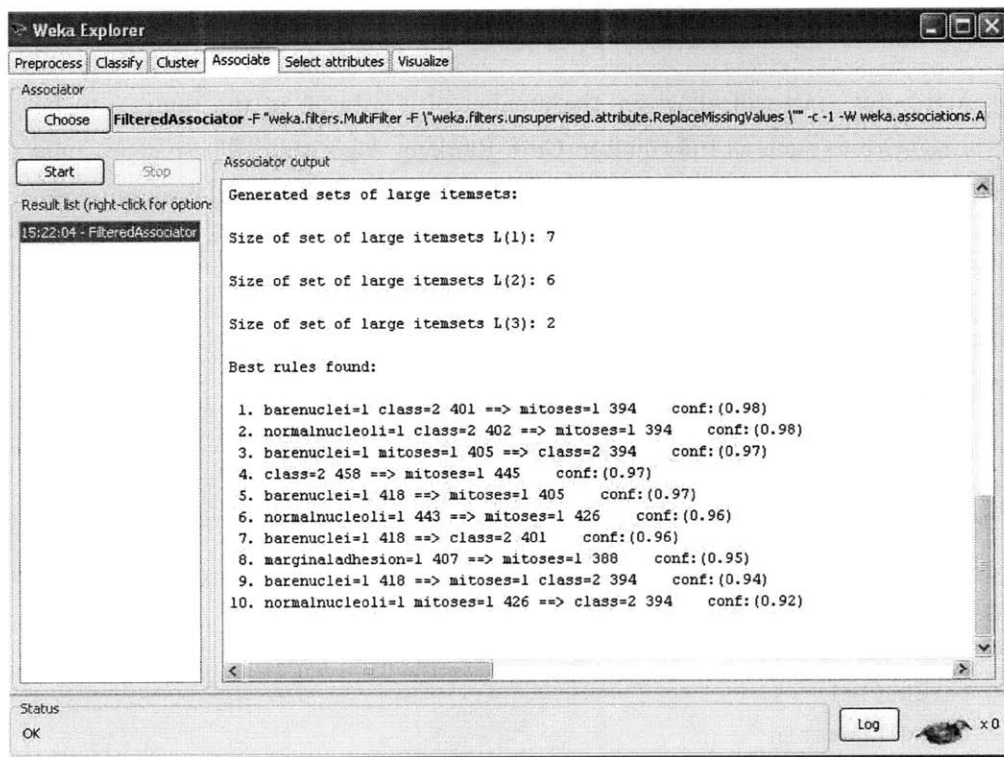


Figure 8.5: Weka-Associate;

Using the FilteredAssociator, the following rules were found for the cancer dataset.

Best rules found:

1. barenucelei=1 class=2 401 ==> mitoses=1 394 conf:(0.98)
2. normalnucleoli=1 class=2 402 ==> mitoses=1 394 conf:(0.98)
3. barenucelei=1 mitoses=1 405 ==> class=2 394 conf:(0.97)

8. APPENDIX

4. class=2 458 ==> mitoses=1 445 conf:(0.97)
5. barenucllei=1 418 ==> mitoses=1 405 conf:(0.97)
6. normalnucleoli=1 443 ==> mitoses=1 426 conf:(0.96)
7. barenucllei=1 418 ==> class=2 401 conf:(0.96)
8. marginaladhesion=1 407 ==> mitoses=1 388 conf:(0.95)
9. barenucllei=1 418 ==> mitoses=1 class=2 394 conf:(0.94)
10. normalnucleoli=1 mitoses=1 426 ==> class=2 394 conf:(0.92)

The first rule says that of the 401 samples that have the barenucllei feature value 1 and are in class 2, 394 have the feature value 1 for mitoses as well. This is a .98 ratio, and rules are listed from highest to lowest certainty.

Select Attributes

The “Select Attributes” tab will allow for the number of attributes in the samples to be condensed via algorithms such as Information Gain Ranking Algorithm. These algorithms are based on different equations for measuring the utility of each feature in determining the class of a sample. (Fig 8.6)

When the Information Gain Ranking Algorithm is selected using the cancer dataset, the following ranking of features is calculated.

Attribute Evaluator (supervised, Class (nominal): 10 class):
Information Gain Ranking Filter

Ranked attributes:

0.684	2	sizeuniformity
0.661	3	shapeuniformity
0.569	6	barenucllei
0.548	7	blandchromatin
0.514	5	singlecellsize
0.475	8	normalnucleoli
0.465	1	clumpthickness
0.449	4	marginaladhesion
0.21	9	mitoses

Selected attributes: 2,3,6,7,5,8,1,4,9 : 9

This information could be considered by the user when deciding which features to measure or collect data on and which to leave out.

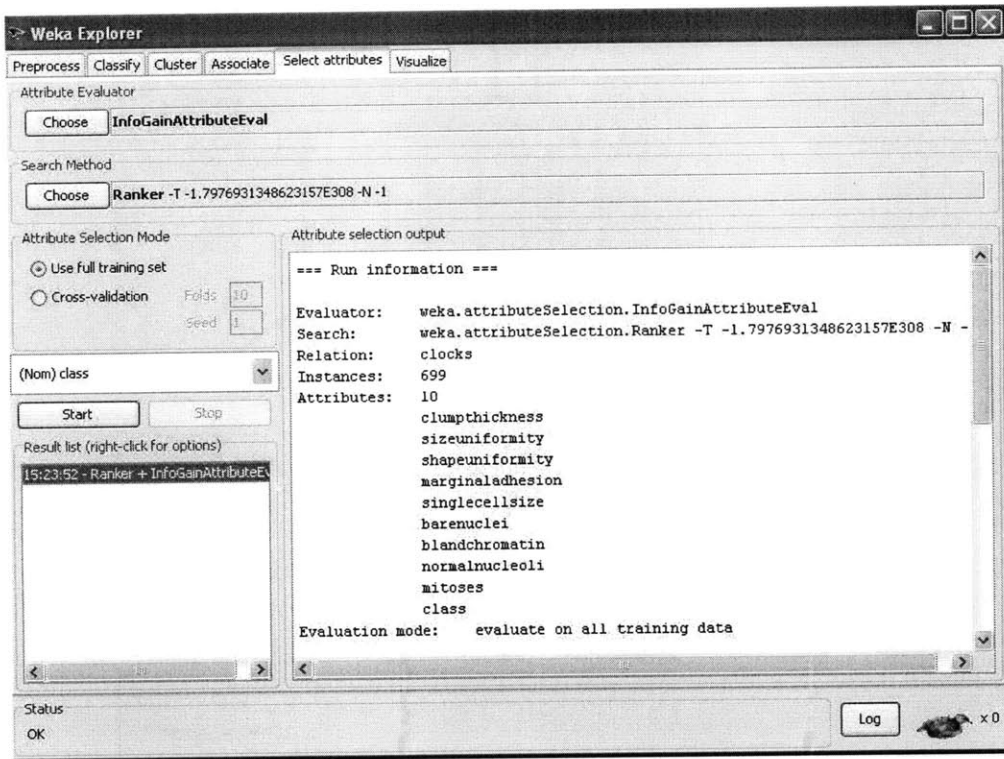


Figure 8.6: Weka- Select Attributes;

Visualize

Finally, the “Visualize” tab allows the user to see pairwise graphs of the features. For example, for the cancer dataset the user might want to see how sizeuniformity changes with shapeuniformity. Many small graphs are presented, as shown in Fig 8.7 below, but by double-clicking the user can see an enlarged graph. (Fig 8.7, 8.8)

C.2 Databionics ESOM Analyzer

The ESOM (emergent self-organizing map) tool provided in the CDT Tools Package has a more focused functionality than the WEKA tool. An ESOM is a projection of high-dimensional space onto a two-dimensional “map”, with distances in the higher-dimensional space indicated by differing colors on the map. The major advantage to using this tool for data analysis is that it provides a detailed map of the data and a visual representation of any clustering that occurs in the data. The program requires the user to load three files, all of which are provided by the file creation functionality of the MDA Tool. One file, with a .lrn extension, contains the data to be analyzed, while the .csl and .names files contain the class labels of the data in numerical and string format, respectively. The first step that the

8. APPENDIX

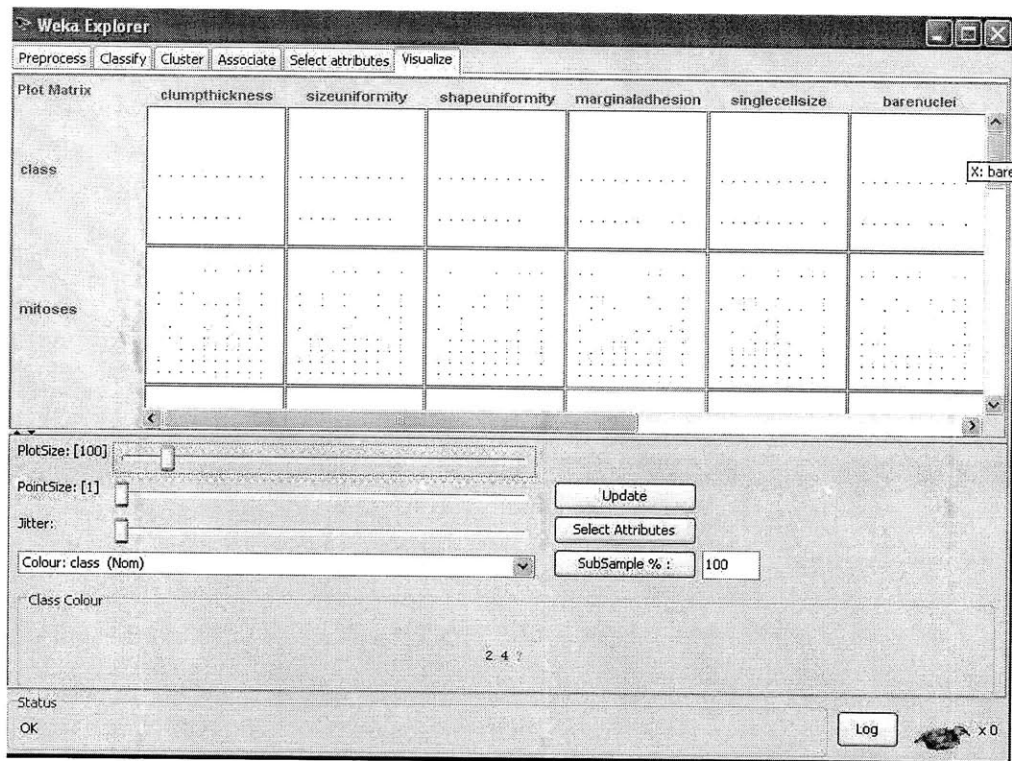


Figure 8.7: Graphs for Pairwise comparison of features;

user should take is to load each of these files, starting with the .lrn file. This can be done under the File tab. The second step is to train a self-organizing map on the data, which can be done by selecting Tools, then Training. The user will be able to edit many parameters of the ESOM, and then click on Start at the bottom of the screen. In general, the default parameters work just fine for a first run on the data. After training is completed, the user should click on close and return to the main screen. The map should appear in the upper panel of the user interface.

(Figure 8.9)

Each dot on the map represents a sample. Colors on the map indicate elevation; the higher the elevation between two points the further they are apart in higher dimensional space. For this example, green colors indicate low elevation and brown/white indicates high elevation.

In the lower part of the user interface, there are many tabs that can be selected for alternate views of the model. A few important ones are highlighted below.

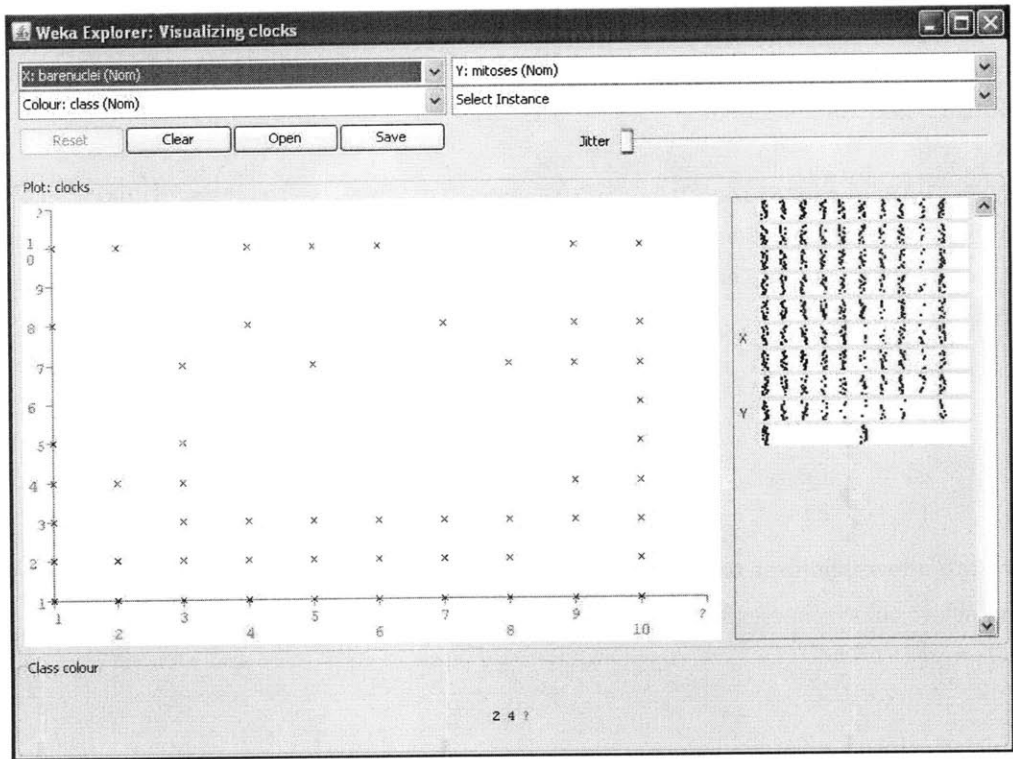


Figure 8.8: Enlarged Pairwise Comparison Graph;

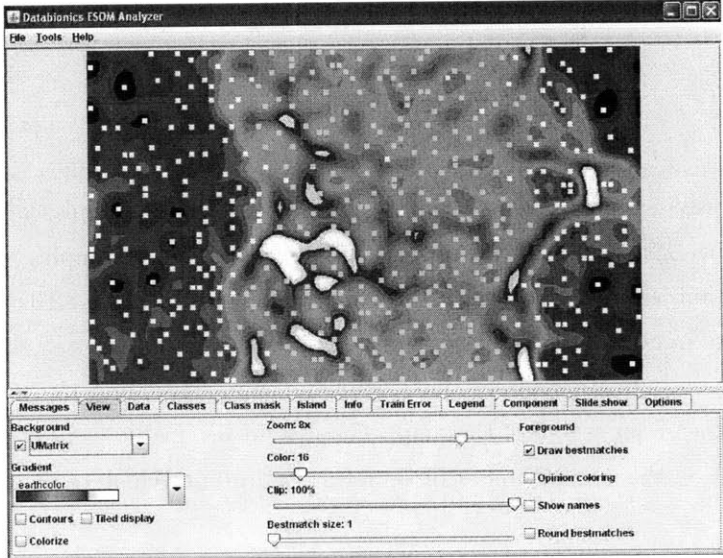


Figure 8.9: ESOM trained on cancer data;

Messages

The “Messages” tab shows output from each task. For example, if a file is not loaded correctly, the user will be able to see why by reading the messages recorded in this tab. An

8. APPENDIX

example of messages recorded while the ESOM is being trained is shown below.

```
training time: 206.562 seconds
100%
saving file: C:\Program Files\ClockDataAnalysis\ClockData_1256776721.06\...bm
Setting file C:\Program Files\ClockDataAnalysis\ClockData_1256776721.06\...bm
saving file: C:\Program Files\ClockDataAnalysis\ClockData_1256776721.06\...umx
exit called!
Training is done.
```

View

The “View” tab allows the user to zoom in/out on the map, or create a tiled version. Also, the user can select color schemes for the ESOM.

Data

This section is where the data can be seen by the user. It is a good idea to check here and make sure that data has been loaded correctly, especially the column indicating the class of each sample.

Classes

The “Class” section provides an easy way for the user to see where samples of the same class are located on the ESOM. By selecting 2 for example, all of the samples that are in class 2 are highlighted on the map. An example of this is shown in Figure 8.10. (note that samples labeled “2” are in class 0 in this case, while samples labeled “4” are in class 1.) All of the bold white squares indicate samples that are of the same class, and they seem to be very well clustered in the green region of the map. The ESOM tool works very well with the cancer dataset, as seen by the small number of samples that are not clustered with samples of the same class.

C.3 Cluster 3.0

The clustering tool provided in the CDT Analysis Package was originally developed for clustering of DNA microarray data, and therefore has the capabilities required to handle large datasets. For this reason, some of the filtering functionality will not be routinely used

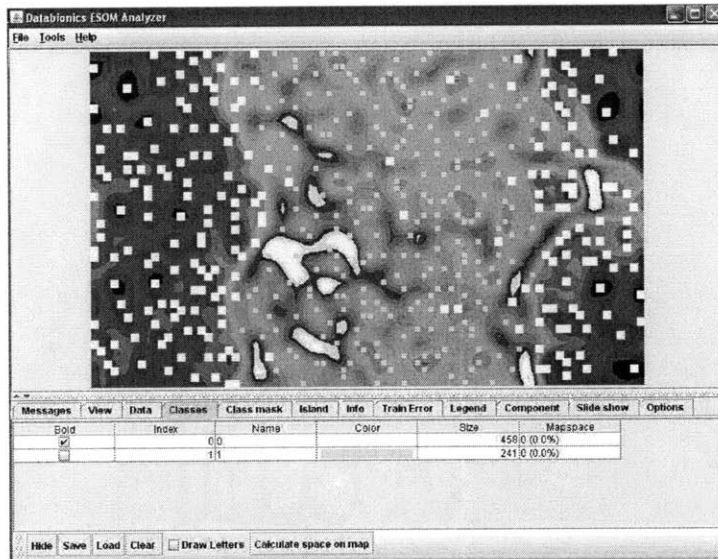


Figure 8.10: Class Two Sample Highlighted;

on CDT data. In addition, whenever the program refers to “genes” it is referring to rows of data. “Arrays” refers to columns of the data. When the program launches, the user interface will appear as in Fig 8.11. The first thing to do is load the data file, which should be in the form of a .txt file in the Cluster directory produced by the MDA tool. If the file is successfully loaded, then the top panel of the user interface will show the file name.

Adjust Data

After the data is loaded, the user can adjust the data if desired. Options given in the second tab include the log-transform, centering data, or normalizing data. If centering or normalizing are chosen for “gene”, this will perform the operation across a row of data. Centering or normalizing the “array” means doing so for the columns of data.

K-means

Although other clustering methods are provided that relate specifically to microarrays, the k-means, pca, and SOM algorithms presented in Cluster 3.0 are general enough to work well with CDT data. For k-means clustering, the user should select “organize genes”, and then choose the number of clusters and runs desired. The typical distance measure for k-means is euclidean distance, however the user can select a different metric as needed. Once “Execute” is pressed, the clustering will occur and the user will see a message (Solution Found X Times) at the bottom left of the interface when it has completed. Two files are put in the same

8. APPENDIX

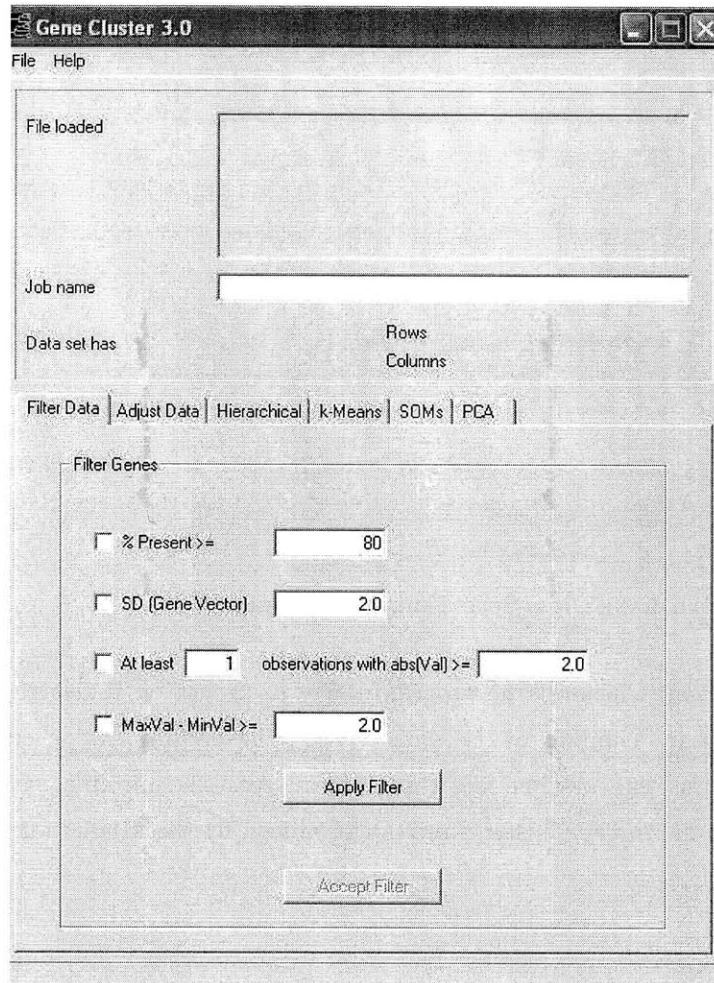


Figure 8.11: Cluster 3.0 Upon Launch;

directory as the original data file; the .kcg file contains the classification of each sample, and the .cdt file contains the original data reordered into the same order as the .kcg file. It is the .kcg file which is most useful, because the user can compare output with the true classes to determine how well the clustering algorithm performed. This file will appear with two columns, as below, the first containing the sample number and the second containing the class it was assigned to.

```
Fun GROUP
2 0
6 0
7 0
15 0
19 0
```

21 0
22 0
24 0
26 0
39 0

In this case, these first samples were all placed in the same class, class 0. A comparison with the true class labels on the samples indicates that 8 out of 10 of these samples really belong to the same class. The clustering functionality of this tool is especially useful because the user can edit all of the parameters- distance metric, number of clusters, and number of runs.

SOM

This clustering tool also provide SOM functionality as in the ESOM tool, but with far fewer editable parameters. To use this functionality, options in the “genes” panel should be selected and then “Make SOM” pressed. The .gnf file that is output will contain the location in high-dimensional space that maps to each x-y grid coordinate on the map. There is no visualization aspect to this tool, so using the Databionics tool for self-organizing maps is preferable.

PCA

The PCA tab has one button, “Execute”. This will create the matrix factorization of the data, and .svu and .svv files will be created. Cluster 3.0 does not provide functionality allowing the user to select parameters such as the desired number of principle components, so the PCA or Weka tools are preferable for this functionality.

C.4 BSVM

Figure 8.12 shows the appearance of the BSVM interface when the program is launched.

The first step in using the BSVM tool is to train a model. This should be done by loading the input file out of the bsvm folder created by the MDA Tool, and selecting a location for the model file. Default parameters will load upon program launch, but you can edit these parameters to choose the type of SVM, kernel type, and values for 12 other parameters. Once the parameters are set, click on “Model” to create the model file. The “Restore Defaults” button will clear any parameters that you have edited and return to the defaults.

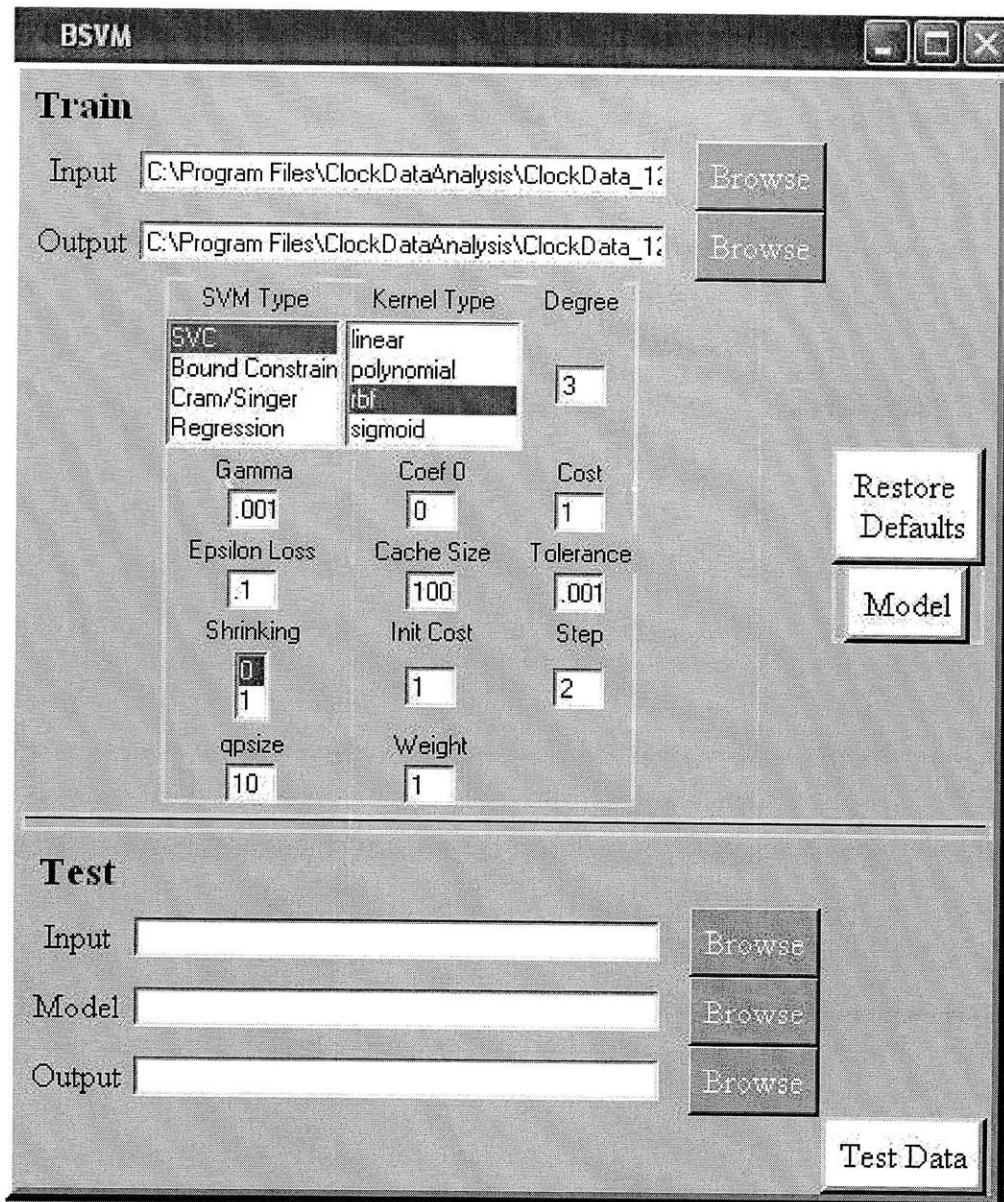


Figure 8.12: BSVM Tool;

Once a model has been created, it can be used to classify samples. In the “Test” part of the interface, you can load an input file, the model file, and an output file. The input file should be the same format as the input file used in training, but may contain different data. It is common to train on a large percentage of the data and save a small number of samples to be used to test the accuracy of the model. The model file is the same file output from the training stage, and the output file is where you would like the output to be stored. Once “Test Data” is pressed, an output file will be generated that contains the class assignment

for each sample in the input file. Comparing these classes to the actual class of the samples will give a good idea of how well the classifier is performing.

C.5 PCA

PCA will reduce the dimensionality of data by creating new features that are combinations of the original features. For example, the first feature in the new dataset might be $.4 * barenuclei + .3 * sizeuniformity$. Figure 8.13 shows the interface for PCA.

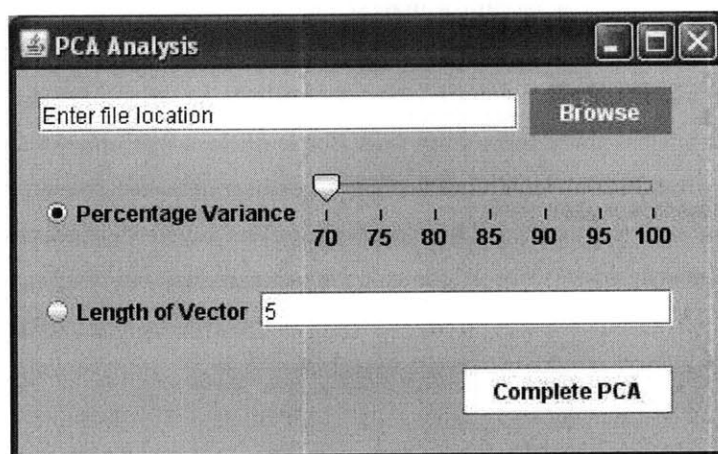


Figure 8.13: PCA;

After loading the data file from the `pca` folder, the user has two options for how data will be reduced in the `pca` algorithm. The first option is to choose the percent of the variance across samples in the original data to be accounted for. This can be done by selecting the button next to “Percentage Variance” and moving the slider to the desired percentage. A second option is to specify the number of new features that are desired. This can be done by selecting “Length of Vector,” and typing the number of desired features in the corresponding box. The first option is the most commonly used, but the second option is useful when data needs to be reduced to a specific size requirement.

After this choice is made, click on “Complete PCA.” Two files will be placed in the same directory as the input file; both will have the same name as the input file, but one will end in `-Out` and the other will end in `-Weights`. The first is the output file, which will contain the original data reduced to a lower dimensionality. The other file is the weights file, which contains the weight of each original feature present in each of the new features. The output file from the PCA tool in the correct format to be used as input to the Cluster 3.0 tool.

C.6 EPCA

EPCA is similar to PCA, but takes into account the fact that features are not all real-valued. For more information on the EPCA algorithm, see Collins et. al. The algorithm used by MDA is similar to that developed in the Collins paper, but allows features to take on different data types within the dataset. (Figure 8.14)

The process of reducing the dimensionality of the data involves choosing the dimensions ahead of time, and then attempting to find a matrix factorization that minimizes the loss between the product of two smaller matrices and the original matrix. For PCA, the loss function that is used is minimum mean squared loss. The EPCA algorithm described in the Collins paper uses Poisson loss for integer data and Bernoulli loss for binary data. In my implementation, either minimum mean squared, poisson, or bernoulli loss is used depending on the type of each column of the matrix. Therefore, binary and integer features are treated differently than real-valued features when it comes to measuring loss.

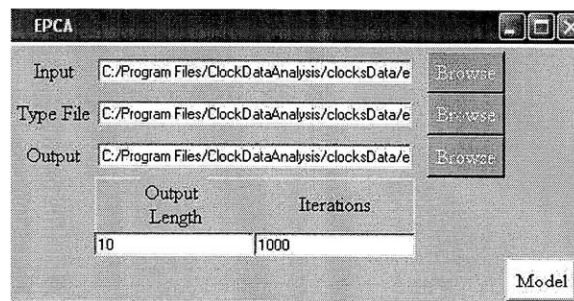


Figure 8.14: Exponential Principle Components Analysis;

First, the user must select the input file, which is located in the epca folder created by the MDA. The type file, which lists the type of each feature in the dataset, also must be loaded from the epca folder. The third file name required is the output file, and the user can choose where to place this file. There are two editable parameters for the epca algorithm- output length and number of iterations. Output length is dimension of the output data, the number of new features that are found. Number of iterations refers to how many loops the algorithm runs in attempt to optimize the solution before quitting.

Once the parameters are set, press the “Model” button. This is a slow algorithm, especially on large datasets. It takes about 2 minutes to run 10 iterations on a dataset with 90 samples and feature length 1000 on a 2.4 GHz Intel Processor. Once the optimization is complete,

the output will be written to a file in the same directory as the input file. As with the output from the PCA tool, this output can be used as input to Cluster 3.0.