# EECS 440: Project 3 Writeup

Austin Feydt (apf31) and Mathanki Singaravelan (mxo174)

November 4, 2017

a) All of these were calculated by running on the full sample, because running on folds would take forever for spam. Also, using a value of $\lambda = 1$ causes the log likelihood to blow up (because of how small I had to choose my learning factor), so I had to use a much smaller value of $\lambda$)

**Voting Accuracy:**

*Naive Bayes:* 88.2%

*Logistic Regression:* 97.8%

**t-test for 95% confidence:**

*Naive Bayes:*

$$\hat{e_D} = 1 - 0.882 = 0.118, \quad V(E_D) = \frac{0.118(0.882)}{440} = 0.00023654, \quad \sigma = 0.0154$$

$$95\%Cl = (0.118 \pm 1.96 * 0.0154) = (0.0878, 0.1482)$$

*Logistic Regression:*

$$\hat{e_D} = 1 - 0.978 = 0.022, \quad V(E_D) = \frac{0.022(0.978)}{440} = 0.0000489, \quad \sigma = 0.00699285349$$

$$95\%Cl = (0.022 \pm 1.96 * 0.00699285349) = (0.00829, 0.03570)$$

**Volcanoes Accuracy:**

*Naive Bayes:* 40.0%

*Logistic Regression:* 98.0%

**t-test for 95% confidence:**

*Naive Bayes:*

$$\hat{e_D} = 1 - 0.4 = 0.6, \quad V(E_D) = \frac{0.4(0.6)}{2231} = 0.000108, \quad \sigma = 0.0104$$

$$95\%Cl = (0.6 \pm 1.96 * 0.0104) = (0.5796, 0.6204)$$

*Logistic Regression:*

$$\hat{e_D} = 1 - 0.98 = 0.02, \quad V(E_D) = \frac{0.02(0.98)}{2231} = 8.785e-6, \quad \sigma = 0.002964$$

$$95\%Cl = (0.02 \pm 1.96 * 0.002964) = (0.1419, 0.25809)$$

**Spam Accuracy:**

*Naive Bayes:* no convergence :(

*Logistic Regression:* no convergence :(

*t-test for 95% confidence:* N/A

b) This time, I used cross validation. I calculated the accuracy for a few different bucket values:

*Volcanoes accuracy, 2 buckets:* 40.88%

*Volcanoes accuracy, 5 buckets* 40.88%

*Volcanoes accuracy, 10 buckets* 40.88%

*Volcanoes accuracy, 25 buckets* 40.88%

As we can see, the effect on the accuracy by changing the number of buckets is nonexistent for my implementation. This is definitely not right. We should expect that, as you increase the number of buckets, the accuracy should also increase. This is because adding more buckets allows for more distinction between different continuous valued attributes. As the number of buckets increases, it begins to appear more like a continuous distribution again, rather than a discrete one. I'm not sure what bug in my code is causing the accuracy to not increase.

c) Again, I used cross-validation, and compared the accuracy across different m-values on *voting*:

*Voting accuracy, m = 0:* 94.53%

*Voting accuracy, Laplace smoothing:* 94.76%

*Voting accuracy, m = 10:* 94.53%

*Voting accuracy, m = 100:* 94.32%

In this case, m has a slight affect on the overall accuracy, but not as much as it theoretically should. We see that Laplace smoothing behaves the best. And, if m is too large, then the accuracy begins to decrease, which is also to be expected, since we are now putting too much emphasis on the prior distribution on the class labels. However, the numbers aren't nearly as drastically different as they should be.

d) As mentioned earlier, $\lambda$ too large results in the log likelihood blowing up. I was able to compare small values of $\lambda$ though:

*Voting accuracy,* $\lambda = 0.0001$*:* 97.77%

*Voting accuracy,* $\lambda = 0.0005$*:* 97.76%

*Voting accuracy,* $\lambda = 0.001$*:* 97.73%

*Voting accuracy,* $\lambda = 0.01$*:* 96.14%

Increasing lambda, which in turn increases the importance of keeping weights small, causes accuracy to begin to drop. This is because we are limiting our hypothesis space by cutting down what possible values our weights can take on, meaning that we could be excluding a really strong hypothesis.

While the program doesn't quite work in all cases, it does demonstrate great classifications for voting, which consists of all nominal features. One nice numpy feature we took advantage of was using the digitize feature to quickly bucket our continuous attributes. This saved so much time, rather than having to manually determine the bucket bounds. Also, we fixed a mistake with our sigmoid function that was causing overflow errors in calculating the log-likelihood. This fix probably would have greatly helped our artificial neural network from Project 2.