Austin Prince

Data Mining

10/26/17

In this project I am using Data Mining techniques to analyze the relevant genes in Alzheimer’s (AD) patients. I have two datasets for this project one containing patients with AD and control dataset of patients who do not have AD. These datasets contain patients and a record of the gene expression levels of 8000 genes in these patients. In the AD patient data set there are 176 patients and in the control dataset there are 188 patients, both of these datasets are matched up in that they contain the gene expression levels for the same 8000 genes in these patients

**Preprocessing**

In order to preprocess my data I translated the *.gex* files which were given to us and put them into an Excel spreadsheet where I could save them as *.csv* files which are much easier to work with in Python, the language I will be using for this assignment. In addition to this I selected only the first 1800 genes from the dataset to analyze in order to reduce the computational time that would be needed to construct the random forest and SVM models.

**Imputation**

The first thing I had to do to this dataset was to account for missing values in the data. In both datasets there were a significant amount of missing values. These showed up as *NA* in the *csv* files. Before I did any analysis with this dataset I had to fill in these values. I decided to use a Gaussian distribution to fill in the values of my missing data. I describe the process in pseudocode below.

translate .csv file into a python DataFrame where it can be easily manipulated

transpose the DataFrame matrix such that the genes (features) are now the columns and the patients (instances) are the rows, this will make RandomForest construction easier in the future

for each column (columns represent features/genes) calculate the mean and var of gene expression for all values that exist (not NaN)

for each value in column

if value is NaN

generate a random number within previously calculated Gaussian distribution and assign it to this value

Using this method of imputing missing data based on a normal distribution we are assuming that our gene expression in patients follows a normal distribution and then randomly generating a data point within that distribution. If there were more instance data, in the form of more patients then the values of the imputed values of the data would be closer to an actual value. Another way to do this would be to use a kNN algorithm in this case but I figured that the ease of computing using a Gaussian distribution outweighed the potentially more accurate kNN estimation. In addition to this I also supposed that given the amount of data points would be enough such that a mis-estimation that is performed using random numbers in a Gaussian distribution would be insignificant in the long run

**Random Forest Feature Selection**

In order to select the genes that are most significant in the development of AD patients I constructed a random forest consisting of 500 trees for each set of frequency and instance percentages specified. I used Python’s scikit learn package to construct the trees and I will briefly describe the pseudocode for constructing a random forest below.

function RandomForest

for i in range num\_trees

A = bootstrap sample from training set

Ta = treeLearn(A)

add Ta to set of trees T

return set of trees T

treeLearn(D)

at each node

cycle through all features in D and all possible splits of features in D

split on best feature (highest information gain)

continue until all data is uniformly classified or pruning criteria is met

return tree

Because a random forest introduces randomness into the construction of decision trees it is a good tool to overcome common problems such as overfitting and bias towards certain features over others.

The decision tree selects attributes based on their information gain and the random forest model randomizes this process so it selects a bootstrap sample from the data. In the random forest that was constructed the features that each tree split on has the highest amount of information gain, which means that it is the most instructive in distinguishing between patients with AD and without AD.

The process of extracting which features were used in the random forest to split on is described in the following pseudocode.

function getFeatures()

for trees in RandomForest

for each split in tree

add to list featureSplit

calculate frequency of features in RF based on how many occurences of feature in the list featureSplit

For this RandomForest test I constructed 18 RandomForest models using a different amount of features and instances from the test data for each tree. In the initial processing of the data I selected only the first 1800 genes from the model (out of 8000 total genes). For the individual RF models I fixed the features (genes) for the RF at 70% and then constructed trees for 50 to 90% of instances with a step size of 5%. I did the same thing with instances and features, fixed instances (patients) at 70% and varied features (genes) by 5% from 50% to 90%. In my preprocessing I set up my DataFrame so that I could select a certain percent of instances and be returned a subset of all the data containing that many instances. The RandomForest in scikitlearn has a feature that allows you to specify the percent of features to include in your RandomForest so I used that to select the number of features.

I plotted the genes that appear in the RF on the attached figure. As can be seen in this figure some genes appear much more frequently in the RF than others. This suggests that these features are more important in determining the class of a patient. Since each split in a tree is based on which feature decreases entropy by the most the genes that appear with higher frequency in the RF are more important/descriptive in separating the classes (AD/non AD patients).

In a random forest the features that are chosen out of the whole subset are chosen randomly, which means that over the course of construction of all of the trees these genes will have the same probability of appearing in the RF. It can be concluded that if a feature (gene) appears in trees in the RF more often then this feature (gene) is more important in separating the classes.

From the above logic we can say that if we want to compare two genes based on their frequency in the RF then the gene that has a higher frequency in the RF is more decisivie in splitting the classes, in that it provides the most information on separating the two classes. And from this we can say that this gene is more important in determining if a patient has AD.

From my data I extracted the following 100 genes that appeared with the highest frequency in the constructed RF. In order to get this data I summed the frequency over all tests (different percentages of feature and instance selection) and determined the genes that had the highest frequency over all tests. These genes are below.



**Support Vector Machine**