AN INVESTIGATION AND VISUALIZATION OF MICRORNA TARGETS AND GENE EXPRESSIONS AND THEIR USE IN CLASSIFYING CANCER SAMPLES

A Thesis

Presented to

The Graduate Faculty of the University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Jarod Rose

May, 2011
AN INVESTIGATION AND VISUALIZATION OF MICRORNA TARGETS AND GENE EXPRESSIONS AND THEIR USE IN CLASSIFYING CANCER SAMPLES

Jarod Rose

Thesis

Approved:

Advisor
Dr. Zhong-Hui Duan

Committee Member
Dr. Richard Londraville

Committee Member
Dr. Yingcai Xiao

Accepted:

Department Chair
Dr. Chien-Chung Chan

Dean of the College
Dr. Chand K. Midha

Dean of the Graduate School
Dr. George R. Newkome

Date
ABSTRACT

Classifying cancer can be a challenging task even for the best medical doctors, but by doing so increases the likelihood that a specialized treatment can be devised to cure that type of cancer. Today’s computers are increasing in complexity and speed and these resources can be utilized to help us locate and understand specific traits of cancers that will help us classify them faster, and with better accuracy in the future. This thesis develops a software system that look at specific traits of both cancerous and normal tissue samples, focusing on microarray gene expression data and microRNA targets. In the thesis, a new average linkage based classification method is developed and compared with standard k-nearest neighbor method. Two visualization tools were developed. The first tool is a gene ontology tree analyzer which includes two components, mapping microRNA targets to a tree structure and mapping the targets to a directed acyclic graph. The analyzer allows users to navigate information pertaining to the microRNA targets and their gene ontology information. The second tool is a 3D viewer for microarray data. The cancerous tissue samples and normal tissue samples are projected to a three dimensional space based on the microarray gene expression values in the samples. My results indicate that the new cluster based classifier outperforms the standard k-nearest neighbors in terms of classifying cancer tissues based on the microarray data. The visualization tools developed in the thesis provide insights into understanding the gene
ontology functions associated different cancers and visualizing the relationships among different cancer samples and normal samples.
ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor, Dr. Zhong-Hui Duan, for her constant encouragement and guidance during this study. I would also like to extend my gratitude to Dr. Richard Londraville and Dr. Yingcai Xiao for taking time serving on my committee and providing invaluable suggestions. I couldn’t have completed this thesis without their help.

I would also like to thank Dr. Anton Enright’s group at EMBL-European Bioinformatics Institute for kindly providing me a copy of the Sanger microRNA database, which is used for the MicroCosm web resource. Without a local copy of their database, it would have taken me a much longer period of time to conduct the experiments required in this study.

Finally, I would like to thank my family and friends for their moral support throughout this study. Without their constant support, I wouldn’t have been able to make it to where I am today.
TABLE OF CONTENTS

LIST OF TABLES ............................................................................................................. ix
LIST OF FIGURES ......................................................................................................... x

CHAPTER

I. INTRODUCTION ........................................................................................................ 1

1.1 Cancer Classification Techniques ................................................................. 1

1.2 MicroRNAs ....................................................................................................... 2

1.2.1 Sanger MicroRNA Database ..................................................................... 3

1.3 Microarray ......................................................................................................... 3

1.4 Analyzing Microarray Data ............................................................................ 5

1.5 Gene Ontology ................................................................................................ 5

1.6 Visualization .................................................................................................... 6

1.6.1 Java3D .................................................................................................... 7

1.6.2 Singular Value Decomposition .................................................................. 7

1.7 Literature Review ............................................................................................ 7

1.8 Objective ......................................................................................................... 9

II. MATERIALS AND METHODS ........................................................................... 11
2.1 MIT Dataset .......................................................................................................... 11
2.2 Sanger MicroRNA Database ................................................................................ 12
2.3 Data preprocessing .............................................................................................. 13
2.4 K-Nearest Neighbor Algorithm ......................................................................... 14
  2.4.1 Euclidean Distance ...................................................................................... 15
  2.4.2 Manhattan Distance ................................................................................... 15
  2.4.3 Chebychev Distance ................................................................................... 16
  2.4.4 Pearson Distance ....................................................................................... 16
2.5 Singular Value Decomposition .......................................................................... 17
2.6 Tree Rendering Methods .................................................................................... 17
III. IMPLEMENTATION .............................................................................................. 19
  3.1 Preprocessing .................................................................................................. 19
  3.2 kNN Classifier ................................................................................................ 21
  3.3 SVD .................................................................................................................. 21
  3.4 3D Viewer ........................................................................................................ 22
  3.5 Database .......................................................................................................... 23
  3.6 GO Tree Mapping ............................................................................................ 24
    3.6.1 Java’s JTree .............................................................................................. 24
    3.6.2 Tree Customized DAG Viewer .................................................................. 26
IV. RESULTS AND ANALYSIS ................................................................................... 31
4.1 Preprocessing Results ........................................................................................... 31

4.2 Classification Results and Analysis ...................................................................... 32

4.3 SVD Results .......................................................................................................... 36

4.4 GO Tree Analysis .................................................................................................. 39
  4.4.1 JTree ............................................................................................................ 39
  4.4.2 DAG ............................................................................................................ 42
  4.4.3 Comparison ................................................................................................. 44

V. CONCLUSIONS AND FUTURE WORK ................................................................. 45
  5.1 Conclusions ...................................................................................................... 45
  5.2 Future Direction ............................................................................................... 46

VI. REFERENCES ........................................................................................................ 47
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An example of the data collected from a microarray chip</td>
</tr>
<tr>
<td>2.1</td>
<td>The distribution of tissue samples used by Lu et al</td>
</tr>
<tr>
<td>2.2</td>
<td>An example of input data to the classifier</td>
</tr>
<tr>
<td>4.1</td>
<td>Results from kNN using the Pearson distance metric with k=3</td>
</tr>
<tr>
<td>4.2</td>
<td>Results with the Pearson distance metric using the clustering approach</td>
</tr>
<tr>
<td>4.3</td>
<td>Classification accuracy for all of the distance metrics tested</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An example of a GO definition that was created by the GO Consortium.</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>An example of generated DAG.</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Using the standard kNN classification method.</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Using the cluster based classification method.</td>
<td>34</td>
</tr>
<tr>
<td>4.3</td>
<td>An example of the 3D visualizer.</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>A GO Tree generated using the tree structure provided by core Java libraries.</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>A DAG demonstrating the resolution of multiple parents.</td>
<td>43</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

1.1 Cancer Classification Techniques

When attempting to identify and classify cancer there are a few common techniques that oncologists and biologists can use. There are visual examinations of the tissue itself, a test can be done on the individual who is thought to be afflicted by the cancer, or genetic testing can be done on the cancer. It doesn’t matter what technique is used, the main goal is to find identifying traits of a cancer so that it can match the treatment that will be most effective against it. The difficulty of finding these traits lies in the similarities of some cancer subtypes. For example there are several subtypes of leukemia, two of which are Acute Myelogenous Leukemia (AML) and Acute Lymphocytic Leukemia (ALL). These types of cancer are easily identifiable as leukemia, but have different treatments that are the most effective against them. To complicate matters there are a lot of similarities between AML and ALL to the point where accurately determining the difference between the two becomes a challenge [6].

When analyzing cancer, the more information that we can acquire, the more accurately we can identify it. So, for the time being, the most information we can get from a cancerous sample is the information from the genes that are being expressed by the cancer. Since the expression of genes is the catalyst that drives biological function then knowing how they are expressed in different tissues, both cancerous and non-
cancerous, will help us find out what is unique about each type of cancer and perhaps find better ways to treat them [6].

More recently, the discovery of the role of microRNA in cancer has opened up a new world of possibilities for classifying cancer. Through the work of Lu et al. [11] and other research groups more information has been gathered on the role of microRNAs in the formation of cancerous tissues. This has lead to new promising ways of classifying cancers that could add more information to cancer diagnostics. Since the discovery of the role of microRNA in cancer formation and progression is fairly recent, there is still a lot that we can learn from it and it makes it a very interesting topic to delve deeper into.

1.2 MicroRNAs

MicroRNAs, or miRNAs, are short RNA sequences that are in the range of 20 to 25 nucleotides long. Unlike other forms of RNAs, miRNAs are not used to code for protein sequences. Instead miRNAs are used to regulate gene expression throughout the cell. Lu et al. [11] published a paper on miRNAs and their uses in classifying cancer. They noted that the first identified miRNAs lin-4 and let-7 have important roles in controlling developmental timing and probably act by regulating message RNA (mRNA) translation. When lin-4 and let-7 are inactivated, specific epithelial cells undergo additional cell division instead of their normal differentiation. Since abnormal cell proliferation is a hallmark for human cancers it seems possible that miRNA expression patterns might denote the malignant state [11]. The genes that lin-4 and let-7 affect are known as targets [11]. Targets are simply mRNAs that are affected in some way by the presence of the miRNA. This could be a positive affect where the mRNAs are coded
more frequently, or a negative affect where the mRNAs are coded less frequently, or the translation of the target mRNAs can be blocked

1.2.1 Sanger MicroRNA Database

The Sanger miRNA database provides a resource for identifying targets of miRNA sequences [17]. This database, available through a web interface developed by Enright Lab [4] called MicroCosm Targets [12], allows a user quickly search for a particular miRNA and retrieve targets targeted by the miRNA. This database contains a vast amount of information regarding miRNAs from different species, and what targets they hybridize to, providing information on sequence alignment as well as gene ontology information and other features of the miRNA targets [17]. This database is an ideal centralized source for information that can be used to examine potential interactions between miRNAs and cancer and to see if there is a relationship that can be used to help classify cancers. The Sanger MicroRNA database was kindly provided by Enright lab to facilitate this thesis study.

1.3 Microarray

To determine the expression levels of genes one needs to be able to find a way to gauge how many RNAs transcribed by the genes are present without physically counting each and every one. To do this the microarray technology was developed to sort the RNA segments into separate clusters [2, 13]. A microarray works by placing DNA fragments called probes in a small spot on the chip and each spot is arranged in array with each spot corresponding to a specific gene of biological importance. The probes contained in the spots are all nucleotide chains that code for the compliments of the RNA
probes. With the probe spots arranged on the chip, the solution that contains the genetic sample that one wants to get the expression information for can be applied. The genetic materials contained in the solution are died with florescent dyes that allow them to be easily seen when they are attached to the probes. Once the genetic materials in the solution have had sufficient time to bind with the probes the chip can be placed into a device that will examine each probe and measure the intensity of the florescent dye. The higher the intensity of a spot is, the higher the expression of the corresponding gene is in the original sample [10]. This measurement of expression levels can be repeated on multiple samples to create a matrix as seen in table 1.1.

<table>
<thead>
<tr>
<th>Description</th>
<th>N_COLON_1</th>
<th>N_COLON_3</th>
<th>...</th>
<th>T_BRST_4</th>
<th>T_BRST_5</th>
<th>T_BRST_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsa-let-7a</td>
<td>12.0092</td>
<td>14.4779</td>
<td>...</td>
<td>10.038</td>
<td>11.7298</td>
<td>12.0928</td>
</tr>
<tr>
<td>hsa-let-7b</td>
<td>12.0597</td>
<td>11.9112</td>
<td>...</td>
<td>11.9322</td>
<td>12.2383</td>
<td>12.1824</td>
</tr>
<tr>
<td>hsa-let-7c</td>
<td>11.9289</td>
<td>11.6147</td>
<td>...</td>
<td>11.1039</td>
<td>11.9222</td>
<td>12.1018</td>
</tr>
<tr>
<td>hsa-let-7d</td>
<td>9.47273</td>
<td>7.33001</td>
<td>...</td>
<td>5</td>
<td>7.46975</td>
<td>9.48947</td>
</tr>
<tr>
<td>hsa-let-7e</td>
<td>8.87858</td>
<td>7.16847</td>
<td>...</td>
<td>5</td>
<td>6.55037</td>
<td>7.91539</td>
</tr>
<tr>
<td>hsa-let-7f</td>
<td>10.8519</td>
<td>9.85145</td>
<td>...</td>
<td>7.85467</td>
<td>9.19339</td>
<td>10.2359</td>
</tr>
<tr>
<td>hsa-let-7g</td>
<td>10.6503</td>
<td>9.87166</td>
<td>...</td>
<td>8.56699</td>
<td>8.72912</td>
<td>10.0011</td>
</tr>
<tr>
<td>hsa-let-7l</td>
<td>9.53855</td>
<td>8.82365</td>
<td>...</td>
<td>7.53378</td>
<td>7.46628</td>
<td>9.66012</td>
</tr>
<tr>
<td>hsa-miR-1</td>
<td>11.5344</td>
<td>11.4035</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>hsa-miR-100</td>
<td>9.26146</td>
<td>8.3682</td>
<td>...</td>
<td>7.2663</td>
<td>6.05682</td>
<td>7.26978</td>
</tr>
<tr>
<td>hsa-miR-101</td>
<td>9.4876</td>
<td>9.47133</td>
<td>...</td>
<td>8.1045</td>
<td>7.7994</td>
<td>8.1944</td>
</tr>
<tr>
<td>hsa-miR-103</td>
<td>9.93602</td>
<td>8.83539</td>
<td>...</td>
<td>9.63715</td>
<td>8.88582</td>
<td>9.2322</td>
</tr>
<tr>
<td>hsa-miR-105</td>
<td>5.8207</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>rno-miR-335</td>
<td>6.8121</td>
<td>5.6345</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5.2822</td>
</tr>
<tr>
<td>rno-miR-336</td>
<td>5</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>rno-miR-343</td>
<td>7.4709</td>
<td>7.141</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5.4027</td>
</tr>
<tr>
<td>rno-miR-344</td>
<td>5.4458</td>
<td>5</td>
<td>...</td>
<td>5.0809</td>
<td>5.0352</td>
<td>5</td>
</tr>
<tr>
<td>rno-miR-346</td>
<td>5</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>rno-miR-347</td>
<td>5.0897</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>rno-miR-349</td>
<td>5</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>rno-miR-352</td>
<td>9.5504</td>
<td>7.96656</td>
<td>...</td>
<td>6.48227</td>
<td>8.43188</td>
<td>8.62774</td>
</tr>
<tr>
<td>rno-miR-7*</td>
<td>5.1323</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
1.4 Analyzing Microarray Data

Once the expression levels are collected for the multiple tissue samples, the data needs to be examined for any useful information that it may contain. One popular approach is to apply data mining techniques to microarray data to see if there are any patterns that emerge [6, 11]. Since the overall goal with collecting expression information is to be able to classify unknown samples of cancer we can focus on a subset of data mining techniques that lend themselves to classifying data points which include, among other techniques, neural networks [14] and nearest neighbor classifications [3].

The overall goal with nearest neighbor classification is to determine the class of a data point based on the classes of the surrounding data points. To do this one needs to look at the classes of the points that are closest to the point in question. The point in question’s class then ends up being the same as the class with the most number of points that are considered close to it. There is a lot of flexibility when it comes to determining how close one data point is to another and what the most data points means, allowing nearest neighbor algorithms to be applied to many different fields. In section 2.4, I will go into more detail of how the nearest neighbor algorithm works and discuss some of the different ways one can modify it to get different results.

1.5 Gene Ontology

Gene Ontology (GO) is a way to develop controlled vocabularies for gene products [16]. The products that a gene produces consist of any RNA or protein that a gene may produce. The Gene Ontology project was created to standardize these descriptions and assign annotated identifiers to them. By doing this they allow for
queries to be performed across different databases. Each GO annotation is identified by a GO ID, the GO ID is a unique identifier for the annotation but play no role in the actual description of the annotation. The project is managed by the GO Consortium, a group of universities that contribute and maintain the new GO annotations that are being discovered [15]. An example of a GO Term definition can be seen in figure 1.1.

<table>
<thead>
<tr>
<th>[Term] id: GO:0015371</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: galactose:sodium symporter activity</td>
</tr>
<tr>
<td>namespace: molecular_function</td>
</tr>
<tr>
<td>def: &quot;Catalysis of the reaction: galactose(out) + …</td>
</tr>
<tr>
<td>xref_analog: TC:2.A.21.3.-</td>
</tr>
<tr>
<td>is_a: GO:0005354 ! galactose transporter activity</td>
</tr>
<tr>
<td>is_a: GO:0015370 ! solute:sodium symporter activity</td>
</tr>
</tbody>
</table>

Figure 1.1 An example of a GO definition that was created by the GO Consortium.

1.6 Visualization

To better understand the clustering of the microarray data, a 3D visualization tool was developed to display the microarray data in a way that could help in understanding how samples from different types of cancer and normal tissue cluster together. Since microarray data is highly dimensional by its nature, a compression method was needed to compress the information into three dimensions so it could be displayed to the user in a concise way. To perform the data compression, singular value decomposition was used on the microarray data.
1.6.1 Java3D

A major component in the visualization, Java3D, is an open source library developed by the java.net which is an online community of researchers, technicians and evangelist at Sun Microsystems as well as the general public working together on interesting projects that use the Java language [8]. This library allows for a wide range of 3D rendering applications including, but not limited to, game design and 3D medical visualization. For this project the full depth of Java3D was not explored, however, some of the core 3D functionality was utilized.

1.6.2 Singular Value Decomposition

Singular Value Decomposition (SVD) is a well known mathematical method of factoring a matrix into three separate matrices [5]. Two of the matrices are unitary matrices and the third is a diagonal matrix. Once the matrix is factored, it can be used for many purposes, including signal processing, matrix approximation and microarray data analysis [1, 19]. For the purposes of this paper we utilized SVD to compress microarray data in the form of an m x n matrix down into an m x 3 matrix that can be graphed visually, where m represents the number of samples and n stands for the number of genes (features). To do this we chose the columns that have the highest corresponding singular-values in the diagonal matrix. These columns give us a compressed matrix with the closest approximation to the original microarray data.

1.7 Literature Review

The use of miRNAs to classify cancers is still a fairly new concept. Recently patterns that miRNAs exhibit in different types of cancerous tissues as well as normal
tissue have been explored. One of the patterns that have been examined is how different miRNAs are being expressed in the samples. If a particular miRNA is being over or under expressed consistently in certain samples then it can be used as a key to help identify cancers in the future. Among the research studies on the expression levels of miRNA in different tissue samples there are two in particular that I would like to discuss. The first study was performed by Lu et al. [11] and the second performed by Volinia et al. [18].

The bulk of the research done by Lu et al. was an attempt at finding a better way to determine the expression patterns of miRNAs in cells and how they relate to cancer growth. At the time, microarray technology didn’t work as well and usually caused noisy data. Their method, using bead-based microarrays, proved to be very useful and they were able to get good miRNA expression data. Their result show that miRNA expression patterns reveal the development history of tissue samples under study.

This led them to discover that miRNAs could be used to discover the developmental origin of cancerous tumors and to differentiate between normal and tumor samples. MiRNAs in cancerous tissue showed overall lower expression levels in cancerous tissues than in normal tissues. They validated the procedure with mouse samples and got the same results.

Volini et al. [18] further examined the different miRNA expression patterns that existed in solid tumors. They examined 540 samples from six solid tumors and analyzed the miRNA expression levels. They compared these expression levels to levels of normal tissue samples and obtained a set of 26 over expressed and 17 under expressed miRNAs. With these miRNAs identified Volinia et al. wanted to refine their results because the
miRNAs that they identified encompass all of the tissue samples that they used. The reason being is that because miRNA expression is tissue specific, the group of miRNA that they discovered might not be the best at classifying all solid tumors. To refine their results Volinia et al. used an independent test to identify the tissue samples that were determined to be tumors. These samples were paired with their normal counterparts and the miRNA expression levels were compared. From this work they managed to identify several miRNAs commonly over expressed among most of the tumor samples.

The over expression of the miRNAs that Volinia et al. [18] discovered contradicts the under expression that Lu et al. [11] discovered during their research. Volinia et al. discussed this and examined some key differences between their research and the research done by Lu et al. One notable difference is that the results from Lu et al. identified several miRNAs, in particular miR-15-5p, miR-20, and miR-92 to be down regulated in tumors while the results from Volinia et al. [18] identified them as upregulated.

The fact that the results of these two studies ended up with different results shows that the study of miRNA expression levels in cancer using microarray technology still has a long way to go before the technology becomes refined enough to provide results that are consistent across studies. However, microarray offers a unique approach to examine the genomic-wide expression levels of miRNAs.

1.8 Objective

The objective of this research is to develop a software tool that can facilitate biologists to analyze and visualize microarray gene expression data and find a link
between miRNAs and the gene ontology annotations of the RNA targets of the miRNAs. To do this I examined miRNA expression data acquired from the work of Lu et al. With this data, I explored several different approaches for classifying cancerous tissue samples. I also obtained the targets of microRNAs based on Sanger MicroRNA database. The link between the targets of miRNAs and their gene ontology annotations were then established using the gene ontology file provided by Gene Ontology Consortium. With the gene ontology annotations one can then see if a pattern emerges with the annotation that might give us some insight into what the roles of miRNAs are in cancer formation and progression. Also, I looked at different ways to visualize the microarray data and gene ontology data, which might help bring insight into different paths that we can take to analyze the microarray data further and perhaps find new ways of classifying cancerous tissue samples.
CHAPTER II
MATERIALS AND METHODS

2.1 MIT Dataset

To be able to classify cancerous tissue a dataset that contains sufficient information about the tissues to be classified is required. In this case, the dataset that was generated by Lu et al. was used since it primarily focuses on the expression of miRNAs in different tumor and normal tissue samples. The dataset is simply a 2 dimensional array of values where the x-axis represents the sample collected and the y-axis represents the expression of a particular miRNA. Samples are also grouped based on the type of tissue from which they were collected from so that any patterns in the data can be more easily recognized, the distribution of these groups can be seen in table 2.1. When preprocessing the data I took advantage of the grouping of the samples to help try and find patterns that may only occur between particular tissue types.

Table 2.1 The distribution of tissue samples used by Lu et al.

<table>
<thead>
<tr>
<th>Normal Colon</th>
<th>Cancerous Colon</th>
<th>Normal Pancreas</th>
<th>Cancerous Pancreas</th>
<th>Normal Kidney</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cancerous Kidney</th>
<th>Normal Bladder</th>
<th>Cancerous Bladder</th>
<th>Normal Prostate</th>
<th>Cancerous Prostate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cancerous Ovary</th>
<th>Normal UT</th>
<th>Cancerous UT</th>
<th>Normal Lung</th>
<th>Cancerous Lung</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cancerous Meso</th>
<th>Melanoma</th>
<th>Normal Brest</th>
<th>Cancerous Brest</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
2.2 Sanger MicroRNA Database

When examining the relationship between the miRNA and the functions performed by its targets, a data source that contains all of the currently known targets and any other relevant information would be useful. With the help of Stijn van Dogen from EMBL-EBI I was able to get a copy of the current database that was used as the data source for the miRBase Targets application, now called MircoCosm Targets. Having a local copy of this database greatly expedited the development of a tool to help with the examination of the miRNA targets by allowing me to more closely examine the schema of the database to best determine how to construct queries to extract the desired information.

Looking closer at the database tables that are used by MicroCosm Targets there were several tables that were of interest to this project. The first table contained information regarding an miRNA and its targets. In particular the first table contained the identifier (ID) of the miRNA and the ID of the target. Another table contained more information pertaining to the untranslated regions that the miRNA hybridize to, including the external name of the gene, the sequence of the region, and a short description of the region. The last table that I utilized contained varying properties that a mRNA may have. In my case, the feature that I’m most interested in is the GO terms associated with the mRNA so that I can see what biological functions this gene controls.

To gather this information from these tables all I needed is to get the transcript ID associated with the mRNA ID. The transcript ID acts like a unique identifier for an mRNA and it allowed me to get the relevant information from the database. From the rest of the database, I was able to get the external name as well as the GO IDs associated
with the targets. Since the GO IDs by themselves aren’t very informative I used a
different table that I acquired from the gene ontology website to get the relevant
information about the GO ID [15].

2.3 Data preprocessing

Patterns in microarray data that could help lead to advances in the ability to
detect cancerous tissues accurately. Some preprocessing was performed on the dataset
acquired from Lu et al. so that it could be manipulated easier. With this dataset there
were two main steps taken to help the analysis process. First, the dataset was divided into
the pairs of normal tissue samples and cancerous samples and 156 t-tests were performed
taking the top samples with p < .0002. The top results from each paring were merged
back together into the original data set of all samples. Again, the t-test was performed on
these samples and the resulting set had p < .034.

Other preprocessing was also performed for readability. For example, the names
of the miRNA IDs were shortened to just be the name of the miRNAs that were being
tested for. The database acquired from Sanger contained genomes other than human, to
reduce the data overhead and improve database performance; any non-human genome
related data was removed. An abbreviated version of the data in its final form can be
seen in table 2.2. With the data in a more manageable form, the actual analysis of the
data can be accomplished in a more programmatic fashion. In the coming sections I
discuss the types of processing that were done on this data and how that processing was
accomplished.
2.4 K-Nearest Neighbor Algorithm

The k-nearest neighbor algorithm is a simple machine learning algorithm that uses a majority vote system to classify a given object based on the information of the neighbors surrounding it. The ‘k’ in the k nearest neighbor algorithm represents the number of neighbors that are to be considered when classifying a given object. Depending on the value of k one will get different results. If k is small, the boundaries between different classes will be clearer but the results are more susceptible to noise. If k is large, then the classification results will be more robust against noisy data at the cost of class boundaries becoming less distinct. In the tool developed in this project, k is defined by a user, allowing for different values of k.

To determine the neighbors in the k-nearest neighbor algorithm a distance metric is used to calculate the distance between two objects. For this thesis, I chose a total of 4 distance metrics. The metrics I experimented with are Euclidean, Manhattan, Chebychev, and Pearson [7]. Each of these metrics determines the distance between two objects in different ways, therefore giving slightly different results depending on which one is used.
2.4.1 Euclidean Distance

Euclidean distance is the probably the first type of metric one considers when considering distance. Some consider this distance an ordinary distance because this metric measures the distance between two points in a straight line. Even though this is a basic way of measuring distance it can still help provide information on the location of an object relative to the other. The equation (2.1) shows how Euclidean distance between two points, $p$ and $q$, is calculated:

$$
\text{distance}(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}
$$

(2.1)

where $p = (p_1, \ldots, p_n)$ and $q = (q_1, \ldots, q_n)$ are data points and $n$ is the total number of dimensions.

2.4.2 Manhattan Distance

Manhattan distance, also known as the taxicab metric measures the absolute distance between two points. To calculate the Manhattan distance one just takes the sum of the absolute difference of the dimensions between two points. This metric is similar to the Euclidean distance metric giving us a semi-straight path from one point to another, however, this metric give us larger values for distance which can help when comparing two points that are very close together. The equation (2.2) can be used to calculate Manhattan distance:

$$
\text{distance}(p, q) = \sum_{i=1}^{n} |p_i - q_i|
$$

(2.2)
where \( p = (p_1, p_2, \ldots, p_n) \) and \( q = (q_1, q_2, \ldots, q_n) \) are data points, \( n \) is the total number of dimensions.

2.4.3 Chebychev Distance

Chebychev distance is a little bit different than other distance metrics such as Euclidean and Manhattan. When calculating this distance the dimension with the greatest difference is used as the value of the equation. This value is good for when you want to focus on the maximum difference between two objects. The equation to calculate Chebychev distance can be seen in equation (2.3):

\[
\text{distance}(p, q) = \max_{1 \leq i \leq n} (p_i - q_i)
\]

(2.3)

where \( p = (p_1, p_2, \ldots, p_n) \) and \( q = (q_1, q_2, \ldots, q_n) \) are data points and \( n \) is the total number of dimension.

2.4.4 Pearson Distance

Pearson distance is used to measure how closely related two samples are based on the correlation of their respective profiles. By doing this, it removes the actual values of each dimension and focuses on the behavior of the dimensions across the entire point.

This can lead to a different type of clustering based on behavior instead of pure value.

The equation to calculate the Pearson distance can be seen in equation (2.4):

\[
\text{distance}(p, q) = \frac{\sum_{i=1}^{n} (p_i - \bar{p})(q_i - \bar{q})}{\sqrt{\sum_{i=1}^{n} (p_i - \bar{p})^2} \sqrt{\sum_{i=1}^{n} (q_i - \bar{q})^2}}
\]

(2.4)

where \( p = (p_1, \ldots, p_n) \) and \( q = (q_1, \ldots, q_n) \) are data points and \( n \) is the total number of dimensions. \( \bar{p} \) and \( \bar{q} \) are the average value of the coordinates of the two data points \( p \) and \( q \).
2.5 Singular Value Decomposition

The goal using singular value decomposition (SVD) is to take the multidimensional data acquired from the microarray compiled by Lu et al. and compress it down into a three dimensional dataset that can then be visualized. Ideally, this compression is done while preserving the relative distances between points, unfortunately there will always be a slight loss of information when compressing the data. The end result of applying SVD to the dataset is to map the data points to a three dimensional space that provide the most information about all of the tissue samples. To get this I just took the factors generated from the SVD algorithm and selected the rows with the highest eigen values and recomputed the matrix. The new matrix is the closest approximation to what the original data visually looks like allowing the potential to see patterns in the data that might have otherwise gone unnoticed.

The desire to visualize the microarray data for this thesis came about from an earlier attempt to classify different cancer samples based on the k-nearest neighbor algorithm. The results of this algorithm on the data showed that the sample points were well clustered. The decision was made to attempt to render the microarray data in 3D in attempt to find a better method of classification.

2.6 Tree Rendering Methods

One of the best ways to visually represent the GO data obtained from the miRNA targets is in a tree structure. Since the classifier code was written in Java one of the simpler solutions is using Java’s built-in JTree structure to do most of the work. To use a JTree there are just a few simple steps that had to be accomplished. First, a tree
containing all of the GO IDs was generated. This established the hierarchy and organization that we will build from later to generate the final tree. Next a list of GO IDs that can be mapped to the tree are needed. From there the program can iterate through each GO ID and find all of the different branches that contain this ID. Since GO annotations can have multiple parents, this implementation duplicates nodes across multiple branches to simulate this. Finally, after all of the GO IDs have been mapped to the tree, the leaves and branches not containing any mapped IDs are trimmed from the tree making it easier to view.

An alternative to using Java built-in tree structures is by mapping the GO ID to a directed acyclic graph (DAG) structure. While having all of the same benefits as using a standard tree a DAG would also allow us to link each GO annotation to all of its parents directly, without duplication. By using a custom DAG viewer that I developed, the annotations can be mapped to each node in a similar manner as the tree, with few exceptions. When the miRNA targets are received from the database they have with them all of the GO annotations leading all the way back up the base annotation, essentially the tree paths are predetermined by the database. By using this knowledge, the final structure of the DAG can be delayed until the end, improving the overall mapping speed. Instead of mapping the targets to the DAG directly, they are first mapped to the individual nodes maintained in a list. Once all of the targets have been successfully mapped, the nodes can be arranged into the DAG based on the gene ontology hierarchy. Since the mapping of the nodes comes before the structure of the tree is known, we can bypass the step of trimming the DAG saving time when dealing with a large number of GO annotations.
There are two main tools that were developed in this thesis to analyze the microarray data acquired from the dataset produced by Lu et al. Each of these tools were designed to look at slightly different aspects of the microarray data to possibly find patterns that could help in classifying cancerous tissues. The first tool examines the structural aspects of the cancerous data. In this case, structural data means the actual expression values contained in the microarray. When looking at the expression data, the focus is on finding patterns in the samples taken in the hopes of being able to predict the classes of future samples based on a classification approach. The other method that was examined is a functional approach to the classification of tissue samples. In this case the differentially expressed miRNAs are examined to obtain their mRNA. These mRNA would be responsible for controlling one or more cellular functions, defined by GO IDs, which could potentially lead to patterns in one or more of the cancer samples. To find these patterns the GO IDs were mapped to a tree structure that would help visualize the data and expose any interesting structures.

3.1 Preprocessing

When analyzing the data from the microarray it is important to organize the information in such a way as to filter out extraneous data while retaining as much
information as possible. With the data acquired from Lu et al.’s work, I already have a semi-filtered set of values to work with. I then took that data and broke it into different groups, trying to pair normal samples to cancerous samples. From there, I applied a t-test to the data to select the best genes. After selecting the genes that provided the data with the lowest t-test score, I then feed this information into the classifier to see if it could differentiate the samples of different classes. The data can also be fed into the 3D viewer to get a better idea of how the samples relate to each other visually.

After formatting, the main input to the classifier is in the form of a tab delimited matrix containing n samples with m attributes. The expected format of the file is as follows. The first line of the file is the number of genes that will be used for the visualization. The next line consists of a header line that describes each column. The first field is reserved for the description of the genes whose expression data was measured in each sample taken. The rest of the files consists of the samples that follow the pattern described above.

Each sample is labeled by the class description of the sample followed by an underscore, followed by an ID number. The class description and the ID number separately do not have to be unique to the dataset, but together they create a unique composite key that identifies a sample in the dataset. The reason for choosing this format for the key is that the program will automatically cluster the samples in the training datasets based on the keys of the samples to assist in the classification of the test data. The first row of table 2.1 shows how the data is formatted in the input file.
3.2 kNN Classifier

One way to classify a sample is to look at the neighbors surrounding it and through a jury vote system, classify the unknown sample by the most prevalent of the known samples surrounding it. This nearest neighbors (NN) method is simple to implement and can give us important information about the distribution of the samples in the microarray which can be viewed using the 3D viewer. To implement the kNN classifier, the basic kNN algorithm described in section 2.4 was followed and applied to the microarray data. To do this, each object to be classified is a sample taken from either normal or cancerous tissue. The expression values of the genes in a sample determine the position of the sample in the multidimensional space. Then the data from the unknown sample is compared to the data from the known samples to get a distance from the unknown sample to the known ones. This distance is calculated from one of the four distance metrics described in section 2.4. The nearest ‘k’ known samples are picked as the data points that will determine the unknown sample’s class, with majority rules jury voting. This information is then displayed to the user with a confidence rating determined by the ratio of winning votes to total votes.

3.3 SVD

A programmatic solution for singular value decomposition (SVD) is a problem that has been solved many times in a variety of different programming languages. Since numerous solutions exist, there isn’t a lot to gain from recoding the solution again so I decided to use a preexisting solution for my work. I found an open source solution called JMathTools [9]. JMathTools is a suite of tools that provides useful methods for
manipulating matrix data. Bundled with these tools was a simple way to perform singular value decomposition. Taking advantage of this, I leveraged the implementation of SVN created by the developers of JMathTools and wrapped it in a interface that I could use.

3.4 3D Viewer

The same data that the kNN classifier used can be compressed into three dimensions with SVD and then plotted onto a 3D graph. The reasoning behind plotting the data to a 3D graph is to present the microarray data in a different format to potentially reveal patterns that might have otherwise gone unnoticed. One point to note is that when the training data is loaded into the program it is labeled with the name of the sample that it was derived from. This labeling is used by the 3d viewer to color the graph such that the array data collected from a particular sample is colored the same. This coloring helps the user differentiate the between the different samples when viewing the graph and aids in the discovery of patterns.

After the data has been compressed the program uses a series of calls to the Java3D API to render the data as points on the screen. Once the points are displayed, there are several different actions that can be preformed to the plot by the user to get more information from the data. One of those actions is the ability to rotate and zoom the viewing field around the plot. By rotating the plot, the user will be able to get a better grasp on any patterns that might appear by examining different angles. Also the user has the ability to select a point on the plot. When a point is selected, its name is displayed on the screen so it can be reference back in the original data. Selecting has a second purpose
when both a training set and a testing set is displayed. When this occurs, and the option is turned on, the user has the ability to select a testing data point and view the points in the training dataset that are the closest to it. Distance to a point is based on one of several distance metrics that the user can select from in the analysis section of the program. This visualization is good for viewing the data from the microarray to help find miRNA regulation patterns, in later sections I will discuss how to view the ontology of the miRNAs to help find patterns in the function of the miRNAs.

3.5 Database

The database that was acquired from the Sanger Institute contains all of the data that is needed to lookup the targets of the miRNAs locally. The database is sectioned into multiple tables containing different types of information. For this project, the data I’m concerned with is contained within only a few tables. The data includes the targets of the differentially expressed miRNAs, gene ontology descriptions of the targets and their corresponding GO IDs. The total amount of information needed to complete this project was small; however, since there were around 11 million records needed to be filtered, the process took some time.

To gather the required data, several stored procedures were implemented in MySQL so they could be updated and debugged without having to change any of the original source code. There are three main steps to get the targets of a miRNA from the database. First the input from the main program is received through the network and it’s put into a table. By putting the input into a table of its own, the rest of the querying process is greatly simplified. Next the input table is then used to search the target
database for any mRNA that is targeted by the miRNA. Finally, with all of the targets found, the database is queried for the specific GO information on each mRNA so it can be mapped to the tree.

3.6 GO Tree Mapping

The GO annotations are organized into a tree like structure and it makes sense to be able to view them as such. If there are only one or two annotations associated with the targets of a miRNA, this structure is relatively simplistic, perhaps only a single path along the tree is involved, the resulting tree may not be very interesting. However, since the numerous targets of the miRNA have, multiple GO annotations, we end up with a fuller, more interesting tree. In the following sections I will discuss two different approaches at mapping the GO annotations, using a standard tree structure, and using a directed acyclic graph to map the GO annotations.

3.6.1 Java’s JTree

The work from the Gene Ontology consortium created the GO IDs to take on a tree structure, with more generic IDs closer to the root. The GO ID structure also allows for a particular GO node to have multiple parents making it difficult to map to a tree structure which only allows for one parent. In this section I will briefly go over the method that I attempted to translate the GO ID tree structure into Java’s JTree structure.

The GO terms defined by the Gene Ontology consortium are arranged in a hierarchical tree structure. To take advantage of this structure I took miRNA targets and mapped gene description ontology of the targets to the hierarchical tree (GO tree). The goal was to analyze the targets of the miRNA and see if there is a conflict between the
miRNAs that were up regulated and the miRNAs that were down regulated. If there was a conflict in targets then we would want to investigate the targets in question further to help medical scientist in discovering the cause of some types of cancer and disease.

The process of mapping the targets of a miRNA to a GO tree is a straightforward process. However, with the large amounts of data that has to be filtered though to find those targets, the process can take some time computationally. With the help of the miRNA target database created by the Sanger Institute, an application can easily be created to map the targets to a GO tree and analyze the data. The following sections present the general layout of the GO Tree mapping program, and the database that it utilizes, and some details about how the mapping program works and what it can tell us about the GO IDs.

The bulk of the GO tree mapping program can be separated into three major groups, the visualization, the tree storage and manipulation, and the database connectivity. First, I’ll go over the components in the visualization and the database connectivity because they are straightforward in their purpose and functionality, then I’ll discuss the tree storage and manipulation, which is the bulk of the program.

The visualization, created using the libraries that are a part of the Java language, supplies some needed functionality including the tree display, and an interface for the user to select which targets that they would like to view. Through the GUI, the user can save and load trees as well as load a custom miRNA list to use as input for the system. Also provided is the ability to view information about a selected GO ID including, the targets that are annotated with the GO ID and the information about the GO ID itself. For machine to machine interfaces a database connectivity module is used. This module
handles all database transactions and then returns the results in a usable format to the main program. The module utilizes a library provided by the creators of mySQL to interface with their database.

The largest portion of the program is the tree handling section. This module transforms the data gathered in the database into a tree structure that is then returned to the user interface. Also, within this module, there are methods that can be applied to the tree to gather more information about it. The tree itself is contained within this module to promote easy manipulation when the tree is being constructed. Each node of the tree contains all of the information that is known about it, except the description of the GO ID. In the case of the GO ID description, a database lookup is performed to get information about the ID. The reasoning behind the lookup is that since this field of study is still relatively new, the descriptions could change between sessions and if a tree is saved for a long time then the most up to date information about the GO IDs can be accessed.

3.6.2 Tree Customized DAG Viewer

One of the disadvantages to using a tree to represent a GO tree is that GO trees aren’t really trees. GO trees are actually directed acyclic graphs (DAGs). One of the defining characteristics of a tree is that each node only has one parent, which is not the same in a DAG. DAGs allow nodes to have multiple edges leading into them, in this case the parents. This caused problems when using the JTree approach because it was difficult to represent the multiple parents when restricted to only one parent per node. Previously, to alleviate this issue, we took a single node and cloned it to both parts of the
tree. This allowed us to force the data to fit the tree structure. However, by forcing the data to fit the tree we artificially change its structure in a way that tends to hide information. When dealing with GO trees generated with small sample spaces this isn’t much of an issue because we can easily view all of the nodes and see the duplicated ones. However, when we get into larger sample spaces, ones that generate thousands of nodes, we need to take care and not duplicate any information. This is where representing a DAG becomes important. If done properly a DAG can be represented in a tree-like structure (Figure 3.1), while still allowing for DAG characteristics, the important one being the multiple edges leading into a node. Then, when a node is displayed to a user, all parentage is known about the node just by looking at the lines leading into it.

![Figure 3.1 An example of generated DAG. Each highlighted node (denoted by a white dot) has hidden children that can be accessed while the other nodes have displayed all available children.](image-url)
Java, unfortunately, doesn’t have built-in libraries to handle DAGs so, in order to implement the DAG in Java most of the components had to be built from scratch. Since I was only dealing with one source node for the DAG, then the behavior is similar to that of a standard tree structure and similar methods can be applied to build and traverse it. The main issue is if there are multiple edges coming into a node then there is the danger of accessing a node twice and skewing the data that we collect. Of course, we also had this issue before while using a standard tree structure because the same node could be in two places. When using a DAG the solution is simpler because we only need to worry about one node instead of nodes that are cloned throughout the tree.

To display a DAG form of the GO tree, we took advantage of the fact that the general shape that a DAG would form is a tree as long as we define a root to be used as a starting point. This can easily be acquired from finding the source of the GO tree and using that as the root. The definition of a DAG allows for multiple roots in the graph. This is even true for the GO tree given all of the possible namespaces that nodes exist in. However for the work of this thesis we focus on one namespace, molecular function.

Now that we have a root, the rest of the tree can be generated easily. The actual root of the tree isn’t needed during construction since all of the node information is contained in the gene ontology file, however knowing the root’s location allows for a good starting point when rendering and traversing the tree.

To actually construct the DAG from the gene ontology file, the data contained in the file is read into the system. Since the data in the gene ontology file is already separated into different nodes, all that needed to be done computationally was to translate those nodes into a form that could be used by the system and remove the nodes that were
not needed because they were outside the namespace that we were focusing on. If a node was in the namespace that we wanted then we could add it to the list and all of its parents if they haven’t already been included. Then this node is added as a child to the parents. This process then continues until the entire file has been processed. By the end of the process a DAG should be made with every possible node that could go in that namespace.

A single namespace in a gene ontology graph can contain thousands of nodes, and in the case of the molecular function namespace, there are approximately 9000 nodes that have to be handled. Thankfully we don’t need all of the nodes in the namespace because we are only concerning ourselves with nodes that are associated with the targets of the miRNA under study. By removing irrelevant nodes, we can shrink the tree down to a more manageable size. This speeds up the traversal of the tree for any other calculations that need to be made, however when displaying the tree to the user, other optimizations need to be made to the tree so that it can be rendered quickly.

The actual rendering of the nodes is just a simple matter of overriding Java’s built-in painting methods to get the desired results. The general structure of the DAG as it is viewed by the user should be discussed because how the information is displayed can have an impact on how the data is interpreted. Having all of the nodes displayed at once is usually too overwhelming for both the user and the computer. However we still need access to all of the nodes so the tree can be examined in more detail, so a slight compromise can be made. When the tree is initially rendered, only a few nodes are displayed on the screen, the root, and possibly one or two levels below the root. By displaying the tree in this way, the user isn’t overwhelmed by the over abundance of nodes, and the computer doesn’t have to take the time to display them. To display the
rest of the nodes the user has the ability to click on any node that has a white dot on them. This white dot signifies that that particular node has children that can be displayed to the user, providing more information about the trees structure. As long as the user doesn’t click on every node in the tree, then the nodes displayed shouldn’t get too out of hand.

Another important feature of the information provided is that the entire parentage of a node is known, that is to say, a line is drawn from a node to all of its parents so that at a glance a user can determine the ancestry of any given node. This has both advantages and disadvantages for the user. The biggest advantage is that the exact parentage is displayed with the least amount of nodes. As discussed before, another way this could be displayed is by duplicating nodes with multiple parents cluttering up the display with unnecessary nodes, not to mention the children of those duplicated nodes. By rendering all of the edges that lead to the parent nodes instead of duplicating children with multiple parents then the interface can become much cleaner and simpler to understand for the user. This method does have its disadvantages however stemming from the extra lines generated by the multiple parents. When too many edges are rendered on the screen at the same time the probability that two or more edges end up crossing paths is likely. In fact, with multiple parents, it’s pretty much guaranteed. Some care can be taken in the layout of the nodes to minimize edge crossing, but it’s possible for a node to have parents that are on two different levels of the tree causing edges to intersect with each other and possibly edges intersecting with nodes.
4.1 Preprocessing Results

Running the t-test on the data collected from Lu et al.’s work I filtered the genes in several different ways. First, I ran the t-test on the dataset as a whole to get the samples with the most information. Next, I decided to first break down the data according to already known classification of the sample paring the normal samples and cancerous samples of the same tissue. The t-test was then run on these groupings to and the genes with the lowest t-test score were selected. Taking the top genes less than a p-value of 0.1 I then constructed a composite selection of genes, composed of the best genes taken from breakdown. This list of seventeen miRNAs was then pulled from the list of geness taken from the poorly differentiated tumors that were gathered by Lu et al.

The dataset that I acquired from Lu et al. was somewhat limited on setting up a training and testing environment to do the experiments. To make the most of what I had available I created files for two main purposes, one set of files was to test the general robustness of the nearest neighbor algorithm when applied to samples that had a reasonable amount of differentiation among them. The other set of data’s purpose was to test the nearest neighbor algorithm against poorly differentiated tumors (PDTs), using the differentiated samples as training data. For the robustness test I took the top genes as described in section 3.1 and applied nearest neighbor test to this file. Since the data was
limited, I used a leave-one-out algorithm where I set up my tests to remove one of the samples and use the rest as the training data. This removed sample was then used to test the engine to see how it would be classified. For the PDT samples, this was a much simpler process since I could just use the PDT file with the same selected miRNAs as the training data and apply the classifier using the two sets of data. The results of these tests are described in the next section.

4.2 Classification Results and Analysis

The first test that I ran against the data was an iterative process that used the most amount of data to classify samples. Since the data that I was working with was a little limited in terms of sample size, the leave-one-out algorithm was used. Starting from the original sample set, one sample was removed and then classified using the remaining samples as the training data. This was repeated for each sample in the series and for each type of distance metric that can be used, then the results were recorded for analysis.

The results of the test were grouped into individual section based on the distance metric used, the type of nearest neighbor used, and how many neighbors used. For each of these groups the total percentage of correct classifications was recorded for analysis. This strategy was employed for both the standard kNN classification as well as my attempt at improving kNN by using the average of the training data (the cluster based approach) to classify an unknown sample.

The standard kNN strategy, seen in figure 4.1 and described in detail in section 2.4, differs quite a bit from the cluster based approach. While the standard kNN approach finds the closes neighbors to the unknown sample, the cluster based approach
takes a different tactic in that it first finds the center point of the different cluster, seen in figure 4.2. After the center point of the cluster is found, the distance is then measured from the point to the unknown sample and the classification is based on the closest center point. By first finding the center point of the different clusters, all of the data in the cluster can be taken into account. Additionally, the cluster based approach also has the benefit of filtering out any data points that could be considered outliers as well as allowing clusters with a lower sample count to have more influence when classifying a sample.

Unfortunately, the results from the standard KNN algorithm didn’t perform as well as expected. When running against the sample data acquired from Lu et al. the classifier only predicted the correct tumor to an accuracy of 46% across the different

![Figure 4.1 Using the standard kNN classification method. Each arrow represents the closest neighbors to the test point.](image)
distance metrics tested with the lowest accuracy being 41% when calculated with Manhattan distance metric (K=3) and the highest being 59% when calculated with the Pearson distance metric (K=3), seen in table 4.1. As a different test I decided to find the center of the training data and use that data point to decide how to classify a sample from the testing data. With the samples classified there was a noticeable increase in the accuracy of the classifier bringing the average percentage of accuracy up to 56% with the lowest accuracy being just over 50% when calculated with the Chebychev distance metric and the highest accuracy being 61% when calculated with the Pearson distance metric, seen in Table 4.2. A breakdown of each distance metric, the algorithm used, and their classification results can be seen in table 4.3.
The increase in accuracy between the standard kNN algorithm and classification using the center of the cluster isn’t a significant gain, but it does show that if all of the data is taken into account for a training set then there is a better chance that we will be able to properly classify an unknown sample. Of course these results are still rather low to be used for any meaningful purpose in real applications.

The final test of the classifier was using the PDT samples as the testing data and recording the accuracy. This, as expected, turned out worse than just running the test against the samples with better differentiation. Overall, the test only showed around a 15% accuracy when identifying the PDTs. However, using the Pearson distance metric the classifier managed to have 26% accuracy for the normal nearest neighbor cluster. Furthermore, when the Pearson distance metric was used with the clustering approach, the following results were obtained:

<table>
<thead>
<tr>
<th>Sample Name Predicted</th>
<th>Actual</th>
<th>Confidence</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_COLON_1 N_COLON</td>
<td>N_COLON</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N_COLON_3 N_COLON</td>
<td>N_COLON</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N_COLON_4 N_COLON</td>
<td>N_COLON</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N_COLON_5 N_COLON</td>
<td>N_COLON</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_2 T_COLON</td>
<td>T_COLON</td>
<td>0.66666667</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_3 T_PAN</td>
<td>T_COLON</td>
<td>0.66666667</td>
<td>0</td>
</tr>
<tr>
<td>T_COLON_4 T_COLON</td>
<td>T_COLON</td>
<td>0.66666667</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_5 T_COLON</td>
<td>T_COLON</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_7 T_COLON</td>
<td>T_COLON</td>
<td>0.66666667</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_9 T_COLON</td>
<td>T_COLON</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_BRST_2 T_MELA</td>
<td>T_BRST</td>
<td>0.333333333</td>
<td>0</td>
</tr>
<tr>
<td>T_BRST_3 T_BRST</td>
<td>T_BRST</td>
<td>0.66666667</td>
<td>1</td>
</tr>
<tr>
<td>T_BRST_4 T_BRST</td>
<td>T_BRST</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T_BRST_5 T_BRST</td>
<td>T_BRST</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T_BRST_6 T_BRST</td>
<td>T_BRST</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Percent correct 59.50%**

Table 4.1 Results from kNN using the Pearson distance metric with k=3.
the accuracy increased again, bringing it up to 31.5%. This increase in accuracy supports the validity of the clustering approach, even though the accuracy is still quite low. It is also worth to point out that the increase in percentage can only be shown for this dataset.

4.3 SVD Results

One of the more interesting features that can be noted when graphing all of the samples data points is that there is a defined split in the graph. The split seems to partition several of the samples; however, it doesn’t form a clear divide between

<table>
<thead>
<tr>
<th>Sample Name</th>
<th>Predicted</th>
<th>Actual</th>
<th>Confidence (NA)</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_COLON_1</td>
<td>N_COLON</td>
<td>N_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>N_COLON_3</td>
<td>N_COLON</td>
<td>N_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>N_COLON_4</td>
<td>N_COLON</td>
<td>N_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>N_COLON_5</td>
<td>N_COLON</td>
<td>N_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_2</td>
<td>T_COLON</td>
<td>T_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_3</td>
<td>T_PAN</td>
<td>T_COLON</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>T_COLON_4</td>
<td>T_COLON</td>
<td>T_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_5</td>
<td>T_COLON</td>
<td>T_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_7</td>
<td>T_COLON</td>
<td>T_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_COLON_9</td>
<td>T_COLON</td>
<td>T_COLON</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>T_BRST_2</td>
<td>T_BLDR</td>
<td>T_BRST</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>T_BRST_3</td>
<td>T_BRST</td>
<td>T_BRST</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_BRST_4</td>
<td>T_BRST</td>
<td>T_BRST</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_BRST_5</td>
<td>T_BRST</td>
<td>T_BRST</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>T_BRST_6</td>
<td>T_BRST</td>
<td>T_BRST</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

Percent Correct: 61.70%

Table 4.2 Results with the Pearson distance metric using the clustering approach.

<table>
<thead>
<tr>
<th></th>
<th>Euclidian</th>
<th>Manhattan</th>
<th>Pearson</th>
<th>Chebychev</th>
</tr>
</thead>
<tbody>
<tr>
<td>kNN</td>
<td>43.82%</td>
<td>41.57%</td>
<td>59.55%</td>
<td>49.44%</td>
</tr>
<tr>
<td>Cluster</td>
<td>53.93%</td>
<td>58.43%</td>
<td>61.80%</td>
<td>50.56%</td>
</tr>
</tbody>
</table>

Table 4.3 Classification accuracy for all of the distance metrics tested.
cancerous samples and normal samples, visible in figure 4.3. Also, there is a natural clustering of the different sample types, for instance the cancerous samples of a particular tissue form one cluster while normal tissue samples from another cluster. However, when the poorly differentiated tumors are plotted they all end up in a similar location on the graph; with the exception of the poorly differentiated colon tumor which actually ends up in the same vicinity as the normal tumor samples.

Another feature of this graph is to identify the programmatically determined nearest neighbors of a node just by clicking on it. Utilizing this feature, I inspected some of the data points to see what the nearest neighbor algorithm identified as the most similar nodes. Doing this doesn’t provide any information that the standard classification could provide, but it did however; confirm suspicions that due to the density of the data, identifying with the standard nearest neighbor algorithm would not yield very accurate results. This became even more evident when the poorly differentiated tumors samples were plotted since they all end up near similar training samples.

To get a different view of the data, the user can select which dimensions to view the data on. These dimensions correspond to the eigen vectors and eigen values calculated from the SVD. The magnitude of the eigen values show the user where the bulk of the information is in the data. In most cases, a user would want to select the three dimensions with the highest values. In my case, this resulted in the plot with a well defined partition in the middle of the cancerous samples and normal samples. However, selecting dimensions with a lower value is also possible, but when doing this in most cases the plot generally becomes more chaotic and there is very little insight to be gained from it.
Figure 4.3 An example of the 3D visualizer. Each colored point represents a different tissue sample.
The final feature that can be used on this plot allows the user to highlight the samples of a specific group to get an idea of where they are located in the graph. The reason for this is that when there are a large number of samples being plotted at the same time, the graph can become overwhelming with all of the different colors and points. To resolve this issue, when a user picks one classes of samples to view, all other samples are grayed out making the samples that the user is interested in stand out. This of course, can be applied to one or more groups of samples for easier comparisons.

The 3D plot greatly assists in visualizing what the data looks like and why classifying poorly differentiated tumors is so difficult. Sometimes, new solutions can be solved by first viewing data in a different light. In this case, a new solution wasn’t found, however, with different datasets a pattern may start to form. The next section will discuss what can be discovered when analyzing the different molecular function of the genes using the GO tree.

4.4 GO Tree Analysis

After creating a GO tree viewer with both the Java API JTree implementation and a customized DAG viewer created using the Java graphics libraries I attempted to compare the two methods and determine which would be the best method for displaying to GO trees. Each method has both advantages and disadvantages and each method gives different information when rendering the same tree.

4.4.1 JTree

The first attempts at displaying a GO Tree, seen in figure 4.4, had both benefits and drawbacks. To do this I used Java’s JTree structure to visualize the tree. Because
the JTree is a fairly robust structure available in Java it was a straight forward task to map the GO Tree nodes to the JTree nodes. This allowed me more time to focus on enhancements to the program and let the JTree handle the rendering details. The final structure of the tree consisted of the hierarchy of the nodes represented in a tree form. The only difficulties of using this method is that there isn’t a one-to-one translation of the GO tree structure to the standard JTree structure because of the multi-parentage of some of the GO tree nodes. To get around this difficulty nodes with multiple parents were duplicated and placed accordingly under the proper parents. This is one of the simpler solutions to the problems and I felt that this would be the best solution for this approach.

The translation of the GO tree to the JTree seems to cause a loss of information of the overall structure to the GO tree. If the tree is large enough, there can be a lot of nodes that a user would have to examine at one time. If I duplicate the nodes to solve the multiple parent problem then it will end up giving the user even more nodes to look at. This could potentially be overwhelming to the user and cause them to lose a grasp on what the entire structure should look like. This method will also cause problems when trying to generate statistics on the tree itself. Since some of the nodes were doubled, I could not use normal tree traversal to generate the statistics about the tree because we would end up counting the features of the same node multiple times skewing the results of the tree, although the problem can be avoided with careful counting of the nodes to make sure that we don’t over count any.

There are a couple of statistics generated by this tree that aid in analyzing the GO annotations of the miRNA targets. The first is the total common targets that are on a single annotation. When only examining the GO annotations from the targets of a single
miRNA this value will always be zero, however this value will vary when there more than one miRNA are select. For some of my tests, I selected three miRNA, hsa-mir-181a, hsa-mir-20a and hsa-mir-15a and mapped them to the tree to view their conflicts. The goal of viewing the conflicts is to see the more active areas in the GO annotation tree that are influenced by the miRNA targets. In this case we consider a target a common target when two or more miRNA have the same target. Then when this target is mapped to the GO Tree we can view how the biological process may be affected. For example we can take a look at the annotation GO:0008428 (enzyme activator activity), which has a total of three common targets targeted by the three miRNAs. We can imagine that if a common target is targeted by more than one up regulated miRNAs, then the expression of this target has higher chance to be altered downward.

Figure 4.4 A GO Tree generated using the tree structure provided by core Java libraries.
Related to the total common targets count, there is another count that could potentially prove useful. Along with the common targets, the unique (non-common) miRNA targets are also recorded as a reference point to figure out if a particular annotation is dominated by common targets. To figure out if an annotation is dominated by common targets the ratio is taken of conflicting targets to non-common targets. This results in annotations with higher ratios if they are dominated by common targets. To help visualize this better, the nodes on the JTree are shaded corresponding to how high the ratio is.

4.4.2 DAG

The main driving force in representing the GO tree as a rooted directed acyclic graph is because a single node can have more than one parent node. Since this is hard to represent in a standard tree form, and Java doesn’t have native capabilities to display DAGs, it wrote my own library to display the nodes. In general displaying the nodes as a DAG seems to be an improvement over the standard tree structure for a couple of reasons. First, as stated earlier, a particular node’s parentage is clearly displayed by the use edges drawn between the node and its parents, seen in figure 4.5. Secondly the node does not need to be duplicated to properly represent the parentage allowing us to limit the information that is being displayed to the user.

One disadvantage of this implementation is that processing time for rendering the graph takes quite a while. While using the a standard tree in Java, the nodes had to be processed to find their correct location in the tree, but once that was solved, the rendering the process was optimized to the point where it didn’t noticeably affect performance.
When rendering the tree as a DAG, significant post processing had to take place to properly space the nodes on the screen. To get around this, most of the tree is hidden from the user when it is loaded. Then as the user clicks on nodes its children will be displayed. If however the user clicks too many nodes, the performance will decrease and the screen will inevitably become cluttered form all of the nodes being displayed.

Another disadvantage of this implementation is that edges in-between nodes can intersect with nodes that have no relation with it. This intersection usually occurs when a node has multiple parents. An attempt was made to sort the nodes to minimize the intersections, but they still occur occasionally. To get around this, I attempted to use Bezier curves to divert the lines around the intersecting nodes with some success.
4.4.3 Comparison

After spending time investigating these two methods of rendering the GO Tree, there are a few key points of comparison that can be looked at. First, rendering the tree could be time consuming. Sticking with a standard tree, the actual rendering can be handled within the engine created for Java. By using an optimized rendering framework, the GO tree can be displayed quickly no matter the size of the tree. On the other hand, with the implementation of a DAG that I created, the rendering of the tree to the screen is relatively slow.

Another point of comparison is the type of view that these two different graphing methods supply. The main difference between the two structures is the placement of nodes required to accurately display the information. If just a tree structure is used, then there is a requirement that nodes are copied to each node that they are a child of. Alternatively, if a DAG is used instead, then there is exactly one node for each GO annotation displayed at a time, keeping the graph concise for the user. The conciseness however, seems to degrade as the number of nodes in the graph increases due to the amount of information that is displayed to the user making it difficult to view the DAG in a meaningful manor. As similar situation happens with the tree structure created by using a JTree when the number of nodes becomes exceeding large, but since there doesn’t have to be a visual relation between a node and all of its parent nodes the tree is more manageable.
CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Developing classifier based on microarray data was an interesting experiment in attempting to classify cancerous tissue samples based on the expression levels of the miRNAs. My investigation of the classification problem shows that my cluster based approach worked better than the standard kNN algorithm. The new algorithm reduces the noise in the data. While finding the center point of the cluster didn’t increase the accuracy of classification that dramatically, it did provide at least some improvement over the standard nearest neighbor methods. It is also worth mentioning that the best improvement was seen by using the Pearson distance metric when finding the center point of the cluster.

When examining the GO tree mappings, and ignoring the tree preprocessing time for each method implemented, it is hard to ignore the general ease of use and rendering time of the standard JTree interface. That being said, I believe that the DAG approach, with the proper improvement, would end up being a better interface. With its ability to show relationships between nodes visually instead of copying nodes with multiple parents it can provide a friendlier user interface.

The most interesting tool that came out of my work was the 3D viewer that I made for displaying the miRNA data points. Being able to plot the data in 3D space
provides a unique perspective that can inspire different approaches to solving the classification problem. In particular, the well defined split in the graph proved to be interesting, showing that even though the samples were not clearly divided by cancerous and non cancerous, there is at least one trait that contained a lot of information.

5.2 Future Direction

The work that Lu et al. accomplished with their work with miRNAs has shown some promising results. My work with classifying samples further demonstrated cancer sample classification problem is a challenging problem. My results also indicate new approaches are needed to classify the poorly differentiated tumors. Quite possibly this search for new ways to classify can be found in the plotting of the information such that new patterns can be discovered visually and utilized in building classifiers.

Further development on the DAG GO Tree viewer would improve performance and increase the usability for the average user. One of the major components that can be improved is the look and feel of the nodes representing the annotation. Along with the nodes themselves, the edges connecting the nodes can also be improved such that the program would prevent an edge from intersecting a node. Another area to work on would be the rendering time of the graph. When there are a large number of nodes that need to be displayed at once, there is a noticeable slowdown when navigating the graph. General maintenance improvements should also happen, but those would naturally come out of the work detailed above.
REFERENCES


