Classifying Cancer Using miRNA Profiles from Blood Samples

Madelyn Gatchel and **Drew Hager**
{magatchel,drhager}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

1 Introduction

Classification problems represent an interesting and relevant intersection between computer science and many other fields. In particular, machine learning techniques can provide insights for other fields more effectively and accurately than can humans. One important intersection is that between computer science and medicine, and the insights can have real human impact.

In this report, we present a method for classifying cancers based on miRNA profiles from blood samples; additionally, we recommend a model for physicians to adopt going forward. We implement three classification algorithms—decision trees, support vector machines, and *k*-nearest neighbors—and evaluate them on miRNA patient data from The Cancer Genome Atlas repository (NCI). To determine which model works best, we compare the F1 scores of the various models and choose the model with the highest overall score. We believe that our model will help physicians more efficiently tailor patients' immunotherapy treatments and we hope that in the future our model may be extended to classifying even more types of cancer.

2 Background

Classification is a branch of supervised machine learning in which we map a set of features to a discrete label or "classification." In this section we describe three different classification algorithms—decision trees, support vector machines, and *k*-nearest neighbors—as well as describe the dataset on which we will be training and testing these algorithms.

The Cancer Genome Atlas (TCGA) Repository

Various cancer studies have shown that while no two cancers are the same, they have patterns that express themselves in a specific type of RNA, called microRNA. Cancers affect the expression of microRNA even at early stages of cancer (Soto). MicroRNA (miRNA) are "small, highly conserved non-coding RNA molecules involved in the regulation of gene expression" (MacFarlane and Murphy 2010). In a study by MacFarlane and Murphy, they identify that "miRNA expression profiles are altered in specific tumors, implying that miRNA may be involved in development of cancer and other diseases" (MacFarlane and Murphy 2010). The blood sample data we will be using from TCGA contain miRNA profiles for six different types of cancer:

- Breast invasive carcinoma
- Kidney renal clear cell carcinoma
- Lung adenocarcinoma
- Lung squamous cell carcinoma
- Pancreatic adenocarcinoma
- Uveal melanoma

Decision Trees

Decision trees are a classification algorithm that use a series of decisions about a given data point to determine what its classification should be. The tree can be visualized as a directed graph that begins with a root and continues downwards towards leaves that represent classifications. For any given node in the tree, a decision must be made about a given data point. In a numerical data set, a given node might ask if a feature holds a value above or below a particular threshold. Based on the answer to this question, the algorithm will proceed in the direction of an edge that represents this value until it has reached a classification. While allowing the algorithm to make fairly precise decisions, this algorithm runs the risk of overfitting.

K-Nearest Neighbors

The *k*-Nearest Neighbors (*k*-NN) algorithm is an instance-based learning algorithm—instead of having a formal training stage where the "learning" takes place, this algorithm stores the training data and uses it in the validation or testing stage to make the classification prediction. More specifically, the *k*-NN algorithm classifies a given point by finding the *k* closest neighbors and assigning the mode of the associated classifications. This method can be resource intensive.

Support Vector Machines

Support vector machines (SVMs) are a type of classification algorithm that assumes data are linearly separable in some fashion. In general, the goal of SVMs is to find the best hyperplane that maximizes the margin, or the minimum distance from any data point to the hyperplane.

It is easiest to understand SVMs in a two-dimensional space in which data is organized in two blob-like clusters. In this example there will be a hyperplane that separates the data in the most ideal way such that the hyperplane is the maximum distance from both data clusters.

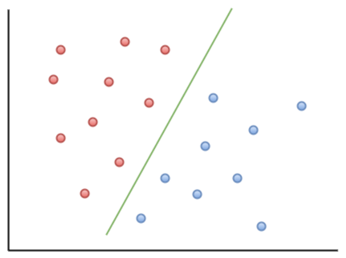

Figure 1: Two-dimensional SVM example.

However, in this project we examine a data set with over 1800 features, meaning that the space is not two-dimensional but rather *n*-dimensional. Based on the determined equation of this hyperplane, any new data will be compared to the hyperplane and classified in the category to which the hyperplane bounds it.

Related Work

Similar research was performed by Loher et al., who examined TCGA data to develop a model that classified 32 different cancers (Loher). The team used R to develop an SVM model that labeled data with an "average sensitivity of 90% and a false discovery rate (FDR) of 3%" (Loher).

3 Experiments

In this project, we will use `sci-kit learn` functions as they are more efficient than handwritten algorithms. Additionally, `sci-kit learn` functions use keyword arguments for hyperparameters, which makes them easy and straightforward to vary.

Preprocessing

Formatting the Data The data for this project is formatted as text files, each contained within patient folders, each of which was contained within a larger cancer classification folder. In addition, some patient folders contain annotation text files that provide different information than the regular data. Before combining the data, we iterate over all of the patients' files in order to ensure that the features are in the same order for each patient. We do not observe any obscurities in the data.

Next, we iterate over all the data files, appending each patients' reads per million miRNA profile to a data frame while also keeping track of which cancer type that patient is classified as. This results in a cleaner data set that tracks each patient's reads per million miRNA for each of the 1800+ features.

Splitting the Data We split the data into training, validation, and test sets with a ratio of 8:1:1. However, a simple random selection of data points would not be sufficient for this data set because the ratio of cancers is uneven. The table below summarizes the number of patients with each cancer type in this dataset. For example, if the randomized training data set included a small sample of a cancer with fewer data points (such as Uveal melanoma), the model may not perform well in classifying this cancer. To combat this, we performed a stratified split on the classifications such that there would be a balance of cancer types within each of the splits.

Cancer Type	Frequency
Breast invasive carcinoma	1096
Kidney renal clear cell carcinoma	544
Lung adenocarcinoma	519
Lung squamous cell carcinoma	478
Pancreatic adenocarcinoma	178
Uveal melanoma	80

Table 1: Cancer frequency breakdown.

Feature Selection As these algorithms can be computationally expensive, we use feature selection to reduce the number of features we compare, which reduces the amount of computation. We chose to use a variance threshold algorithm to determine which features to keep. This algorithm removes all features that have a variance below some threshold. Through trial and error, we determined that a threshold of 300 removed a sufficient number of features (1571 removed, leaving 310 features) while maintaining quality (based on F1 scores).

Data Normalization After performing feature selection, we normalize the columns of our data using the `sci-kit learn` normalization function. Since our targets are discrete classifications, we do not normalize our outputs.

Hyperparameter Tuning

For each algorithm, we iterate over all possible combinations of the hyperparameters with which we are familiar. The hyperparameters with which we are not familiar we do not vary because we trust that the `sci-kit learn` default options will be optimal. Note that if the algorithm uses a random process, we seed the process at 150 to ensure we get consistent results and can make fair comparisons among models. Below we explain the hyperparameters we vary for each algorithm.

Decision Tree Hyperparameters For the decision tree classifier we vary three hyperparameters: criterion for measuring split quality, splitting strategy, and max features. The two criterion for measuring split quality are Gini impurity and entropy. Entropy measures the disorder of the data and splits the data in order to increase the purity of the resulting partitions of data. The Gini impurity criterion splits the data to minimize the probability of misclassification. The splitting strategy is the strategy used to actually choose the split at each node; the options for this hyperparameter are "best" and "random." Finally, the max features hyperparameter is the number of features considered when looking for the best split; options for this hyperparameter are None (max features = num features), log2 (max features = log2 num features), and sqrt (max features = sqrt num features).

Support Vector Machine Hyperparameters With the support vector machine algorithm, we focus on two hyperparameters: loss and fit intercept. We use the hinge loss function, which is considered the standard SVM loss, as well as the squared loss function, which is simply the squared values from the hinge loss function. Additionally we vary the fit intercept hyperparameter, which specifies whether the hyperplane found has an intercept (fit intercept = True) or if it is anchored at the origin (fit intercept = False). Note that we increased the maximum number of iterations to 5000 in order for the model to converge.

***k*-NN Hyperparameters** With this algorithm, we focus on three hyperparameters: distance metric, weights, and the value of k. We experiment with two different distance metrics: L1 and L2, which represent Manhattan distances and Euclidean distances, respectively. The Manhattan distance is the distance between two points measured on axes along right angles, whereas the Euclidean distance is the "straight-line" distance between the two points. We also experiment with two weight options: uniform weighting (traditionally used) and weighting based on distance. According to the `sci-kit learn` documentation, the weighting by distance option "weight[s] points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away" (Pedregosa et al. 2011). Finally, we experiment with k ranging from 1 to 10.

Evaluation

For each model, we look at the F1 score and choose the model that has the best F1 score overall. Recall that the F1 score combines the precision and recall:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

We use the F1 score as opposed to accuracy, precision or recall because the F1 score quantifies the effects of different misclassifications (false positives and false negatives) using both precision and recall. Note that F1 scores range from 0 (worst) to 1 (best).

4 Results

Validation

Tables 2-4 show our validation results on three different classification algorithms using different hyperparameter values to generate a total of 32 different models.

Criterion	Splitter	Max Features	F1
gini	best	sqrt	0.87
gini	best	log2	0.79
gini	best	None	0.89
gini	random	sqrt	0.83
gini	random	log2	0.80
gini	**random**	**None**	**0.91**
entropy	best	sqrt	0.85
entropy	best	log2	0.79
entropy	best	None	0.93
entropy	random	sqrt	0.78
entropy	random	log2	0.76
entropy	random	None	0.89

Table 2: Decision tree model performance varying criterion, splitter and max features hyperparameters.

k-value	Distance Metric	Weights	F1
1	l1	uniform	0.89
4	**l1**	**uniform**	**0.94**
7	**l1**	**uniform**	**0.94**
10	l1	uniform	0.93
1	l2	uniform	0.86
4	l2	uniform	0.90
7	l2	uniform	0.91
10	l2	uniform	0.90
1	l1	distance	0.89
4	l1	distance	0.92
7	l1	distance	0.93
10	l1	distance	0.92
1	l2	distance	0.86
4	l2	distance	0.90
7	l2	distance	0.90
10	l2	distance	0.91

Table 3: k-NN model performance varying k, distance metric and weights hyperparameters.

Loss	Fit Intercept	F1
squared hinge	**True**	**0.95**
squared hinge	False	0.94
hinge	True	0.89
hinge	False	0.89

Table 4: SVM model performance varying loss and fit intercept hyperparameters.

We can see that the best decision tree algorithm has an F1 score of 0.91, the k-nearest neighbors model has an F1 score of 0.94, and the best support vector machine model has an F1 score of 0.95 (as seen in the bolded metrics in each table). Therefore, based on our results, we conclude that the best model is the support vector machine model with squared hinge loss and fit intercept set to True.

Testing

The results from running our best model on the testing set are summarized in the table below.

Cancer Type	F1
Breast Invasive Carcinoma	0.93
Kidney Renal Clear Cell Carcinoma	0.97
Lung Adenocarcinoma	0.81
Lung Squamous Cell Carcinoma	0.91
Pancreatic Adenocarcinoma	0.71
Uveal Melanoma	1.00
Overall	**0.90**

Table 5: Best SVM model performance on test set.

Because the F1 score from the testing set (0.90) is similar to the F1 score from the validation set (0.95), we say that our model generalizes well, which further supports that these settings produce the best model.

5 Conclusions

We believe that physicians can benefit greatly from our findings. We determined the model that performed best on our validation set and concluded that it generalized well on our test set. Our recommended model is a support vector machine using squared loss and fit intercept set to True. We believe that using our model for classifying cancer based on miRNA profiles from blood samples will allow physicians to identify cancers earlier, which will likely improve treatment success.

Should physicians wish to continue refining the cancer classification process, we recommend the following avenues for future research: extending the model to classify additional types of cancer; looking at other SVM models as well as neural networks; and extending the model to identify the cancer stage (severity) in addition to the cancer type.

6 Contributions

Both authors wrote the script that formatted the data as well as wrote the code for the three models. Both authors ran initial experiments and collected data. M.G. conducted further tests to determine the best model. M.G. wrote the Introduction, Results, and part of the Experiments section. D.H. wrote the Background and the other part of Experiments. Both authors wrote the Conclusion and proof-read the entire document.

References

MacFarlane, L.-A., and Murphy, P. 2010. MicroRNA: Biogenesis, Function and Role in Cancer. *US National Library of Medicine National Institutes of Health.*

The cancer genome atlas program.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Soto, J. The future of early cancer detection?