Cody Falzone SID: 860929046

April 7, 2017

CS171 PS1

Problem 1:

Let H = probability you have foobarinosis,
let T = Test Positive

P(T | ~H) = 1%

P(~T | H) = 0.2%

P(T | H) = 1 - 0.002 = 0.998

P(~T | ~H) = 1 - 0.01 = 0.99

P(H) = (1/4000)

= 0.00025

P(~H) = 1 - H

= 0.99975

P(T) = P(T | H) * P(H) + P(T | H) * P(~H)

= (0.998*0.00025) + (0.01*0.99975)

= 0.010247

P(H |T)= [P(T | H) * P(H)] / P(T)

= [0.998 * 0.00025] / 0.010247

= 0.02434855898

= 2.43%

The probability you have the disease is 2.43%.

Problem 3:

The data being plotted is concerning the housing values in suburbs of Boston. There are 14 attributes including one class attribute, which is the median value of owner-occupied homes in $1000's.

There is a strong positive linear correlation between attributes 6, which is the average number of rooms per dwelling, and output/target y attribute 14, which is median value. This is clear because there is a good line that can fit the data with little error, there is little variance. This makes sense because as the average numbers of rooms in a house increases, it is safe to expect the price of the house to also increase.

On the contrary, theres not a strong correlation between attributes 4 and y. For the same x value we get various different values for y, and there is no good fit for this data plot. This makes sense because feature/attribute 4 is the Charles River dummy variable. Therefore, there should be near-zero correlation.

Another correlation seems to occur between attribute/feature 13 and the y. Feature 13 is % lower status of the population and the median value of ower-occupied homes. This correlation is a negative correlation. That is, as the percent of lower status population increases, the price of the house decreases. This also makes sense because it is expected that higher status population live in more expensive homes.

The rest of the plots do not have a strong correlation. Some even have a near-zero correlation with the target/output. This means that the features do not share a relationship with the median value of homes.

Problem 4 Part c:

Ridge regression is the process of determining our weights (w vector) and some offset b to fit our training data in order to use these same weights and offset to predict future unseen instances. In addition to the learning algorithm, there is a term added that is a mechanism used to discourage the learning algorithm from using all of the features, or adjusts how much to consider specific features. This regularization term involves lambda times a summation of our weights. However, this lambda does not penalize our offset term b, which is w0. This regularization term allows us to smooth the plot, that is, as lambda increases, the plot becomes more and more smooth. Therefore, lambda is a hyper parameter constant that controls the bias-variance trade-off.

In the plot generated by the function plotacc, there are two curves plotted with different values for lambda as our x axis and our values for corresponding mean squared error as the y axis. Therefore, the two curves represent how the mean squared error for our training or testing set changes as our lambda increases.

When lambda is very small we are considering the weights of each features found very highly. For the training set, this means there is very low amounts of error because the weights were generated or trained on this set of data. For the testing set, when lambda is still very low so we are considering our features very strongly, the error is higher than that of the training set. This is due to the fact that the weights were fit to the training set, and now being used to predict our testing set.

On the contrary, when lambda is higher we don't consider our features as much. Therefore, we see a drop in error in our testing set , and an increase of error in our training set. When we start to consider the weights of features less, the error of the training set increases because the weights were fit to that set. The error decreases for the testing set because the weights were not perfectly fit to this set and lambda is “smoothing” the plot out. However there is a optimal value for lambda that considers the weights and simpleness/smoothness of the graph just right. This appears to be around a lambda value in between 10 and 100, around 40-50, say 42. This is where the error drops in our testing set and is not too high in our training set. When we start to bring lambda lower then this, we are overfitting, likewise when we bring the value of lambda higher than this we are under fitting. When lambda is at the highest value, we are not considering any features at all and because lambda does not apply to our first weight w0, which is b, we are left with the value of our offset b being the different between the two curves error values.