Naive Bayes Project Report

CS429/529 Machine Learning

Team Member:

Chihyu Shen
Haijin He
Christina Xuan Yu

October 19, 2017

Theoretical Background:

In machine learning, Naive Bayes classifier is based on applying Bayes theorem with strong(naive) independence assumptions between the features.

Assume we have the set of categories $\{y_1, y_2, ..., y_n\}$, and x is the description of an instance. The probability of instance E belongs to category y_i :

$$P(y_i|x) = \frac{P(y_i)P(x|y_i)}{P(x)}$$

And P(x) can be ignored since it is a normalization constant, so the equation becomes:

$$P(y_i|x) = P(y_i)P(x|y_i)$$

Then, for instance $\{x_1, x_2, ..., x_m\}$,

$$y* = h(x) = argmax(\ P(y_i)P(x_1, x_2, ...x_m|y_i)\ ),\ for\ i\ =\ 1\ to\ n$$

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 word comes from a 50,000 word vocabulary).

Answer:

There would be a huge computation cost. (e.g. 1000*50000 possible values per one document). Much more parameters are needed for Model 1 than Model 2.

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 c_ij is the number of times a document with ground truth category j was classified as category i).

Answer:

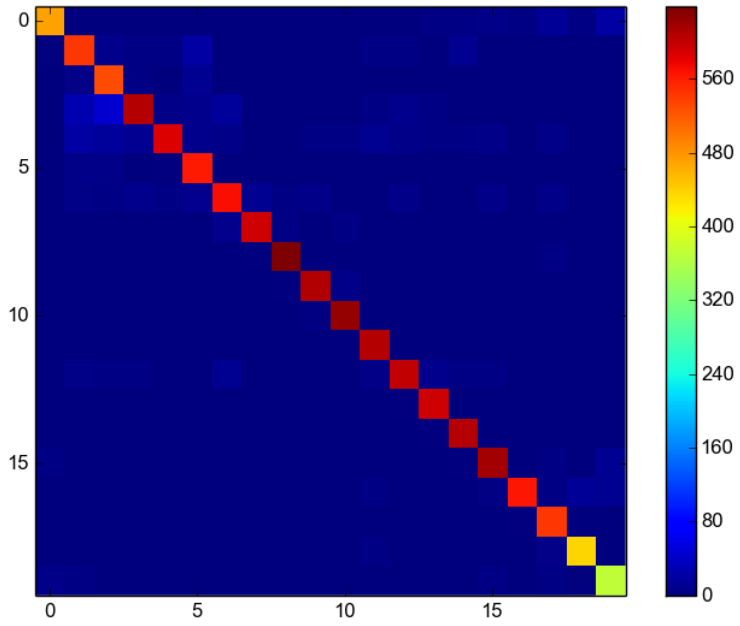
code for implement naive bayes:

```
for row in testReader:
  row = np.array(row)
  row = row.astype(float)
  # use matrix multiplication
  row2=np.reshape(row[1:], (VOC_NUM,1))
  answer =np.dot(voc_in_group,row2)
  # adding PY
  for i in range(GROUP_NUM):
    answer[i]=answer[i]+PY[i]
    #getting the max
    m=max(answer)
  for i in range(20):
    if answer[i]==m:
      r=i+1
      break
  outputWriter.writerow([int(row[0]),int(r)])
```

Overall testing accuracy: 0.88692

Confusion matrix:

Visualized:



code for plotting confusion matrix:

```
import matplotlib.pyplot as plt
plt.imshow(matrix,interpolation='none');
plt.colorbar()
plt.show()
```

Raw data:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	471	1	0	0	1	2	0	0	0	0	0	0	1	3	3	7	3	13	3	21
2	0	544	9	4	3	21	0	1	0	1	0	3	4	2	10	0	0	2	1	2
3	0	7	528	3	2	10	1	1	0	0	0	0	0	0	0	0	0	0	0	0
4	0	28	47	609	5	8	15	1	0	1	0	3	8	4	1	2	1	0	0	0
5	1	20	17	10	585	9	6	2	2	3	3	10	6	4	4	5	1	5	1	0
6	0	6	7	2	0	564	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	0	5	4	9	4	8	571	11	4	6	2	1	5	1	2	5	2	5	1	0
8	0	2	2	0	0	1	8	592	3	1	3	0	0	1	1	0	2	1	1	1
9	0	0	2	0	0	2	1	0	639	0	1	0	0	2	0	1	1	4	2	1
10	0	0	0	0	0	0	1	1	0	611	6	0	0	0	1	0	1	2	1	0
11	0	0	1	1	0	0	1	0	0	4	626	0	0	0	0	1	0	0	0	0
12	0	1	0	0	0	1	0	0	0	0	1	608	0	0	0	0	1	2	2	0
13	0	7	3	4	2	2	10	2	1	1	1	6	600	9	4	4	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	593	2	1	0	0	1	1
15	1	0	0	0	0	1	1	0	0	0	0	0	0	0	607	0	0	1	0	2
16	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	617	0	3	1	10
17	0	0	0	0	0	1	1	1	0	0	0	4	1	1	0	3	565	3	14	12
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	542	2	1
19	1	0	1	1	0	0	2	1	0	0	2	4	0	1	2	0	2	7	435	2
20	6	3	1	0	0	0	0	1	0	0	0	0	0	0	0	3	1	3	2	373

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

Answer:

Top 5 newsgroups that are confused more often:

Ground Truth	Wrong Prediction	Times
3 comp.os.ms-windows.misc	**4** comp.sys.ibm.pc.hardware	**47**
2 comp.graphics	**4** comp.sys.ibm.pc.hardware	**28**
6 comp.windows.x	**2** comp.graphics	**21**
20 talk.religion.misc	**1** alt.atheism	**21**
2 comp.graphics	**5** comp.sys.mac.hardware	**20**

From the chart above we can tell most of the confusions occur in the same bigger category, e.g. group 3 “comp.os.ms-windows.misc” are mis-classified to be group 4 ”comp.sys.ibm.pc.hardware” most often, and they are both in the “computer” category. That makes sense since documents might share similar words in the same category. The only confusion group that is not in the same category is group 20 “talk.religion.misc” and group 1 “alt.atheism”, which is still understandable because they are both religious-related.

Question 4: Re-train your Naive Bayes classifier for values of β between .00001 and 1 and report the accuracy over the test set for each value of β. Create a plot with values of β 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 β.

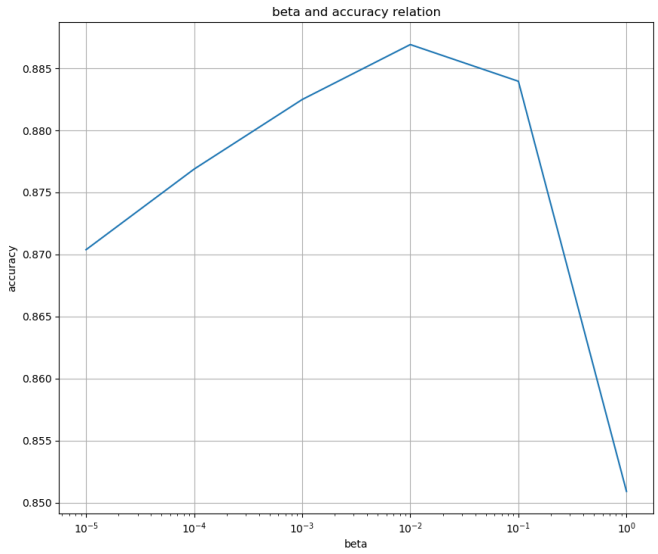
Answer:

the NB classifier is retrained with beta value ranging from 0.00001 to 1.

The beta/accuracy relation is shown below:

beta	0.00001	0.0001	0.001	0.01	0.1	1
accuracy	0.87038	0.87688	0.88249	0.88692	0.88396	0.85090

Plot with Matplotlib:



Observation: the beta value at 0.01 gives the best result at 0.88692. while values smaller or bigger have a tendency to reduce accuracy. The reduction is more significant when beta is very big.

Explain:

When beta is big, it is the dominant factor in the probability calculation, so it will overwhelm any estimation from real data. And when beta is too small, it's not playing any role in the probability estimation, so it's basically like without the smoothing.

An appropriate value of beta not too small and too big is required to get an better estimation.

Question5: 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 extremely rarely throughout the whole dataset. Finally, in your method this should be an overall ranking for the words, not a per-category ranking.

Answer:

Entropy is a good measurement of the uncertainty of a random variable. For a word Xi which is quite indicative of the group it belongs to, we expect the entropy of the $H(Y|X_i)$ to be low. It means its distribution in all groups is highly unbalanced, it only favors one or a few of classes. Of course, we must consider the possibility of rare words that appears in only certain groups just by chance. A smoothing method like adding a Dirichlet prior is necessary.

$H(Y|X_i)$ can be calculated with using this formula:

$$H(Y|X_i) = -\sum P(Y_j|X_i) log P(Y_j|X_i) \ (\ the\ sum\ of\ all\ Y_j)$$

And $P(Y_j|X_i)$ can be calculated with: $P(Y_j|X_i) = \frac{P(Y_j)P(X_i|Y_j)}{P(X_i)}$

We have already calculated $P(Y_j)$ and $P(X_i|Y_j)$ in the NB algorithm. $P(X_i)$ is a normalizing factor, and we get it by summing up all 20 $P(X_i|Y_j)$. Then we normalized $P(Y_j)P(X_i|Y_j)$.

With the above results, we can calcite $H(Y|X_i)$ easily.

Which options work well and why:

1. We have tried the measurement without the normalizing factor. We printed the top 3 words using this measurement: cummington, incontrovertible, earmarks. And we found that each of these 3 words appeared only once and only in one Class. Then we realized this measurement is not good enough. Because the words occurred more than once in only one class should have a lower entropy than those three entropies. Thus we applied the normalizing factor. And our result become more accurate and generalized. For example, each of the new top 100 vocabularies occurred between 21 to 299 times and each of them occurred in different but one class only.

2. We found that ranking words based on $H(Y|X_i)$ is a better measurement than $H(X_i|Y)$, $H(Y|X_i)$ takes account of P(Y). And we saw some variance with P(Y) for different groups. If P(Y) distributes evenly among classes, there would be no difference between the 2 measurements.

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

Answer:

Please see file “Top_100_Vocs.csv”.

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?

Answer:

We do see signs of dataset bias. Among the top 100 vocabularies that the classifier is relying on, lots of them are *not a complete word*. They could be *artifacts* that appears only in one or a few of the newsgroups. Their total entropy is the lowest, but they are very discriminative.

For example, our top 1 vocabulary “wolverine” occurred 135 times total but all 135 occurrences in class 7 “misc.forsale” only.

These 100 vocabularies are separately from 15 different classes. When we *feed* them back to our *classifier*, the classifier got very confused and thus the feedback prediction are all *wrong*. Thus we know the vocabularies which the classifier is relying on is very bias.