CS221 Fall 2016 Project Report

SUNet IDs: udai, nng1, bcui19
Names: Udai Baisiwala, Natalie Ng, Brandon Cui

Task definition

In this update, we are attempting to separate cancerous versus noncancerous samples in a single dataset using various artificial intelligence techniques. In our final project, we will run multiple AI techniques on multiple datasets to explore if the same AI technique works best on various datasets.

Task dataset

The dataset used for this task was obtained from the Gene Expression Omnibus, which is a functional genomics data repository of high throughput gene expression data and hybridization arrays, chips, and microarrays. As of now, we have worked with a single dataset with 70 patients and an initial total of 34,731 genes. Each of these samples has metadata which details if the sample comes from cancerous kidney tissue versus noncancerous kidney tissue.

An example datapoint with only 4 genes for a patient is below (Table 1):

Gene Name	Expression Level
$A23P100011$	-1.584
$A23P100022$	-0.695
$A23P100056$	-0.661
$A23P100074$	-0.641

Table 1: Example gene expression level for a given patient

Task Methodology

Since we have a small dataset, we decided to use a resampling technique, described as follows. We run a standard 10 cross fold validation on our dataset, so we split the dataset into 10 folds, one of the folds was saved for testing and the other 9 folds were used for training. The algorithms and results that we have implemented thus far are detailed below.

Algorithms/Concrete Features - Artificial Intelligence

Baseline - logistic regression

For our task we propose to run logistic regression for our baseline. For this we utilized the logistic regression module from the linear regression package of scikitlearn. For our case logistic regression acts as a predictor of the presence of a dependent variable, cancer, based upon some input independent variables, our features in this case genes. From a probabilistic standpoint we are trying to find $P(Y|X)$ where Y is our classified value, whether or not it is cancerous, and X is our input vector of features, which is the gene expression. For every

timestep in logistic regression it uses maximum likelihood estimatior in order to determine the beta values or the weights assigned to each feature, and then update the weights using gradient descent. As a general baseline the accuracy is as follows:

Cancerous	Prediction Number	Actual Number	percent accuracy
no	16	18	88.89%
yes	46	49	93.88%

Table 2: Logistic Classification Result

Notice that already with logistic regression we have a very high accuracy. Additionally, the false positive rate is only 11.11% and the false negative rate is 6.122%. we distinguish false positive and false negative rates (rather than just reporting accuracy) because they have very different implications from a clinical classification context. Clinically, it is preferred to have a higher false positive rate than false negative rate (it is worse to tell a patient with cancer that he/she is cancer-free).

Our Implementation: SVM

Our current implementation leverages support vector machines (SVMs) from the sklearn package. In general SVMs differ from Logistic regression because they are a non-probabilistic binary linear classifier, so in general they too classify points into one of two classes, cancerous or non-cancerous. However, in particular SVMs try to differentiate between the two classes as widely as possible. in our case this would be given a set of gene expressions and try to determine whether or not this set of genes would be cancerous or non-cancerous. The results from our testing is found below (see Table 3):

Cancerous	Prediction Number	Actual Number	percent accuracy
no	15	18	83.3%
yes	49	49	100%

Table 3: SVM Classification Result

Notice that for an SVM, it has a 0% false negative classification, but it has a 16.6% false positive classification rate. Notably, there is a tradeoff between an SVM and logistic regression here, since we are removing the possibility of a false-negative classification but in the process we are increasing the probability of a false-positive.

Concrete Features - Normalization and Filtration

Normalization

One of the issues with gene expression data is that there is variation in the amount of each sample inputted into the chip that measures expression. Thus, we want to normalize the expressions of each sample. To do this, we sum the expression of the genes for each sample. We then divide each expression by this sum. Essentially, what this does is it scales all the samples to have the same total expression, helping to reduce noise for the experiment.

Initial Filtration

Another issue that we noticed in the above techniques is that since we have 34,731 genes, and therefore 34,731 features, we may be overfitting the models (and they take a long time to run). Thus, we decided to perform some initial filtration on the data to only use important features that are likely to be relevant. We have implemented the following two filtration techniques.

1. Only include differentially expressed genes. Essentially, we look at each gene and see if there is a stastiscally significant difference in the expression for cancer samples and noncancerous samples. We test for significance using a Welch t-test in the sci-py package in python with a significance cutoff of 0.01. We only include genes in our analysis that are differentially expressed between cancerous and noncancerous samples. This prevents overfitting and ensures that our features are likely to be biologically relevant.

2. Ensure a minimum fold-change of 5 (the cutoff was arbitrarily assigned as to cut down the number of features to less than 5000). We do this for a similar reason as using differentially expressed genes.

Following filtration, we have reduced the number of features by approximately an order of magnitude – now have 4,277 genes and thus features.

Post-Filtration Machine Learning Results

Logistic Regression Filtration Results

Running logistic regression utilizing our filtered gene dataset we get the following results (Table 4):

Cancerous	Prediction Number	Actual Number	percent accuracy
no	16	18	88.89%
yes	52	52	100%

Table 4: Logistic Regression Result for the Filtered Dataset

Notice that with our gene filtering technique we are able to reduce the number of false negatives; however, we still retain the same number of false positives.

SVM Filtration Results

Running an SVM on our filtered gene dataset we get the following results (Table 5):

Cancerous	Prediction Number	Actual Number	percent accuracy
no	16	18	88.89%
yes	52	52	100%

Table 5: SVM Result for the Filtered Dataset

Notice that by filtering the number of genes we reduce the number of false positives and retain the same 0% false negative rate, which is crucial in clinical practice.

Runtime Analysis

Below is a table of all of the overall runtimes for our machine learning algorithms, including loading the dataset and running the machine learning algorithms in our cross validation:

Methodology	Time (s)
Logistic Regression w/o Filtration	14.514
Logistic Regression w/ Filtration	1.677
SVM w/o Filtration	11.802
SVM w/ Filtration	1.533

Table 6: Runtimes of different algorithms on filtered on non-filtered dataset

Notice that the amount of computation time was reduced by about an order of magnitude when we ran the machine learning algorithms with the filtered dataset. This is sensible since the number of features was also reduced by approximately an order of magnitude. Thus, with additional filtration, we were able to not only achieve a higher level of accuracy but also a reduction in computation time.

Analysis of Filtration

We notice that filtration only marginally improves the accuracy. This seems to indicate that overfitting is not an issue with our current implementation of SVM and Log-reg. We may also have issues with our classification problem (see below) that makes it very difficult to actually compare the accuracies of different techniques. The filtration step is nonetheless useful in that it leads to a drastic reduction in the runtime without compromising on accuracy, so we will continue to use the filtration step in the future. With an update in classification problems in the future, hopefully we will actually be able to see if our filtration step prevents overfitting.

Issues with current classification problem

The obvious immediate issue that we saw is that our dataset is too small, and therefore does not have enough test samples. Unfortunately, there may be a deeper problem. We realized that perhaps the classification problem of cancerous versus non-cancerous might be slightly too easy. This makes it very difficult to compare the results from different AI techniques (ultimately, we might be saying things like 95% true positive rate is better than 94.8% true positive rate, and there is no way to know that this is not noise or luck from a certain separation of test and validation sets).

The reason that this classification problem is too easy may be because there are definite markers in actual cancerous tissues. Thus, from here on out we will shift to harder classification problems using the same type of data. We are currently looking at two datasets that will enable the following classifications:

1. A dataset that measures circulating blood cells of patients with and without leukemia. This dataset is more representative of something that a clinical test could actually measure.

Instead of measuring the expression of an isolated sample of cancer cells, this dataset instead measures the effect of having cancer on other cells in the body, leading to more noise and more difficult, but more realistic, classification.

2. A dataset that records the relapse rate (whether or not the patient gets leukemia again in the next 10 years) of patients with leukemia. Essentially, this would be an attempt to predict the prognosis of patients, which is a much more difficult classification problem. The markers that lead to relapse are not known, and without a doubt, this dataset will have more noise.

For both of these datasets, we will again use our baseline as an unoptimzed form of logistic regression and our oracle as the best results from clinical tests (see our first writeup for more details on this). These classification problems will allow for a wider range of classification accuracies, so we can actually see how different techniques impact our results. Fortunately, since all of the datasets are formatted identically, all of our code is easily transferable to another dataset, so this is not at all starting from scratch.