Leukemia Classification Using Support Vector Machines, K-Nearest Neighbors, and Artificial Neural Networks

Daoyu Liu, Ybarra Magtoto, Meng Zhou, Xin Zhou

I. Introduction

Leukemia is a deadly cancer that affects millions of people in the world. However, it should be noted that there are multiple types of leukemia. There are four primary types of leukemia: Acute Myeloid Leukemia (AML), Chronic Myeloid Leukemia (CML), Acute Lymphoblastic Leukemia (ALL), and Chronic Lymphoblastic Leukemia (CLL).

Treatment for these primary types of leukemia are different. What chemotherapy and medication regimen works for an ALL patient may not necessarily work for an AML patient. Doctors want to try to maximize the efficiency of their treatments while limiting or eliminating the amount of toxicities from radiation a patient is subject to. Thus, it is important to be able to classify whether a patient is one type of leukemia or another.

Our research aims to classify ALL vs AML type patients using three machine learning methods: Support Vector Machines, K-Nearest Neighbors, and Neural Networks. Our study also aims to compare the performance of these methods.

II. Algorithms

For our study, we used three different methods so that we could compare their effectiveness in determining leukemia types. These methods are briefly described below.

SVM

Support Vector Machines (SVM) are supervised learning that output an optimal hyperplane which categorizes testing samples [2]. The optimal hyperplane is produced by mapping the training vectors into a higher dimensional space which then get separated by maximizing the margin between different classes [3].

The reason why we thought SVM may be a good candidate for our project is that our data vectors are already in a very high dimensional space.

KNN

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor [4].

ANN

Artificial Neural Networks (ANNs) is a type of computational model based on a large collection of artificial neurons that are roughly analogous to axons in the human brain.

Unlike other methods, such ANN systems can be trained from examples, rather than explicitly programmed, and excel in areas where the solution or feature detection is difficult to express in a traditional computer program. Like other machine learning methods, neural networks have been used to solve a wide variety of tasks, like computer vision and speech recognition, which are difficult to solve using ordinary rule-based programming [5].

III. Related Research

In 1999, Golub et al. wrote in their paper *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring* that from gene

expressions in DNA microarrays they collected from 72 patients, with their method of class predictions, they were able to predict with 100% accuracy [1].

Their approach in this study was to find the genes that were highly correlated to the AML-ALL distinction. They accomplished this using their own neighborhood analysis algorithm for feature selection. Their neighborhood analysis was simple; if a gene was expressed highly in one class and expressed uniformly low in the other then it was a good "indicator" gene. Their neighborhood analysis showed that roughly 1100 genes were good indicators.

In creating their predictor, they used the most informative genes, and based on their expression levels, make a prediction. Each gene casts a weighted vote for a class, where the weight is determined by the degree to which the gene correlates to the class [1].

After creating their approach and developing a prediction model, their predictor was 100% accurate in predicting ALL and AML on a sample of 34 leukemia patients.

IV. Methods

Data pre-processing methods

The data we used to train and test our models was the same dataset used in the Molecular Classification of Cancer (Golub et Al. 1999) research study. The data set given to us was a truncated version. It contains the gene expressions for 2000 genes from 72 leukemia patients, classified as either having ALL (classified as a 1) or AML (classified as a 2).

Before any testing, the data is normalized. This is because we use PCA to reduce the dimensionality of our data. The reason normalization is important before PCA is because PCA tries to maximize the variance of each feature, and variance may be large in some components relative to others, skewing what becomes a principal component. Our data was normalized to the range of [0,1].

Data was split into training, validation, and testing sets. The training set is used to get our models and the validation set is used to tweak our parameters and get the best model based on validation accuracy.

In order to make training and validation sets comparable, we split the data manually. The golubsmall dataset is initially split with 38 patients in the training set and 34 patients in the testing set. However it is important to note that in the training set, there are 27 ALL and 11 AML patients. We get the validation set from this initial training set by manually choosing the amount of ALL and AML patients go into both. Thus, our training set became 14 ALL and 6 AML patients and our validation set became 13 ALL and 5 AML patients.

Dimensionality Reduction (PCA)

The patient gene expression dataset contains 2000 features, which is a fairly large amount, and we believed it would be a good idea to reduce these features. To reduce the features Principal Component Analysis was used (PCA). PCA maps the data linearly to a lower-dimensional space. It transforms it while also maximizing the variance of the data. There is data loss, however the most important variance is maintained.

The motivation for doing PCA was that we believed not all gene expressions would directly correlate to a leukemia type, and that there would be some that are more likely than others to be better "predictors." Another reason is that neural networks are oftentimes computationally to run. Reducing features lightens the load computationally.

Our study used MATLAB's native function princomp() and pca() to apply PCA to our dataset. The princomp() function was used in the K-NN script to generate comparisons of the accuracies computed from 10-dimensional to 200-dimensional. There is a script named pca_determination.m which shows the identical results that princomp() and pca() produces. The

only difference between the results of the two functions are the length of the coefficient matrix. The pca() function cuts off the redundant columns after the number of features that cover 100% of the variance, in our case 37, whereas princomp keeps all of them. The pca() function was used in SVM classification per MATLAB's recommendation.

After applying PCA, we found that 95% of the total variance is represented by 28 dimensions. We chose 95% as a threshold because it covers most of the variance of the data.

SVM methods

A third-party library for SVM, LIBSVM, is available for implementation while avoiding user interactions with the complicated deep structure of the model. LIBSVM was used for our project because of its power and simplicity. During the implementation, three kernel functions, were selected for comparison. These kernels are linear, polynomial, and radial basis function (RBF).

There are several important parameters in each kernel function and a fair value needs to be determined during the validation process to optimize our training such that the model can predict the testing data at its best. The technique used to determine these parameters is grid search. For linear SVM, 100 different cost values ranging from 0.1 to 10 were tested; for polynomial SVM, 100 different degree values ranging from 0.1 to 10 were tested; for RBF SVM, 100 different values of the parameter gamma from 1e-11 to 1e-09 were tested.

KNN methods

For KNN, the native MATLAB function fitcknn() is used to do the training. A model was obtained by applying the function to the training data. The MATLAB built-in predict() function was called to test our model. A confusion matrix, from which the accuracy was computed, was generated eventually.

To determine what number of neighbors produces the best model, odd integers from 1 to 9 were tested to avoid the chance of getting the same label of k neighbors.

ANN Methods

Neural Network Structure:

The choice of structure of the neural network is ambiguous. The number of hidden layers and neurons are dependent on the total inputs. However, there is no specific rule or equation to calculate the optimal number of neurons or layers. We tried MATLAB's built-in function patternnet() and feedforwardnet() from the Neural Networks Toolbox to construct the neural network. The patternnet() function has self optimization on learning rate, but needs 1000 iterations for each run. The feedforwardnet() function is fast, but does not have any optimization.

Implementing patternnet() with a while loop that stops when the sum of squared error is less than a specific threshold is very slow due to the 1000 iterations per run. Therefore, we decided to use 2-dimensional or 4-dimensional inputs on the network built with patternnet(). The network constructed using feedforwardnet() runs fast but it is unstable. Thus, we applied 28-dimensional inputs on the network built with feedforwardnet():

Function & Reduced Dimension:	NN structure:
-patternnet(), 2D	[4] / [10] / [4 4] / [10 10]
-patternnet(), 4D	[4] / [10] / [4 4] / [10 10]
-feedforwardnet(), 28D	[100] / [100 100]

Train & Backpropagation:

We use train() which has default setup of "scaled conjugate gradient descent" which is a good method for training a feedforward neural network.

Training Stop Condition:

After many running tests, we found that the results were very randomized due to the initial weights of the neural network setup. Then we implemented a while loop stopping condition, which use a variable called sum of squared error to compare the results.

V. Experiments & Results

After creating models from our validation, we obtained the following accuracies

SVM Results:

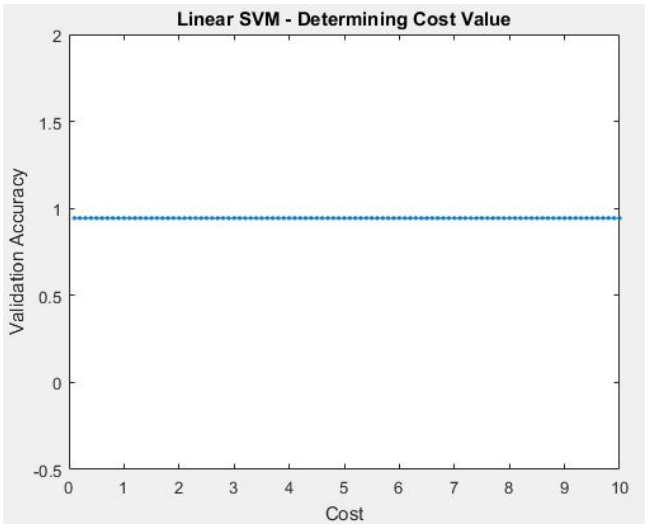
Before PCA:

Kernel Function	Validation Accuracy	Testing Accuracy
Linear	94.44%	91.18%
Polynomial	94.44%	91.18%
RBF	94.44%	76.47%

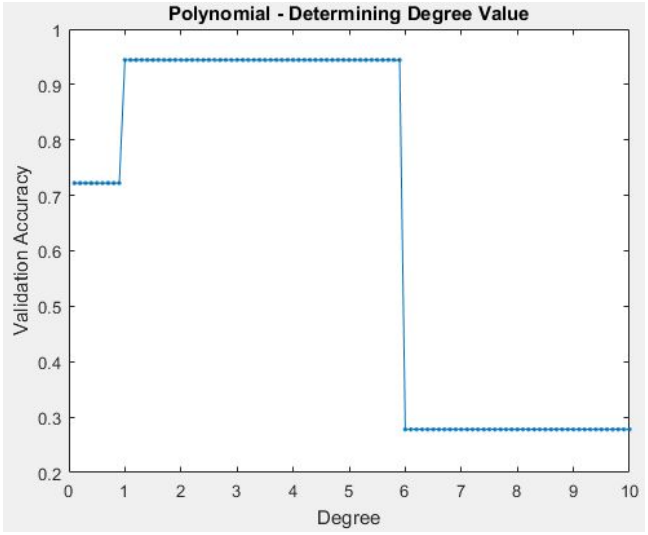
After PCA:

Kernel Function	Validation Accuracy	Testing Accuracy
Linear	94.44%	91.18%
Polynomial	94.44%	91.18%
RBF	94.44%	76.47%

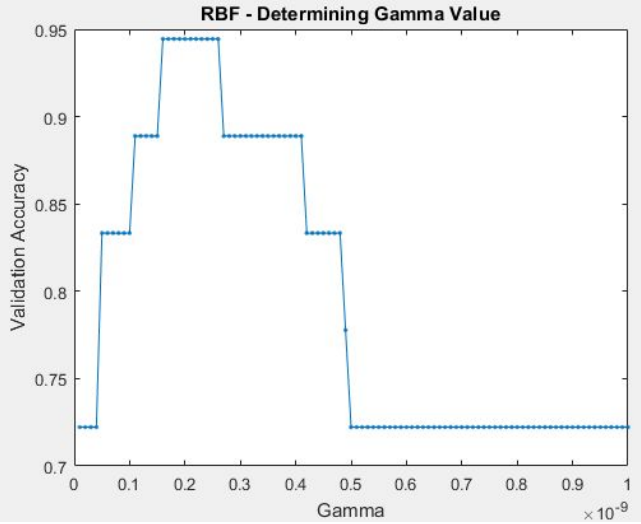
Parameters Determination (For simplicity, only the figures for non-PCA trials are shown):



Best cost value = 0.1 (both non-PCA and PCA)



Best degree value = 1.0 (both non-PCA and PCA)



Best gamma value = 1.6e-10 (non-PCA)
Best gamma value = 1.4e-10 (PCA)

KNN Results:
Before PCA:

Validation Accuracy	Testing Accuracy
94.12%	94.12%

After PCA:

Validation Accuracy	Testing Accuracy
94.12%	94.12%

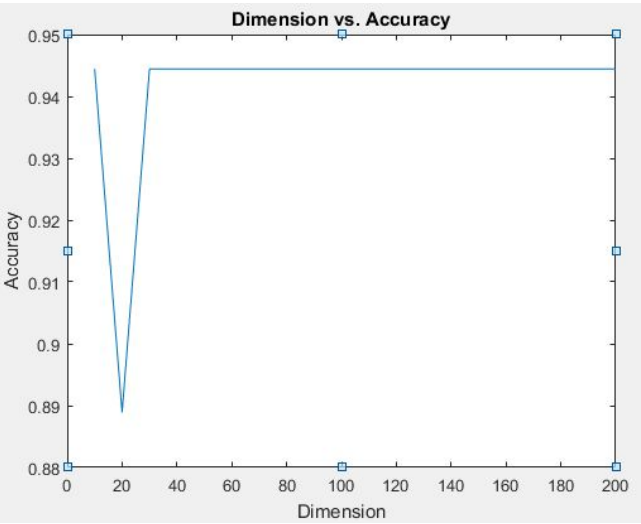
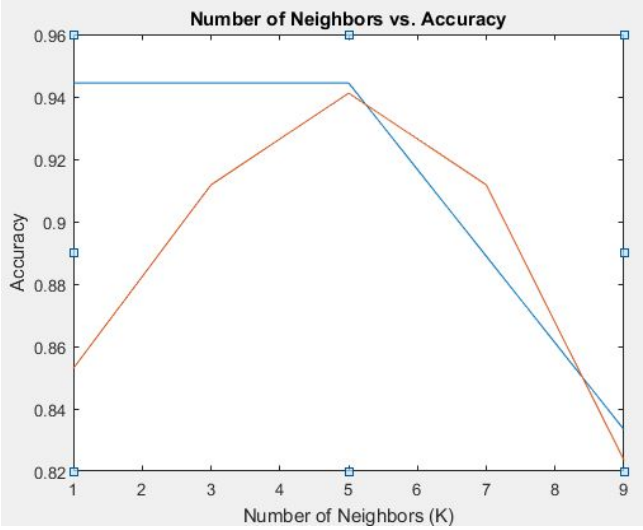


Figure. Dimensionality Determination



Number of Neighbors (K) = 5

ANN Results:
2 PCA Dimension -patternnet():

Case #	NN Struct	Train ACC	Train SSE	Test ACC	Test SSE
1	[4]	92.1%	3.39	82.35%	3.85
2	[10]	86.8%	2.94	91.17%	3.35
3	[4 4]	81.57%	3.75	82.35%	4.22
4	[10 10]	89.45%	2.07	91.17%	3.54

4 PCA Dimension -patternnet():

Case #	NN Struct	Train ACC	Train SSE	Test ACC	Test SSE
5	[4]	92.1%	3.16	73.5%	6.13
6	[10]	94.7%	2.49	91.1%	3.58
7	[4 4]	92.1%	3.66	79.4%	5.52
8	[10 10]	92.1%	3.29	76.5%	5.88
9	[20 20]	84.2%	3.71	64.7%	8.21

28 PCA Dimension -feedforwardnet():

Case #	NN Struct	Train ACC	Train SSE	Test ACC	Test SSE
10	[100]	97.3%	3.08	55.8%	35.98
11	[100 10]	N/A	N/A	N/A	N/A

Applying [100 100] froze the computer.

VI. Discussion/Conclusion

SVM, KNN, ANN are similarly effective at classifying AML and ALL. After using PCA for dimension reduction, it does not seem to have a noticeable effect on SVM or KNN methods results. With and without PCA returned us similar accuracies for using SVM and KNN. However, increasing PCA reduced features slow down the ANN processing and result in bad accuracies.

Thus, SVM, KNN can be used in a lab setting interchangeably in terms of classification for good results. Instead of basing classification on appearance and characteristics, we can use these methods to accurately and quickly classify a patient's leukemia type, which will result in better treatments, and less exposure to unwarranted toxicities due to chemo.

Golub et Al. achieved better results with their method compared to our three methods, as they achieved 100 percent accuracy in predicting on their testing set. However, with more focus on one method, as opposed to three, we believe we could increase our accuracy.

As mentioned above, PCA did not seem to have any noticeable effect on our accuracy. This could be just something in terms of this dataset. Applying PCA for dimension reduction on other datasets for classifying diseases may have noticeable and wanted results.

Remember, this is only for predicting the type of leukemia a patient has and only classifying between ALL and AML types. There are two other types of leukemia that may require different types of treatment. Although this study shows that these models can accurately predict AML and ALL, it is unknown without further research if gene expression analysis is also effective in predicting other types of diseases and cancers. However, from our findings on this particular dataset, these algorithms can be used interchangeably to achieve accurate results.

Our study, unlike the original Golub study, uses a smaller dataset. The dataset from their particular study had 6817 gene expression recorded per patient versus our 2000 gene expression per patient in our dataset. Having more of this data, would bode well for SVM, which performs well in higher dimensional problems. Another limitation of this study is that the sample size seems fairly small. Having a larger sample size would help give us more confidence in our results. Achieving the accuracy rates with our models which were only trained on 38 patients causes a little concern.

It would make sense for future research to use SVM, KNN, and ANN on other gene expression DNA microarray datasets and for other diseases to see how well these models perform. We could also compare methods of feature selection and feature transformation on this same dataset. Our study only used PCA for dimensionality reduction. It would be a good idea to test the efficacy of LLE (Locally Linearly Embedding), LDA (Linear Discriminant Analysis) to see if there is any positive effect on accuracy.

Overall, these methods for classification worked reliably and with consistent results.

Reference

[1] Golub, T. R., *et al.* (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286**, 531.

[2] OpenCV. Introduction to support vector machines. Retrieved on May 11, 2017, from: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[3] Hsu, C. W., Chang, C. C., & Lin, C. J. (2016). A practical guide to support vector classification. Retrieved on May 11, 2017 from: http://www.csie.ntu.edu.tw/~cjlin

[4] Wikipedia (2017). K-nearest neighbors algorithm. Retrieved on May 11, 2017 from: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] Wikipedia (2017). Artificial neural network. Retrieved on May 11, 2017 from: https://en.wikipedia.org/wiki/Artificial_neural_network