CS429

Naive Bayes Document Classifier

The University of New Mexico

Team:
Eros Espínola González
Mauricio David Zaragoza Ibarra

I. Code description

What our code does is first of all is read the number of documents in the training.label file for each one of the classes and save those numbers in an array called *cy*, this is used to compute the priors through MLE. Then with the same file's information the class of each one of the documents is saved in the *doc_label* array. Once that information is retrieved, the next step is to get the bag of words with the train.data file, with this file we get the count for each one of the words in the vocabulary used in each one of the 20 different classes, so we end up with a *bag* matrix that has on the columns all the words in the vocabulary and on the rows all the classes.

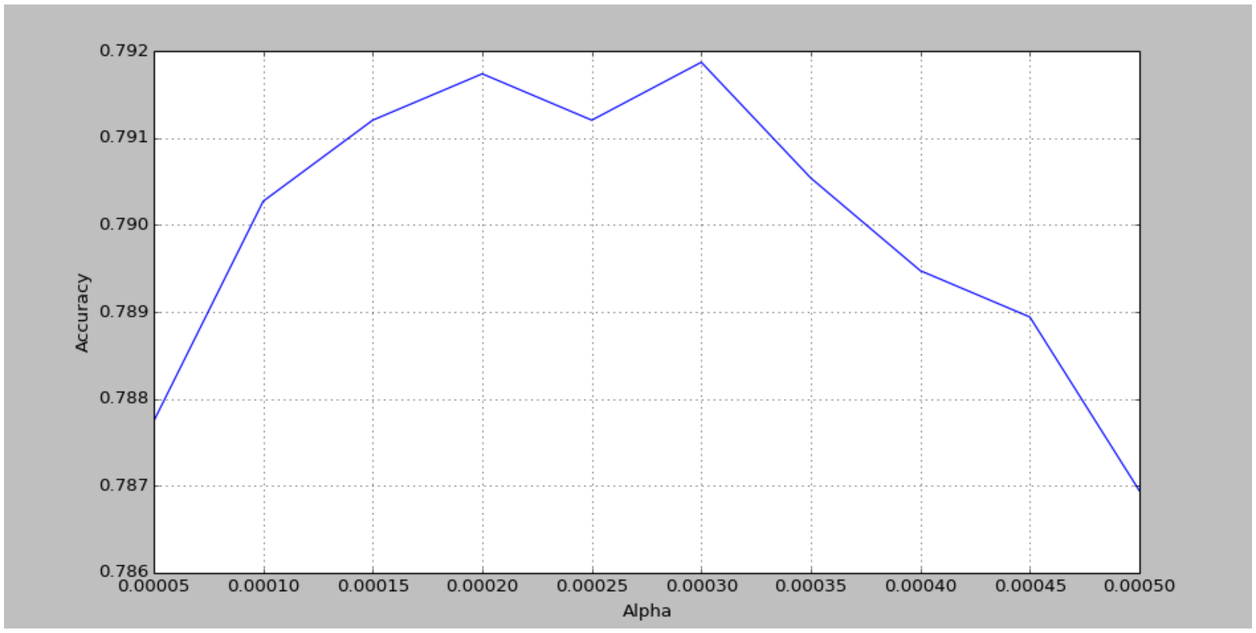
The next step is to calculate the MAP estimates for each one of the elements in the bag, so first we get the number of appearances of each word and we saved it in an array called *total_words*, then an hallucinated value (alpha) is added to each value of *total_words*, then the values of each word in the bag are divided by the *total_words* + *alpha* array, this will give us regularized MAP estimates in the bag.

Once we have the MAP estimates, the test.label file is read and for each document their class is saved in the *test_label* array, then the words used and their respective count are save for each one of the documents, once all the words and their counts are stored in a vector, each sample is classified by computing the dot product of the current sample vector and the log-posteriori probabilities plus the priors, this reduces to a standard matrix-vector-multiply which yields the probability of the sample corresponding to each class, then we get the label with the higher probability.

Every time the software hits on a document, a counter is incremented, so at the end we can calculate the accuracy. Also, the confusion matrix and the top 100 words are printed in files called confusion.txt and rank.txt. To calculate the top 100 we get the higher probability P(X|Y) in the vocabulary, this allow us to see which words appear frequently in a few newsgroups, so the classification can rely on the words with the high score.

II. Accuracy

To compare accuracy among different setting, we trained the Naive Bayes classifier for different values of alpha, as you can see in the chart below, the best performance of the classifier is with an alpha of 0.0003, which gave us an accuracy of approximately 0.792. Bigger numbers than 0.0003 will just start reducing the accuracy of the classifier. Also, smaller numbers of alpha will reduce the accuracy, except by 0.0002, which gave us the second biggest accuracy of the Naive Bayes classifier. This make sense since large values of alpha simulate documents having every possible word in them, causing low-frequency words which the classifier relies the most on to appear uniformly across all documents, the opposite effect happens on smaller values of alpha which cause the classifier to rely very heavily on such low-frequency words, thus decreasing accuracy significatively.



III. Questions

Question 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 wordcomes from a 50,000 word vocabulary). [3 points]

There is an extremely large number of possible arrangements for a vocabulary of 50,000 words in 1000-word documents which makes this problem computationally intractable.

Question 2: 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 cij is the number of times a document with ground truth category j was classified as category i). [7 points]

Alpha: 1.63430738053e-05
Correctly classified documents: 5897
Total number of tested documents: 7505
Accuracy: 78.57%

Confusion Matrix:

249	0	0	0	0	1	0	0	1	0	0	2	0	3	3	23	2	3	4	26
0	**287**	13	14	9	21	4	1	1	0	1	11	8	5	11	2	1	0	0	0
1	33	**204**	56	19	21	4	2	3	0	0	11	5	11	9	2	1	0	6	3
0	11	29	**276**	21	1	11	2	1	0	1	4	32	1	2	0	0	0	0	0
0	17	13	31	**268**	0	12	2	2	0	0	3	21	8	4	0	1	0	1	0
0	54	16	6	4	**284**	1	1	3	0	0	5	3	6	4	0	1	1	1	0
0	7	4	31	15	2	**273**	16	8	1	2	0	7	4	6	0	2	1	2	1
0	3	1	2	0	0	15	**330**	17	0	0	1	13	0	4	2	0	0	6	1
0	1	0	1	0	0	2	28	**359**	0	0	0	3	1	0	0	1	1	0	0
0	0	0	1	1	0	3	1	2	**352**	17	0	1	3	3	5	1	1	5	1
2	0	1	0	0	0	2	1	2	4	**382**	0	0	0	0	1	3	0	1	0
0	3	0	3	4	1	0	0	0	1	2	**360**	3	2	2	0	9	0	5	0
3	20	4	24	7	3	8	10	6	0	0	22	**264**	10	7	1	3	0	1	0
4	7	0	2	0	0	3	5	4	1	0	1	9	**323**	7	7	6	5	7	2
1	8	0	1	0	3	1	0	1	0	1	4	6	5	**342**	3	2	1	12	1
11	2	0	0	0	2	1	0	0	0	0	0	0	2	1	**359**	0	1	2	17
1	1	0	0	0	1	1	2	1	1	0	4	0	5	2	1	**301**	5	23	15
12	1	0	1	0	0	1	2	0	2	0	2	1	0	0	6	4	**325**	18	1
6	1	0	0	1	1	0	0	0	0	0	5	0	10	6	1	63	6	**196**	14
39	2	0	0	0	0	0	0	1	1	0	1	0	2	6	27	10	4	7	**152**

Question 3: Are there any newsgroups that the algorithm confuses more often than others? Why do you think this is? [2 points]

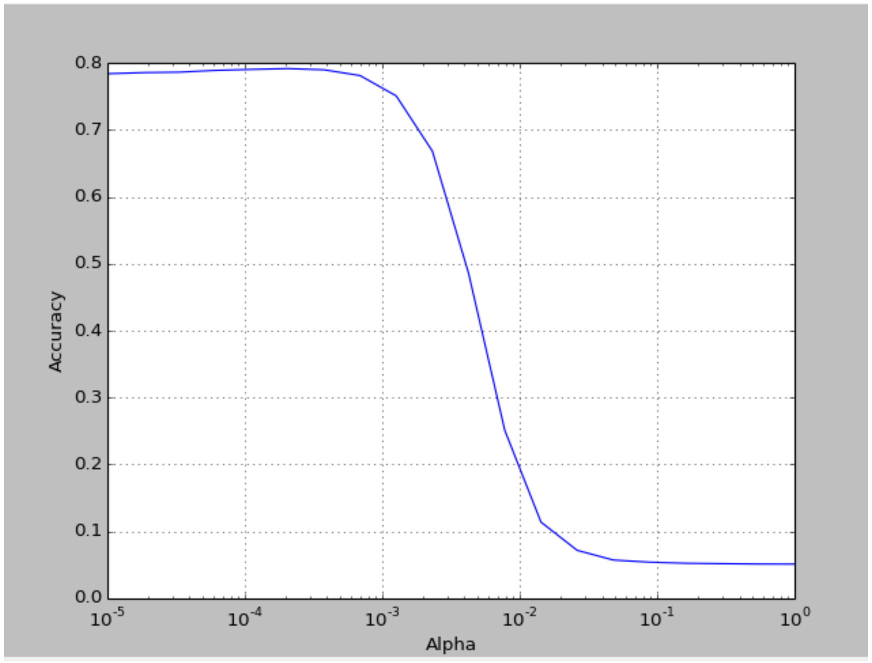
The groups that are more often confused are:

- *talk.politics.misc* and *talk.politics.guns*
- *rec.autos* and *rec.motorcycles*
- *comp.os.ms-windows.misc*, *comp.sys.ibm.pc.hardware*, *comp.sys.mac.hardware*, *sci.electronics*, *comp.graphics* and *comp.windows.x*
- *alt.atheism*, *talk.religion.misc* and *soc.religion.christian*

This happens because those are very similar topics and very similar words are used to write those articles, so the classifier sometimes get confused among the different possible categories.

Question 4: Re-train your Naive Bayes classifier for values of α between 0.00001 and 1 and report the accuracy over the test set for each value of α. Create a plot with values of α 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 α [5 points]

Large values of α simulate documents having every possible word in them, causing low-frequency words which the classifier relies the most on to appear uniformly across all documents, the opposite effect happens on smaller values of α which cause the classifier to rely very heavily on such low-frequency words, thus decreasing accuracy significatively.



Question 5: Propose a method for ranking the words in the dataset based on how much the classifier 'relies on' them when 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 appear extremely rarely throughout the whole dataset. Finally, your method this should be an overall ranking for the words, not a per-category ranking. [3 points]

With our method we are getting the words with the higher probability P(X|Y) in the vocabulary, this allow us to see which words appear frequently in a few newsgroups, so the classification can relies on the words with the high score.

Question 6: Implement your method, set α back to 1/|V |, and print out the 100 words with the highest measure. [2 points]

```
acne                  bdftopcf              xgetwindowproperty    blit
danieley              xsunmono              newroot               xcopyplane
ned                   inexpensively         bytesafter            hardcoded
waveforms             brookline             redrawing             condensed
xterminal             gurung                nitems                paints
chatterjee            raju                  vroot                 pencom
shash                 recruiting            numchildren           hershey
svenv                 bmwmoa                parentreturn          bmc
amit                  haferman              rootreturn            glyph
sunbar                parentcvtargs         scr                   shear
yali                  xmform                xatom                 xybitmap
prakash               xmucvtstringtowidget  getvroot              mkdirhier
xinstallcolormap      ccur                  karlton               retrieves
xstorecolors          westford              xquerytree            perf
defaultcolormap       xcms                  timestamp             xbiff
defaultvisual         xtici                 matrices              imp
displaycells          icaen                 xputimage             activites
defaultscreen         xclrp                 drawable              laundromat
cmap                  xclrs                 merged                progressives
duvvuri               xutil                 dislocation           martinsville
xid                   infrequently          moschetti             eastbrook
arrhythmia            tracking              koala                 mikes
figments              antisocial            lehors                focuses
makedepend            defaultrootwindow     flickers              hellnet
editres               ssetroot              hardwares             clara
```

Question 7: 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? [3 points]

Labels are evenly distributed across the dataset, however, most of the words the classifier is relying on are related to technology, therefore, the classifier might not be able to classify non-technology groups as good as technology-related groups such as comp.windows.x, whose words are fairly common across the top 100 ranked words.