# Naive Bayes and Logistic Regression

Carolyn Atterbury and Keira Haskins
June 17, 2020

## 1 Introduction

In this paper we use the Naive Bayes method, as well as Logistic Regression with Gradient Descent in order to do a topic categorization task. Given a set of words which contain the number of times each word appears in the document, we identify one of twenty topics that correspond with the set of words. Using the Naive Bayes method we are able to get a top accuracy score of 88% when classifying new data, and with Logistic Regression we are able to get an accuracy score of 81%.

## 2 Methodology

For this topic categorisation task, we have 12000 vocabulary sets that correspond to a particular topic. Overall there are 20 topics, and 61189 words which are stored in csv format, where each row is an ordered set of words in alphabetical order, and each column contains an integer representing how many times the word appeared in that corresponding document. The last column has an integer from 1 to 20 which is the id corresponding to a particular topic.

We select X word sets from the data to use as training data, while leaving the remaining Y word sets for testing purposes. To reduce the time and memory needed to store this dataset, we store the training and testing data in a sparse compressed row storage (CRS) matrix format. From there, we use two methods to train the data: Multinomial Naive Bayes, and Logistic Regression with Gradient Descent.

### 2.1 Multinomial Naive Bayes

The Naive Bayes method uses conditional probabilities and Bayes' Theorem to classify a topic $Y^{new}$ given a set of words $X$. In order to use Naive Bayes method, we assume that each word is conditionally independent of each other, given the topic. The Naive Bayes approach classifies the $k$th value as follows,

$$P(Y = y_k|X_1...X_n) = \frac{P(Y = y_k)P(X_1...X_n|Y = y_k)}{\sum_j P(Y = y_k)P(X_1...X_n|Y = y_j)}. \tag{1}$$

Using the assumption that each word is conditionally independent to each other given the class, we can simplify the equation by turning $P(P(X_1...X_n|Y = y_k)$ into $\prod_i P(X_1|Y = y_j)$. As we simply want to find the maximum value, we can take the log to get the simplified equation,

$$Y^{new} = [\log_2(P(Y_k)) + \sum_i (X_i^{new}) \log_2(P(X_i|Y_k)]. \tag{2}$$

We can compute the maximum likelihood estimate (MLE) by,

$$P(Y_k) = \frac{\text{number of texts that were classified } Y_k}{\text{total number of examples}}. \tag{3}$$

The Maximum A Posteriori Estimation (MAP) by,

$$P(X_i|Y_k) = \frac{(\text{number of } X_i \text{ in } Y_k) + (\alpha - 1)}{(\text{total words in } Y_k) + ((\alpha - 1) \cdot (\text{total number of words}))} \tag{4}$$

### 2.2 Multinomial Logistic Regression with Gradient Descent

The Logistic Regression method attempts to maximize the conditional likelihood of a set of data. In our case we want to assign a probability of between 0 and 1 for each of our 20 possible classes to a new set of data. We update the weights of our model for the gradient descent step by,

$$W^{t+1} = W^t + \eta((\Delta - P(Y|W,X))X - \lambda W^t) \tag{5}$$

where $\Delta$ is a $k$ x $m$ and $\Delta_{ji} = \delta(Y^i = y_y)$. k in this case is the number of classes and m is the number of documents or training examples. $\eta$ is the learning rate, $\lambda$ is a penalty term, n is the number of attributes for each example, $X$ is an $m$ x $(n+1)$ matrix of examples, $Y$ is a vector of correct classifications, $W$ is a matrix of weights, $P(Y|W,X)$ is a matrix of probability values.

Once we compute $W$ we classify our new dataset by taking the argmax of $hY$, where we compute $Y$ to be equal to $Y = WX^T$

## 3 Code Design

Overall our code is split up amongst a number of files including files for both Naive Bayes and Logistic Regression, files for main, utility, and testing.

### 3.1 Utility

Since there are similarities between the Naive Bayes and Logistic Regression algorithms, we have general functions that apply to both methods. These are all stored in a utility.py file. These methods allow us to read in the data from a csv, and partition the data into training and testing data. After partitioning the data, the training and testing information is stored in sparse CSR format and serialized so it can be read by the Naive Bayes and Logistic Regression algorithms. The data for Naive Bayes and Logistic Regression are read from the csv, parsed, converted to CSR format, and serialized in very similar ways, although for Logistic Regression we add a column of 1s in the 0th position of the matrix.

In addition we have a function for determining the accuracy score after the algorithm has been run. The Naive Bayes and Logistic Regression programs output a file containing the classified answers. The answers in this file are compared against the original answers in order to obtain an accuracy score.

### 3.2 Naive Bayes

The Naive Bayes algorithm reads the serialized CSR formatted data structure containing the training and testing data, and converts it back into CSR format. We then use the training data to build up a matrix of conditional probability values, where an entry $(i,j)$ corresponds to $P(X_j|Y_i)$ where $X_j$ is a particular word, and $Y_i$ is a particular class. The conditional probability matrix is calculated by taking the total number of $X_j$ in $Y_i$, then adding an alpha term as seen in equation 4. This value is then divided by the total number of $Y_i$ plus a similar alpha term. We also compute all of the class probabilities, $P(Y_i)$ and store them in a data structure. After computing these data structures we can then classify the data.

In order to classify the data, we iterate over each row of the testing data, $X_new$. We then iterate over all the classes $k$, and compute $x = \log(P(Y_k))$. We then iterate over all the words $X_i$ and multiple the value of $X_i$ with $P(X_i|Y_k)$ which is stored in the conditional probability matrix. The computed value is then added to $x$. After iterating through all the words, the value $x$ is added to an array containing the corresponding $x$ value for each class. After the class loop terminates, we take the argmax of all the $x$ values to find the class that is most likely to correspond to the set of words. We store this class value, and then go on to the next $X_new'$ in the testing data set.

Once we classify all the training data, we then use our accuracy score utility method to determine the overall accuracy of the Naive Bayes approach.

### 3.3 Logistic Regression and Gradient Descent

The Logistic Regression algorithm first reads the serialized data structure containing the training and testing data, and converts it back into CSR format. The training and testing data are then normalized, so that each row (excluding the first element in the row which was set to 1), sums to 1.

We then create a matrix $W$, containing 0 values, which contains the initial values for the weight matrix. We then create a matrix $Delta$, where each column stores a 1 in index corresponding to the class value in each row in the training data set. All of the other values are set to 0. We then call a logistic regression method, and pass in the $W$ and $Delta$ matrices, as well as additional parameters $eta$ and $lambda$ which were read in through the command line.

The logistic regression method loops over a fixed number of iterations $N$, and computes a new $W'$ each time, that is based off the previous value for $W$. The computation for $W'$ is as follows: $W' = W + eta * ((Delta - WX) * X - (lamda * W))$, where $X$ is the training data set, and $WX$ represents $P(Y|W, X)$. We compute the $P(Y|W, X)$ value by taking $W * X^T$ and then raising each element $x$ to $e^x$. After $N$ iterations, we will have a modified weight matrix $W$, which we can use to classify the testing data.

In order to classify the testing data, we compute the class values $Y'$ by

$$Y' = \text{argmax}\left(W\widetilde{X}^T\right)$$

where $\widetilde{X}$ is the testing data. We then write these values to a csv and calculate the accuracy score by using the accuracy score utility method.

## 4 Results

In the Naive Bayes implementation we have an accuracy score that varies between 82% and 87% depending on the value of Beta. The accuracy score in reference to Beta can be seen in Figure 1. The highest score of 87% occurs with a Beta value of $10^{-2}$, as the values for Beta increase, the accuracy go down.

The logistic regression accuracy scores can be seen in Figure 2, 3, and 4, for varying values of Lambda and Eta. Overall, the accuracy sores were lower than those of Naive Bayes, with a top accuracy of 81% when the algorithm was run when $\eta = 0.009$ and $\lambda = 0.007$.

## 5 Conclusion and Discussion

In this paper we show how to implement both multinomial naive Bayes and logistic regression for classification and demonstrate relatively high accuracies for both on a dataset consisting of 61188 words and 20 different newsgroup classes. For naive Bayes we are able to achieve an accuracy of around 88% for a $\beta$ value of about 0.001. For logistic regression at 10,000 steps, $\eta = 0.009$ and $\lambda = 0.007$ we get an accuracy of 81%.

While implementing both naive Bayes and logistic regression, we ended up encountering a number of challenges that would cause us to get accuracies closer to 5%. For naive Bayes it ended up being because we forgot a part of our update step and did not add values correctly. For logistic regression we eventually figured out by help of Jamie, that our delta matrix did not work quite right. After this we were able to improve our accuracy significantly.

## 6 Questions

1. In your answer sheet, explain in a sentence or two why it would be difficult to accurately estimate the parameters of this model on a reasonable set of documents (e.g. 1000 documents, each 1000 words long, where each word comes from a 50,000 word vocabulary). [5 points]

   It would be difficult to accurately estimate the parameters of this model on such a set of documents because the number and size of the documents would need to be quite a bit larger for such a vocabulary. Such a model requires data sets that scale exponentially with the number of features. In this case we would want a data

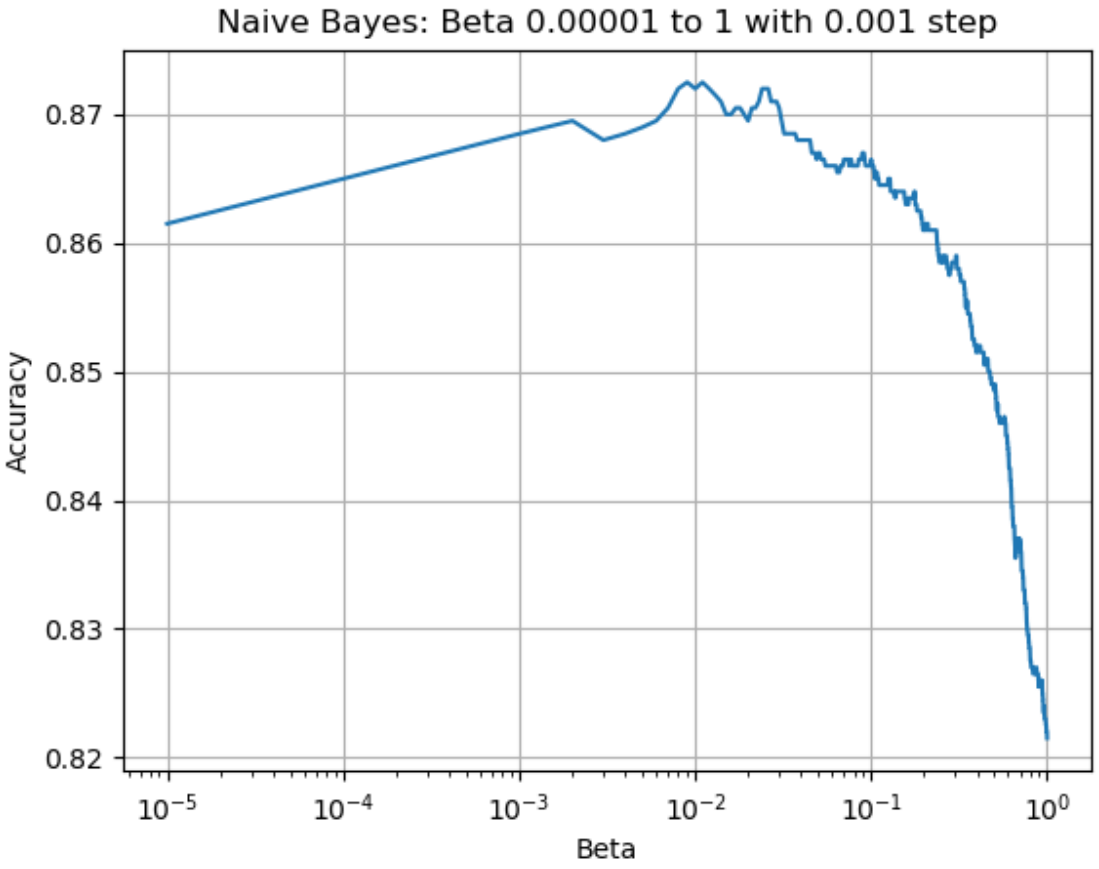



Figure 1: Naive Bayes with 0.001 step size.

set which is exponentially larger than our set of features, 50,000, whereas we only have a data set of around 1,000,000, we would instead want a data set closer to 2,500,000,000.

2. Re-train your Naive Bayes classifier for values of b between .00001 and 1 and report the accuracy over the test set for each value of b. Create a plot with values of b on the x-axis and accuracy on the y-axis. Use a logarithmic scale for the x-axis (in Matlab, the semilogx command). Explain in a few sentences why accuracy drops for both small and large values of b [5 points]

   Beta ends up acting as a smoothing value, and it affects the strength of the prior. The stronger or larger the beta, the more we ignore differences in the data. If beta is too small, then outliers may have more of an effect on the classification process, but if beta is too large then the differences between the data entries will not be as pronounced and the prior information becomes less useful. (Figure 1)

3. Re-train your Logistic Regression classifier for values of $\eta$ starting from 0.01 to 0.001, l = 0.01 to 0.001 and vary your stopping criterium from number of total iterations (e.g. 10,000) or = 0.00001 and report the accuracy over the test set for each value (this is more efficient if you plot your parameter values vs accuracy) . Explain in a few sentences your observations with respect to accuracies and sweet spots [5 points]

   We trained the logistic regression classifier from $\eta$ starting from 0.01 to 0.001, l = 0.01 to 0.001 and varied the stopping criteria. In Figure 2 we stopped after 1000 iterations. In Figure 3, we stopped after 5000 iterations. Finally, in figure 4, we stopped after 10,000 iterations.

   Overall, the accuracy increased as we increased the numbers of iterations. Figure 4 with 10000 iterations was the most accurate, with the highest accuracy score of 81%. The plots with 5000 and 10000 iterations were more

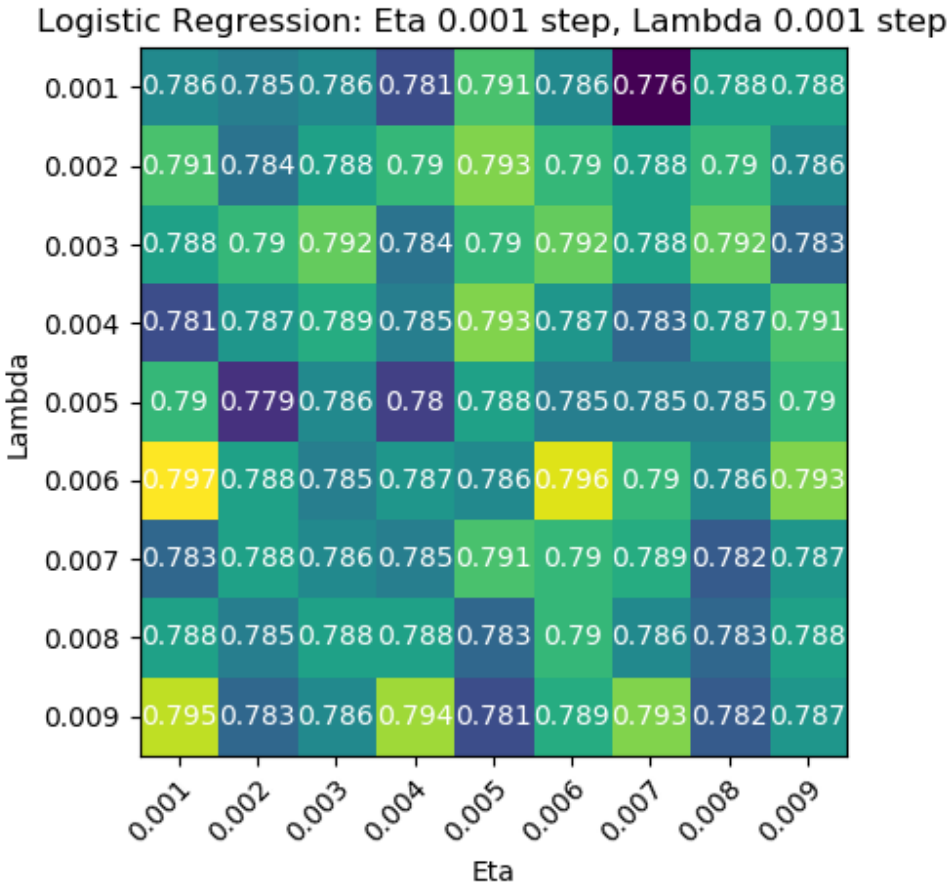



Figure 2: Logistic regression with 0.001 step size, 1000 iterations.

similar to each other overall, whereas the plot with 1000 iterations was more different. This could be because the algorithm was farther from converging on the correct weight values at that point. Overall the accuracy was highest for $\eta = 0.009$ and $\lambda = 0.007$.

4. In your answer sheet, report your overall testing accuracy (Number of correctly classified documents in the test set over the total number of test documents), and print out the confusion matrix (the matrix C, where $c_{ij}$ is the number of times a document with ground truth category j was classified as category i). [5 points]

   Our overall testing accuracy was 85.6% for Naive Bayes, and 79.65% for logistic regression.The confusion matrix can be seen in Figure 5.

5. Are there any newsgroups that the algorithm(s) confuse more often than others? Why do you think this is? [5 points]

   News groups 20(talk.religion.misc) and 16(soc.religion.christian), 7(misc.forsale) and 4(comp.sys.ibm.pc.hardware), 3(comp.os.ms-windows.misc) and 4(comp.sys.ibm.pc.hardware) were confused more than other groups. The first pair (talk.religion.misc and soc.religion.christian) likely have a lot of words in common as they both are related to religion. There are likely words that are unique to those topics, or that are at least uncommon outside the topic of religion. Similarly 3 and 4 (comp.os.ms-windows.misc and comp.sys.ibm.pc.hardware) both have to do with computers, and PC computers. It is likely that there is specialized terminology in those groups as well. Groups 7 and 4 (misc.forsale and comp.sys.ibm.pc.hardware) are not immediately related, but it could be possible that computer related items are often up for sale.

6. Propose a method for ranking the words in the dataset based on how much the classifier 'relies on' them when

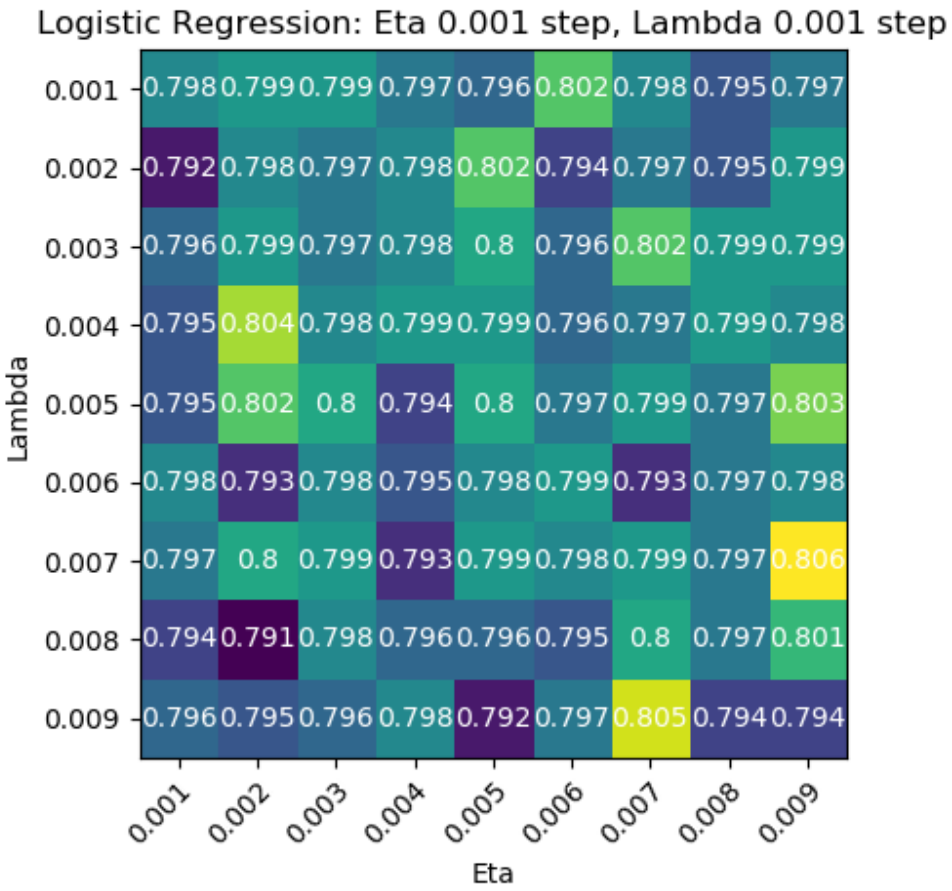



Figure 3: Logistic regression with 0.001 step size, 5000 iterations.

performing its classification (hint: information theory will help). Your metric should use only the classifier's estimates of P (Y) and P (X|Y). It should give high scores to those words that appear frequently in one or a few of the newsgroups but not in other ones. Words that are used frequently in general English ('the', 'of', etc.) should have lower scores, as well as words that only appear extremely rarely throughout the whole dataset. Finally, in your method this should be an overall ranking for the words, not a per-category ranking.[5 points]

In order to get a ranking of words in the dataset based on how much the classifier 'relies on' them, we calculated the conditional entropy $H(Y|X = x)$ for each word $x$, and then divided by the total instances of the word $x$ in the dataset. By calculating the conditional entropy, we were able to get a sense of the amount of information a class $Y$ was able to get from a specific word $x$. High conditional entropy means that more bits of information was needed in order to use the word determine a class based on a specific word. By dividing by the total instances of each word $x$ in the dataset, we were able to ignore words that were incredibly rare. This way, we target words that are both freqent, while also having a low conditional entropy.

7. Implement your method, set b back to 1/|V |, and print out the 100 words with the highest measure. [5 points]

   The table containing the first 100 words with the highest measure as well as their conditional entropy values $H(Y|X = x) \cdot \frac{1}{\sum_i x_i}$

8. If the points in the training dataset were not sampled independently at random from the same distribution of data we plan to classify in the future, we might call that training set biased. Dataset bias is a problem because the performance of a classifier on a biased dataset will not accurately reflect its future performance in the real world. Look again at the words your classifier is 'relying on'. Do you see any signs of dataset bias? [5 points]

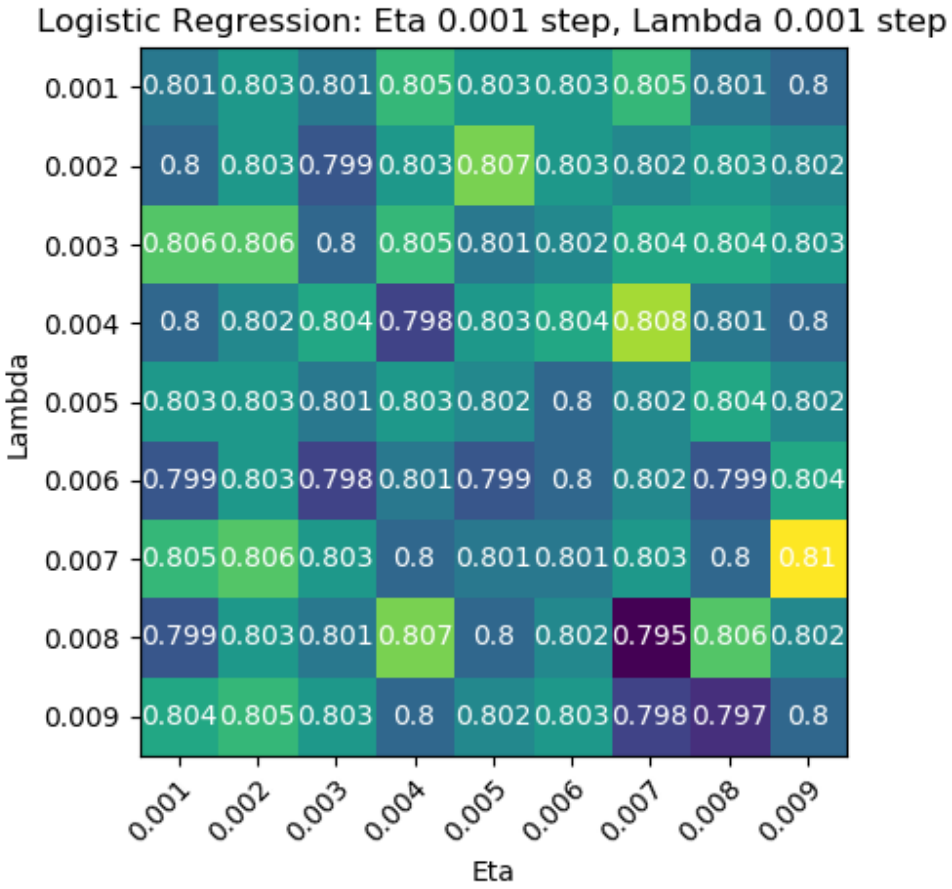
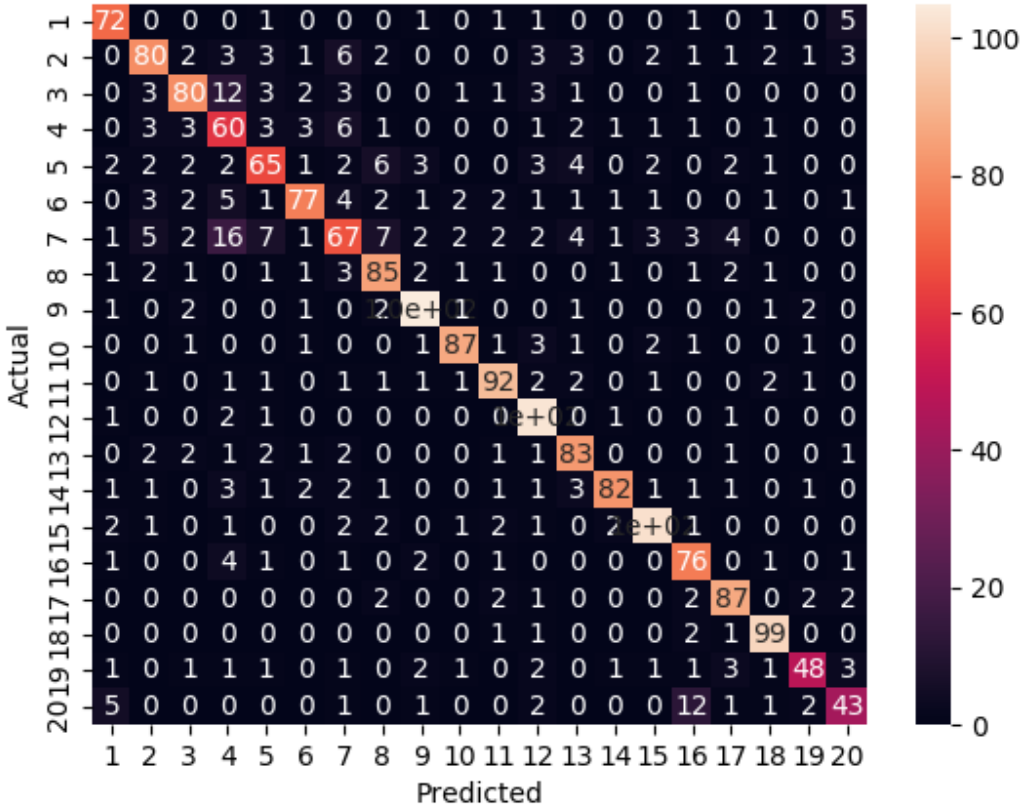



Figure 4: Logistic regression with 0.001 step size, 10000 iterations.

We see a lot of words pertaining to electronics, science, religion that correspond to the newsgroup categories. There are 6 categories that have to do with electronics, 4 that have to do with science, 3 that have to do with religion. This dataset seems to be biased towards a number of these types of categories. It is likely that there would be a larger variety of categories that people may be interested in using this type of classifier for, therefore it seems to be a bit biased or limited in scope.




2

Figure 5: Confusion matrix generated from a logistic regression run where $\eta = 0.01$ and $\lambda = 0.01$.

| | |
|---|---|
| 0.0120981013601408 | sandwich |
| 0.011204159353169893 | counselling |
| 0.009267269878467028 | mem |
| 0.009077808399325807 | hulk |
| 0.009067832206358482 | magellan |
| 0.00840171440297521 | armory |
| 0.008010725895241488 | triangulates |
| 0.007060763074748773 | moguls |
| 0.006889503814447673 | oddly |
| 0.006293077597601309 | clears |
| 0.006153746650361505 | xff |
| 0.0060751283464804 | abiding |
| 0.005990236304094543 | electrics |
| 0.0059383589235180115 | cheer |
| 0.005886657906472192 | jbrandt |
| 0.005826294830945416 | trek |
| 0.0057521743655198024 | antimatter |
| 0.005712339454768443 | mitsubishi |
| 0.0056777856257669065 | somwhere |
| 0.005552955368599424 | recover |
| 0.005491557839347412 | sdscpub |
| 0.005484160490295596 | superb |
| 0.005279143124730506 | gle |
| 0.005275539581223029 | xevent |
| 0.005163895904192544 | chips |
| 0.005142288121222069 | plavsic |
| 0.005106331431510958 | got |
| 0.0050325305573529445 | firehawk |
| 0.004955810852248687 | stderr |
| 0.004932955873680124 | veos |
| 0.004924682752356375 | copperud |
| 0.004883393678085363 | huub |
| 0.00468488818267948 | successor |
| 0.004675201078540893 | exhaustive |
| 0.004574092974362231 | unm |
| 0.004560977933292182 | importing |
| 0.004534232065682351 | unaided |
| 0.004426348257420589 | egg |
| 0.00440653393084751 | unaccelerated |
| 0.0044008567754391745 | valentine |
| 0.004395089924811891 | problems |
| 0.004330934081938665 | rim |
| 0.004258915118063995 | pieces |
| 0.004249070251284033 | tufnel |
| 0.004231739623550694 | growth |
| 0.004215601089217545 | innocence |
| 0.0041209001953568785 | philip |
| 0.004109906047760895 | dt |
| 0.0040867961198271215 | cows |
| 0.004086597238833344 | intelligibly |

| | |
|---|---|
| 0.004082398542505464 | renderings |
| 0.004078447416275514 | peruse |
| 0.004075216550895741 | dkmiller |
| 0.00404923856394164 | rovers |
| 0.004021482950562163 | auras |
| 0.003915562518102052 | yo |
| 0.00390422978946447 | cicero |
| 0.003902829044305945 | outliers |
| 0.00389021262071442 9 | dsr |
| 0.0038701699503953 38 | limitation |
| 0.003870116649330459 | denver |
| 0.0038700325436864707 | grace |
| 0.003855437083382283 | reid |
| 0.0038323763754599853 | ric |
| 0.0038051689307420007 | rats |
| 0.003769193754765818 | officially |
| 0.00376650067089420 7 | revealing |
| 0.003744311229260774 | ruthless |
| 0.0037433263760243717 | enforce |
| 0.0037251171973499008 | inspired |
| 0.0037017477394581484 | xli |
| 0.0036980444669590913 | believable |
| 0.0036907681903283153 | reinterpretation |
| 0.003679338305638894 | snowhite |
| 0.0036635745672393277 | beep |
| 0.00365892230829877 | lived |
| 0.0036153578462291883 | sourced |
| 0.0036044716770822164 | latest |
| 0.003592134859615033 | belgian |
| 0.00359071120297543 4 | lando |
| 0.00357710954463394 7 | recalls |
| 0.00357027541258435 9 | qlfd |
| 0.0035630836379488134 | granted |
| 0.0035612614183293855 | study |
| 0.0035518255771345435 | rle |
| 0.00354040802254662 6 | pointing |
| 0.0035170402835835 7 | satisfaction |
| 0.0035040317424100856 | parameter |
| 0.00349992329531061 1 | gentiles |
| 0.00349442766835994 8 | breads |
| 0.0034793873443368 41 | opener |
| 0.0034731639504168347 | miya |
| 0.00347068504831818 8 | leadership |
| 0.0034645393677256134 | ethercard |
| 0.00342985459772441 6 | solomon |
| 0.00342491048503289 9 | scip |
| 0.0034184765349822626 | logos |
| 0.00341703312900131 3 | zeh |
| 0.00341661680174725 1 | clamp |
| 0.0034115879983355165 | oxygen |