Lab 1: Naïve Bayes Classifier for Text Classification

Eric Murphy

February 10, 2017

In this lab we will use an implementation of the Naïve Bayes Classifier to classify texts in some standard corpus. Below are a list of outputs that were asked for within the Lab 1 assignment.

Calculate the Prior $P(\omega_j)$ from labeled training data:

Below is the output of my program

```
Calculating Prior Probabilities from training data
('Class', 'Prior')
(1, 0.04259472890229834)
(2, 0.05155736977549028)
(3, 0.05075871860857219)
(4, 0.05208980388676901)
(5, 0.051024935664211554)
(6, 0.052533498979501284)
(7, 0.051646108794036735)
(8, 0.052533498979501284)
(9, 0.052888455053687104)
(10, 0.0527109770165942)
(11, 0.05306593309078002)
(12, 0.0527109770165942)
(13, 0.05244475996095483)
(14, 0.0527109770165942)
(15, 0.052622237998047744)
(16, 0.05315467210932647)
(17, 0.04836276510781791)
(18, 0.05004880646020055)
(19, 0.04117490460555506)
(20, 0.033365870973467035)
```

All 20 classes of texts seem to be roughly equiprobable. However, there are some notable exceptions with classes 20, 19 and 1 occurring less than 4.3% of the time.

The likelihood predictions from the P

Performance:

Training data using the Bayesian estimation:

The training data produced the following output for class specific accuracy:

```
Calculating class Accuracy for Training set
('Class', 'Accuracy')
(1, 0.9520833333333333)
(2, 0.9156626506024096)
(3, 0.8986013986013986)
(4, 0.9250425894378195)
(5, 0.9443478260869566)
(6, 0.9459459459459459)
(7, 0.781786941580756)
(8, 0.9577702702702703)
(9, 0.964765100671141)
(10, 0.9713804713804713)
(11, 0.979933110367893)
(12, 0.9814814814814815)
(13, 0.9306260575296108)
(14, 0.9781144781144782)
(15, 0.9831365935919055)
(16, 0.9833055091819699)
(17, 0.9853211009174312)
(18, 0.9663120567375887)
(19, 0.9676724137931034)
(20, 0.7952127659574468)
```

It is clear the accuracy on the training set is quite high, with most above 92% accurate. However, the class 20 is under 80% accurate. This might be due to the relatively low Prior probability

The overall accuracy was found to be

```
Calculating Overall Accuracy for Training set
('Overall Accuracy of the BE on the Training Data =', 0.942852072056083)
```

The overall accuracy is approximately 94%, which is quite high.

The confusion matrix is output as

```
[457, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 14, 0, 2, 2, 3]
[0, 532, 8, 16, 1, 9, 2, 0, 1, 0, 0, 2, 1, 1, 2, 4, 0, 0, 2, 0]
[1, 8, 514, 23, 0, 17, 2, 0, 0, 0, 0, 3, 1, 1, 0, 1, 0, 0, 1, 0]
[0, 10, 6, 543, 5, 4, 6, 1, 0, 0, 0, 0, 4, 0, 1, 2, 0, 2, 2, 1]
[1, 4, 3, 6, 543, 2, 1, 0, 2, 0, 0, 1, 2, 2, 2, 3, 0, 1, 2, 0]
[0, 15, 8, 1, 1, 560, 0, 0, 1, 0, 0, 2, 0, 1, 1, 0, 1, 0, 1, 0]
[0, 3, 2, 33, 8, 2, 455, 12, 1, 3, 3, 17, 16, 4, 4, 6, 5, 1, 7, 0]
[0, 1, 0, 2, 1, 2, 4, 567, 1, 1, 0, 1, 3, 0, 0, 1, 2, 1, 4, 1]
[0, 1, 0, 1, 1, 0, 4, 1, 575, 0, 0, 0, 0, 2, 0, 2, 5, 1, 3, 0]
[0, 3, 0, 1, 0, 1, 1, 3, 0, 577, 4, 0, 0, 1, 0, 1, 2, 0, 0, 0]
[0, 0, 1, 2, 0, 1, 0, 2, 0, 0, 586, 1, 0, 0, 0, 2, 0, 1, 2, 0]
[0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 583, 0, 1, 0, 0, 2, 1, 5, 0]
[0, 4, 1, 15, 5, 0, 3, 1, 0, 0, 1, 3, 550, 1, 1, 2, 2, 0, 2, 0]
[0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 2, 581, 0, 5, 2, 0, 0, 0]
[1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 2, 583, 1, 0, 0, 1, 0]
[0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 589, 1, 3, 2, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 2, 537, 1, 3, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 2, 0, 8, 0, 545, 5, 0]
[1, 2, 0, 0, 0, 0, 0, 0, 1, 1, 0, 3, 0, 1, 0, 1, 3, 2, 449, 0]
[17, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 33, 15, 4, 5, 299]
```

This matrix is pretty sparsely populated, which means that there is a definite pattern in the misclassification. For example, class 20 is being frequently misclassified as case 1, and classes 16 and 17. Another noteworthy aspect is that this matrix is not symmetric in structure. For example, while class 20 was misclassified as classes 16 and 17, documents those classes were never misclassified as class 20.

Testing data using Bayesian estimation:

The testing data produced the following output for class specific accuracy:

```
Calculating class Accuracy for Test set with BE
('Class', 'Accuracy')
(1, 0.6918238993710691)
(2, 0.7506426735218509)
(3, 0.5473145780051151)
(4, 0.7729591836734694)
(5, 0.7232375979112271)
(6, 0.7846153846153846)
(7, 0.5916230366492147)
(8, 0.8911392405063291)
(9, 0.9017632241813602)
(10, 0.8790931989924433)
(11, 0.9573934837092731)
(12, 0.9164556962025316)
(13, 0.6641221374045801)
(14, 0.8269720101781171)
(15, 0.8622448979591837)
(16, 0.9472361809045227)
(17, 0.8956043956043956)
(18, 0.8670212765957447)
(19, 0.5967741935483871)
(20, 0.3745019920318725)
```

With overall accuracy being output as

```
('Overall Accuracy of the BE on the Test Data =', 0.7825449700199867)
```

Clearly the Naïve Bayes classifier is less accurate on the testing data than the corpus it was trained on. This is to be expected. In one particular case it is less than 40% accurate. However, there are still accuracies as high as 95%.

The confusion matrix:

```
[220, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 2, 3, 56, 4, 9, 6, 14]
[3, 292, 5, 12, 9, 25, 0, 1, 2, 0, 0, 18, 6, 3, 7, 4, 0, 1, 1, 0]
[2, 28, 214, 58, 11, 31, 0, 1, 1, 3, 1, 15, 1, 4, 5, 6, 0, 0, 9, 1]
[0, 8, 16, 303, 22, 2, 4, 4, 0, 0, 1, 5, 25, 0, 1, 0, 1, 0, 0, 0]
[0, 8, 9, 36, 277, 2, 3, 3, 1, 1, 0, 6, 18, 8, 2, 0, 3, 0, 6, 0]
[0, 41, 9, 10, 2, 306, 1, 0, 2, 1, 0, 10, 0, 0, 3, 1, 1, 1, 2, 0]
[0, 7, 7, 49, 22, 1, 226, 26, 7, 0, 1, 3, 13, 2, 3, 3, 2, 3, 6, 1]
[0, 1, 0, 2, 0, 1, 5, 352, 5, 2, 1, 1, 6, 0, 2, 1, 6, 1, 9, 0]
[0, 1, 0, 0, 0, 0, 1, 21, 358, 2, 0, 0, 1, 1, 0, 1, 4, 2, 5, 0]
[3, 0, 0, 1, 1, 3, 2, 3, 1, 349, 15, 2, 2, 0, 0, 3, 1, 2, 9, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 5, 382, 1, 0, 2, 1, 3, 1, 1, 2, 0]
[0, 4, 1, 1, 2, 1, 1, 0, 0, 0, 0, 362, 3, 2, 0, 2, 7, 0, 8, 1]
[0, 17, 1, 29, 6, 3, 1, 7, 2, 0, 1, 47, 261, 6, 4, 5, 0, 3, 0, 0]
[8, 8, 1, 2, 0, 0, 0, 3, 1, 1, 0, 1, 3, 325, 2, 19, 3, 8, 8, 0]
[1, 7, 0, 0, 0, 1, 0, 0, 1, 0, 1, 4, 4, 4, 338, 6, 1, 2, 21, 1]
[7, 2, 1, 0, 1, 2, 0, 0, 0, 0, 0, 1, 0, 1, 0, 377, 2, 2, 1, 1]
[0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 3, 0, 2, 2, 3, 326, 3, 15, 4]
[6, 1, 0, 0, 0, 0, 0, 2, 1, 1, 1, 3, 0, 0, 0, 10, 5, 326, 20, 0]
[4, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 3, 7, 4, 96, 5, 185, 1]
[38, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 3, 5, 73, 19, 5, 9, 94]
```

This confusion matrix on the test data is in-line with what we would expect knowing the comparison between the test and training data using the accuracies. There is a lot that is similar between these matrices however. This matrix is denser than the training example, which means different misclassifications are happening. However, the frequency of misclassification is roughly in proportion, e.g. classes 1 and 20 is still most frequently classified as class 16.

Testing data using Maximum Likelihood estimation:

The testing data produced the following output for class specific accuracy:

```
('Class', 'Accuracy')
(1, 0.9842767295597484)
(2, 0.36246786632390743)
(3, 0.13043478260869565)
(4, 0.17346938775510204)
(5, 0.10966057441253264)
(6, 0.1358974358974359)
(7, 0.17277486910994763)
(8, 0.12658227848101267)
(9, 0.08312342569269521)
(10, 0.07808564231738035)
(11, 0.13283208020050125)
(12, 0.07341772151898734)
(13, 0.035623409669211195)
(14, 0.07888040712468193)
(15, 0.0663265306122449)
(16, 0.09547738693467336)
(17, 0.04945054945054945)
(18, 0.0851063829787234)
(19, 0.02903225806451613)
(20, 0.03187250996015936)
```

With overall accuracy being output as

```
('Overall Accuracy of the MLE on the Test Data =', 0.14736842105263157)
```

These results are striking, and make me inclined to believe there is a bug in my code. However, the only difference between the MLE and BE in my code are the inclusion of the vocabulary in the denominator. I expected this to be less accurate due to many words appearing only once or not at all in a classes documents. These low counts or rare events mean that there is a high chance of misidentification. It was most surprising to see that the accuracy for class 1 is an order of magnitude higher than many of the others.

Confusion matrix

```
[313, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1]
[231, 141, 5, 1, 1, 6, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0]
[260, 52, 51, 16, 6, 2, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0]
[246, 45, 20, 68, 8, 0, 4, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[267, 49, 7, 10, 42, 0, 1, 0, 0, 0, 0, 1, 4, 2, 0, 0, 0, 0, 0, 0]
[253, 64, 11, 5, 1, 53, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0]
[225, 49, 11, 14, 8, 1, 66, 3, 2, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1]
[319, 15, 1, 0, 1, 0, 3, 50, 2, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 1]
[347, 14, 0, 0, 1, 0, 0, 2, 33, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[346, 5, 2, 2, 0, 1, 2, 2, 1, 31, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[334, 5, 0, 0, 2, 1, 2, 1, 1, 0, 53, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[349, 10, 4, 1, 1, 0, 0, 0, 0, 0, 0, 29, 1, 0, 0, 0, 0, 0, 0, 0]
[322, 38, 1, 2, 3, 3, 4, 1, 0, 0, 0, 4, 14, 1, 0, 0, 0, 0, 0, 0]
[343, 7, 1, 2, 1, 0, 1, 4, 1, 0, 0, 1, 1, 31, 0, 0, 0, 0, 0, 0]
[350, 14, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 26, 0, 0, 0, 1, 0]
[356, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 38, 0, 0, 0, 1]
[340, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 18, 0, 0, 2]
[339, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 32, 1, 0]
[294, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 4, 0, 9, 0]
[233, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 5, 0, 0, 3, 8]
```

This structure leads me to believe that the code does have some bug. There doesn’t seem to be a reason for the misclassification as class 1. Could not debug source of error in time.