Deven Damiano
Bioinformatics
Final Project
4/18/2019

Microarray Data Analysis

Introduction

As our final project, we chose the problem of Microarray Data Analysis, which focuses on classifying and predicting types of Leukemia, based off of given data that has been classified prior. Using machine learning we were hoping to train a classification model on provided data, to then test our model on its success rate in classifying a provided set of test data, in which we classify the type of Leukemia that a gene contains. The two types of Leukemia we studied for the project were AML and ALL. As a start, Leukemia is considered a cancer that would begin the development of different types of blood cells. Leukemia most commonly begins in white blood cells, but it can also begin development in numerous other cell types. There are several types of leukemia, and they are divided by being either acute (fast-growing), or chronic (slow-growing), and whether they begin in myeloid or lymphoid cells. Acute Myeloid Leukemia (AML) begins in bone marrow (myeloid cells), and usually moves into the blood. It begins in cells that would likely develop into white blood cells. Acute Lymphocytic Leukemia (ALL) begin in lymphocytes, which are immature white blood cells. Again, like AML, ALL begins in cells around bone marrow and quickly move into the blood. Looking at these cancer types from an overview shows how similar these two different types of are, yet they contain minute differences that can alter the treatment forms needed, and the location we need to target to fight the cancer. This is why analyzing the genes that make up these types of cancer is important. As we study and retrieve more data on these types of cancer, the better we are able to train models and more efficiently predict how to classify the cancer. As our prediction accuracy increases, so does the percentage of cure success rate.

Method and Materials

Our project utilized the Python programming language, as well as the pandas and Sci-kit frameworks to create our preprocessor, and our machine learning model. First, in preprocessing our gene data, taking the .res training and test data provided, which contains the gene expression data for AML and ALL Leukemias from over a thousand genes, I fed the data into a pandas data frame, which is represented as a table values, and I organized the data based off of the provided stipulations. I removed endogenous control genes, genes with A's across the experiments, and genes with less than two-fold change across the experiments. I also declared a threshold cut-off value of 20, in which I replaced all expression data values of less than 20 with the value of 20. With this fixed data, using excel, I calculated the p-value of all remaining genes, and with the 50 genes with the smallest p-values, I created a separate .csv file, and this file is considered the

program's training data. Following this same process for our testing data, I also obtained the top 50 genes for testing data.

Inside of our final portion of the project, begin our classifier, I read in the preprocessed training and test data. Using a KNN classifier from the Sci-Kit learn Python package, I trained a classification model with the training data, which trained to predict whether a given gene is AML or ALL Leukemia. Inside the program this is represented as either a 0 or a 1. Once the model had been trained, I fed the testing data to the model and it gave predicted output, which I print to the screen and also calculate the accuracy of the predictions.

Results and Discussions

Our results found that when our KNN classifier was trained based off of the top 50 genes (based off of p-value) and put against our test data, we had a 60% - 66% success rate in prediction. When testing on our original training data, we resulted in a 100% success rate. Below shows the given output of our program:

```
Training Predictions:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1
 1]
Test Accuracy: 1.0

Test Predictions: [0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 0
 1 0 0 1]
Test Accuracy: 0.6571428571428571
```

The lower success rate when testing against the test data was expected, as we trained on only 50 samples. Accurate models are usually subject to training on thousands of different samples, which would allow for more accurate results. Considering such a small sample of training data, a success rate of 66% is a fairly good. What I found most interesting is the effect of manipulating the amount of neighbors that the KNN Classifier trains off of. First, a KNN classifier makes its predictions based off of data points from the training data that are the closest to the test data points. When training the model to predict based off of 1 nearest neighbor, our prediction accuracy flatlined at around 66% percent. However; when I trained the model to predict based off of the 3 nearest neighbors, or the 5 nearest neighbors, the accuracy of our classification model decreased to 60% success. Output can be found below:

```
Training Predictions:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1
[1 1 1 1 1
 1]
Test Accuracy: 0.9736842105263158

Test Predictions: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
]
Test Accuracy: 0.6
```

Conclusion

Based on our findings, again, I think training on a higher number of genes, our prediction success rate could rise a bit more, but with the small number that we had, our success rate of 66% is fairly good. Looking at the expression data, we can see just how similar AML and ALL Leukemia is, and why it is difficult to classify the difference. I can surmise that the neighbors in the KNN Classifier are similar distances from each other, so that is why reducing our prediction modifying neighbors from 5 or 3 to 1 allows for an increase in prediction accuracy. I was very interested in testing a different type of classifier to see if I could increase the prediction accuracy, so I tested the data with a Decision Tree Classifier, and I actually did increase my prediction accuracy quite a bit, reaching 80% accuracy as seen below:

```
___ DECISION TREE CLASSIFICATION RESULTS ___

Test Predictions:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0
1 1 0 1 0]
Test Accuracy: 0.8
```

With experience using decision tree classifiers in the past, I have found a greater stability and accuracy than with any other classification model, as you are able to manipulate the amount of branches the tree will create during training. Though this type of model must be handled carefully, as you have the potential to actually overtrain the tree, which could actually result in a lower accuracy of the tree. I am attributing the higher accuracy score of the decision tree classifier to that of the KNN classifier to the way a decision tree classifier works. Each root of the tree is a decision based on a given training value, if the value is higher than the given, it will move to the right branch, if less than the given root value, it will move to the left branch. Thousands of these root comparisons can be done in a single classification, and once the end of the branch is reached, the predicted resulting output is given. When we utilize the accuracy and specificity of a decision tree classifier for data that is as similar as that of the AML and ALL experiment results for each gene, we can see how this higher specificity can result in a better prediction outcome, to rather than relying on a nearest neighbor approach. Again, while using the KNN classifier, the most accurate outcome I could gather is that of a 66% success rate. This was again with comparing 1 nearest neighbor, and the accuracy actually decreased to 60% as I predicted based on the 3 nearest neighbors or the 5 nearest neighbors. Attributing this to the fact that the data was so close, I think I'd rather use a decision tree classifier in this scenario. You will sacrifice the performance of the KNN classifier, as a decision tree will ultimately take longer to generate, but I would rather receive accurate output than rely solely on performance. I would most likely utilize the KNN classifier on a dataset like the classic iris dataset which classifies the type of flower species based on factors such as petal length. When looking at this data compared to that of our AML and ALL experiments, there is an obvious difference in the data between each flower species. I presume that a KNN classifier would perform quite well on this dataset, and this would definitely be a case where I would use a KNN classifier over a decision tree classifier.

References

Data files used in the project:
http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43

AML Background Info
https://www.cancer.org/cancer/acute-myeloid-leukemia/about/what-is-aml.html

ALL Background Info
https://www.cancer.org/cancer/acute-lymphocytic-leukemia/about/what-is-all.html

KNN Classifier Reference
https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Decision Tree Reference
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Appendices

Project Source Code:
https://github.com/damianoda96/Microarray_Data_Analysis

Group Responsibility

For the project, I was responsible for the following:
- Implementation of the data preprocessor
- Implementation of the KNN and decision tree classifiers
- AML and ALL Leukemia research