Lab 4

Milo Knowles

November 16, 2018

1 Part I: Naive Bayes

1.1 Part A

1.1.1 Maximum Likelihood Parameter Estimates

Let K_{spam} be the number of examples with label "spam", and K_{ham} be the number with label "ham". Finally, K is the number of training examples. Then,

$$K_{spam} \triangleq \sum_{k=1}^{K} \mathbb{1}(C_k = spam) \tag{1}$$

$$K_{ham} \triangleq \sum_{k=1}^{K} \mathbb{1}(C_k = ham) \tag{2}$$

To get a MLE estimate of s,

$$s_{MLE} = \sum_{k=1}^{K} \mathbb{1}(C_k = spam) = \frac{K_{spam}}{K} \tag{3}$$

To estimate q_i,

$$q_{i,MLE} = \frac{1}{K_{spam}} \sum_{k=1}^{K} \mathbb{1}(K_{spam}) \mathbb{1}(Y_i^{(k)} = 1) \tag{4}$$

Similarly, to estimate p_i,

$$p_{i,MLE} = \frac{1}{K_{ham}} \sum_{k=1}^{K} \mathbb{1}(K_{ham}) \mathbb{1}(Y_i^{(k)} = 1) \tag{5}$$

1.2 Part B

My naive bayes model correctly classifies **47 out of 49 spam messages, and 31 out of 51 ham messages**. For some reason, it shows high accuracy for spam messages, but often misclassifies ham messages as spam.

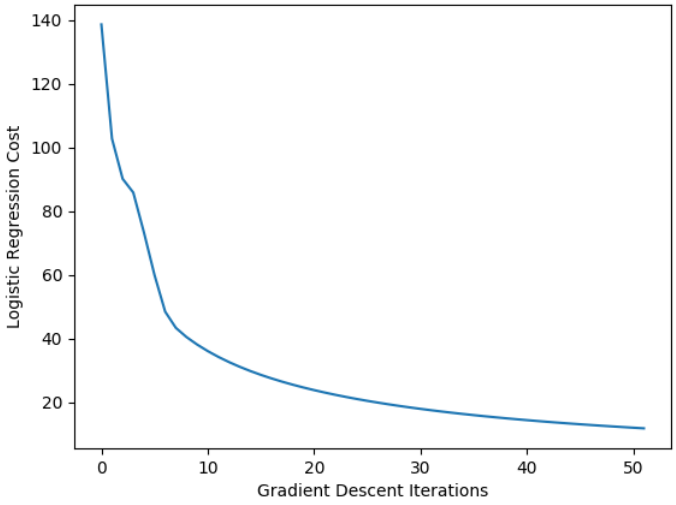

Figure 1: (Part E) Training loss binary features, with default meta-parameters of $\eta = 0.5$ and $t_{convergence} = 1e-3$.

1.3 Part C

$\hat{s} = 0.001 \implies$ 47 out of 49 spam messages, and 38 out of 51 ham messages.
$\hat{s} = 0.5 \implies$ 47 out of 49 spam messages, and 34 out of 51 ham messages.
$\hat{s} = 0.999 \implies$ 47 out of 49 spam messages, and 30 out of 51 ham messages.

Because my model seems biased towards spam messages, decreasing s should help more of the ham messages be classified properly. As seen above, choosing a very small value for s gives a low prior to spam messages, causing more of the true ham messages to be classified correctly.

In general, the sum of log terms due to feature likelihood dominates the resulting probability.

2 Part II: Logistic Regression

2.1 Part E: Testing Logistic Regression

My logistic regression classifier performs much better than the Naive Bayes. It correctly classifies **44 out of 49 spam messages, and 46 out of 51 ham messages**.

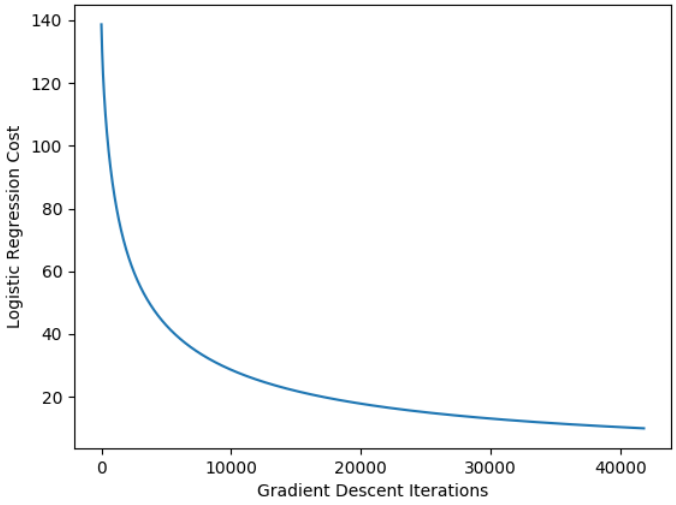

Figure 2: (Part F) Training loss using word frequency features, with $\eta = 4.0$ and $t_{convergence} = 1e-6$. Considerably more iterations of gradient descent are used to converge to a low value of loss. This leads to overfitting, and worse performance on the test set.

2.2 Part F: Alternative Features

I chose to use *word frequencies* as an alternative for the simple binary features. For each file, I count the number of occurrences of each word, and divide by the total number of words in the file. Dividing by the total number of words provides normalization, and prevents the size of an email from skewing results.

Using word frequency features, I had to reduce the convergence threshold, and raise the learning rate to achieve a loss value that was comparable to training on binary features ($\phi(\theta) < 10.0$). My guess is that normalizing the features by document length created numerically small features, and small gradients as a result. With the default learning rate, gradient descent took very small steps and would prematurely hit the convergence threshold.

This new set of features correctly classifies **39 out of 49 spam messages, and 38 out of 51 ham messages**. This is noticeably worse than logistic regression with binary features. I believe that using word frequency features leads to considerable overfitting on the small number of training examples. Although word frequency provides additional information about an email, it does not seem to a feature that generalizes well to test data.