Denver Quane and Matthew McChesney

Project 2: Naïve Bayes Answers

Questions 1-7

**Question 1:**

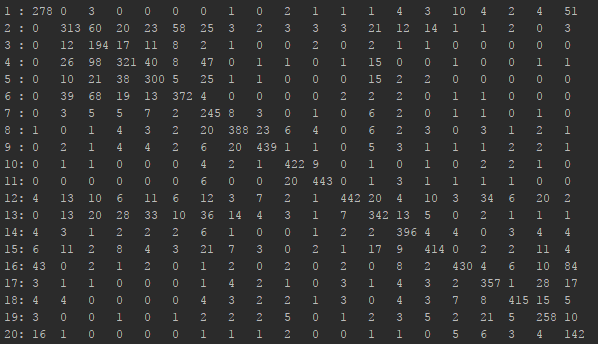
It would be hard to estimate the parameters due to the huge amount of them. We would be calculating 500001000 different parameters. This is because, for every individual document label prediction P(Y), we have to expand the terms where P(Y | Xi), where Xi is the word in the vocabulary (of which there are 50,000). This basic algorithm does not scale efficiently for large vocabularies or number of documents to classify.

**Question 2:**

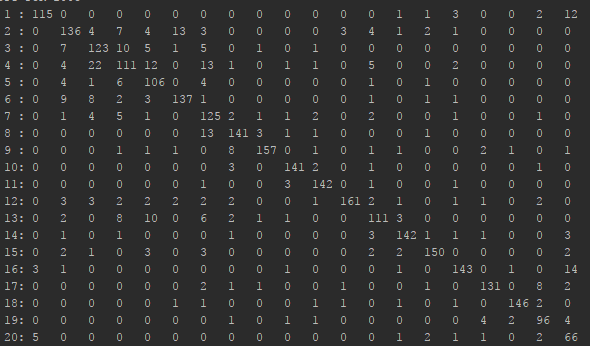
For the following tests, β is the default value of (1 / |V|), or in this case, roughly 1.6x10-5

|  |  |
| --- | --- |
| **Percentage of 12,000 Documents used for Training** | **Accuracy in Classifying Remaining Elements** |
| 10% | 68.611% |
| 25% | 76.788% |
| 50% | 83.083% |
| 75% | 86.0% |
| 100% | 98.875%  (tested on same docs used for Training) |

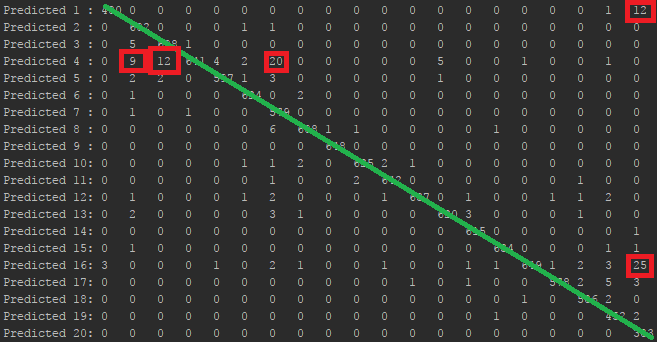
**25% Training, 75% Testing**



**75% Training, 25% Testing**



**100% Training, 100% Testing:**



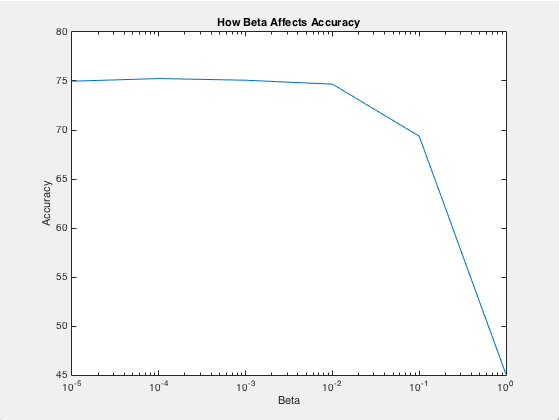
The numbers on the diagonal, overlaid in green, are those predictions Cij where i=j (meaning that the algorithm successfully categorized the document with its actual class).

**Question 3:**

As seen in the confusion matrix labelled above (for 100% Training, 100% Testing), we can see three separate instances of high inaccuracy when the algorithm predicts class 4, which is “comp.sys.ibm.pc.hardware”. The three instances of relatively high inaccuracy are when the actual inputs are of class “comp.graphics” (with 9 instances), “comp.os.ms-windows.misc” (with 12 instances), and most notably “misc.forsale” (with 20 instances). These misclassifications make sense, as the content of the documents are relatively similar: articles about “IBM PC hardware” likely have several words and phrases in common with computer graphics, Windows OS, and miscellaneous for sale listings or advertisements.

Additionally, we see two other significant instances where the algorithm misclassifies several documents; in Row 1, column 20, and Row 16, column 20. These correspond to the algorithm predicting articles about “alt.atheism” which are actually about “talk.religion.misc” (12 instances), and predicting “soc.religion.christian” that are actually about “talk.religion.misc” (25 instances) respectively.

**Question 4**



The above experiment of the effects that beta has on our classification of documents was done with training on 25% of our training.csv, and testing on the remaining 75%. As we can see from the above graph the impact of having a low beta (10-5), and a high beta (1) causes drops in accuracy. We found our best accuracy at around 10-4. When beta becomes small it starts to modify our data too much. This is because in our MLE equation the beta in the denominator multiplying against the number of unique words starts to take away that value, and has too large of an effect on the equation. When beta is 1 it no longer plays an effect on the equation, so it does not help us.

**Question 5**

In trying to come up with a ranking system for seeing how often the Naive Bayes algorithm relies on certain words, the obvious answer sprang to mind: entropy. In its essence, determining the entropy of all classes based on a single word at a time, we can see how “unstable” using that particular word for making a classification would be. For example, the most common word throughout all documents is “the”, unsurprisingly. When we calculate the entropy of this word across the classes we observe in our testing documents, “the” scores a whopping 4.9. This indicates that not only is “the” common in several, if not all, documents, but it also occurs frequently (almost 5 times per document) **within** these same documents. These are the sorts of words that Naive Bayes seeks to avoid: common (*within* documents), and widespread (*across* documents).

In a similar way, we also don’t want to rely heavily on words that occur rarely amongst documents, but also occur rarely **within** these same documents. Luckily, by modifying our entropy “filter” to only select those with high enough entropy to indicate that they are found in some documents, but not all, and not a single document, we can get a good selection of words. Namely, we aim to obtain words that occur more than roughly once in twenty documents (our total number of classes), but **also** have low entropy. Low entropy ensures the word is not shared by all classes, while also filtering out the rarest words (which have the lowest entropies). For example, “drporter” is found in no documents, so it would therefore have the lowest entropy, but needs to be filtered as it won’t help Naive Bayes in determining the likelihood of a specific class. By combining these methods, we guarantee obtaining words that are not spread amongst all classes (entropy), but also aren’t rare within the classes that they are found (filtering).

The process for ranking these elements is to 1. Sort the words by entropy in likelihood (MLE), but in *reverse order*. 2. Remove/ignore elements with entropy < 1/20 (removing typos and extremely rare words). 3. Select 100 elements from the front of the ordered set. The accuracy of this method was verified by viewing an explicit count of the total occurrences of given words, and ensuring that words being ranked highly were both occurring in very few classes (<4), and did not occur rarely throughout the dataset as a whole (<300).

**Question 6**

Ordered from 1-100 in descending rank (1 is higher ranking than 100).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1: mhz  2: mine  3: red  4: sale  5: states  6: christian  7: turn  8: certainly  9: goes  10: started  11: pay  12: consider  13: correct  14: deal  15: claim  16: server  17: via  18: os  19: package  20: haven | 21: hardware  22: night  23: netcom  24: sort  25: company  26: white  27: posting  28: sound  29: offer  30: advance  31: technology  32: cause  33: everyone  34: making  35: screen  36: top  37: programs  38: level  39: per  40: model | 41: center  42: cd  43: display  44: fine  45: similar  46: couple  47: fast  48: uiuc  49: important  50: difference  51: care  52: size  53: sell  54: although  55: uucp  56: oh  57: paul  58: wouldn  59: simply  60: national | 61: live  62: co  63: copy  64: contact  65: understand  66: later  67: went  68: ok  69: write  70: science  71: anybody  72: board  73: single  74: light  75: love  76: monitor  77: almost  78: nice  79: instead  80: mind | 81: anyway  82: area  83: including  84: unless  85: hi  86: agree  87: children  88: tried  89: open  90: black  91: side  92: whole  93: matter  94: saying  95: steve  96: color  97: often  98: guess  99: american  100: include |

**Question 7**

As we can see highlighted in the chart above, there are several words identified by our method that are inherently biased to the dataset at hand. While there are numerous words throughout this dataset that rank similarly (indicating that they are fairly unique, but not rare) and are what one might expect from an average article from a variety of sources, terms like “mhz”, “netcom”, or acronyms like “uucp” or “uiuc” are far from ordinary. Terms such as these -which are indicative of choices that the NB classifier would use to classify documents- are excellent illustrations of how such a powerful algorithm can be deeply influenced by the training data that is supplied to it. Given an even larger sampling of 200+ words, we have no doubts that similar instances of training/classification data bias would be identified.

Questions 1-7

**Question 1:**

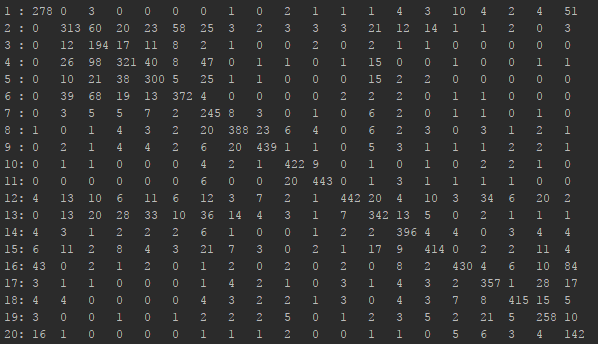
It would be hard to estimate the parameters due to the huge amount of them. We would be calculating 500001000 different parameters. This is because, for every individual document label prediction P(Y), we have to expand the terms where P(Y | Xi), where Xi is the word in the vocabulary (of which there are 50,000). This basic algorithm does not scale efficiently for large vocabularies or number of documents to classify.

**Question 2:**

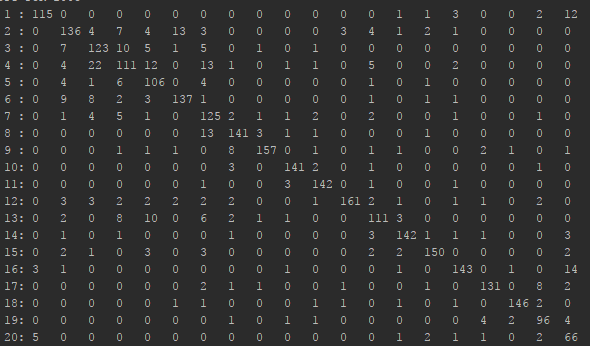
For the following tests, β is the default value of (1 / |V|), or in this case, roughly 1.6x10-5

|  |  |
| --- | --- |
| **Percentage of 12,000 Documents used for Training** | **Accuracy in Classifying Remaining Elements** |
| 10% | 68.611% |
| 25% | 76.788% |
| 50% | 83.083% |
| 75% | 86.0% |
| 100% | 98.875%  (tested on same docs used for Training) |

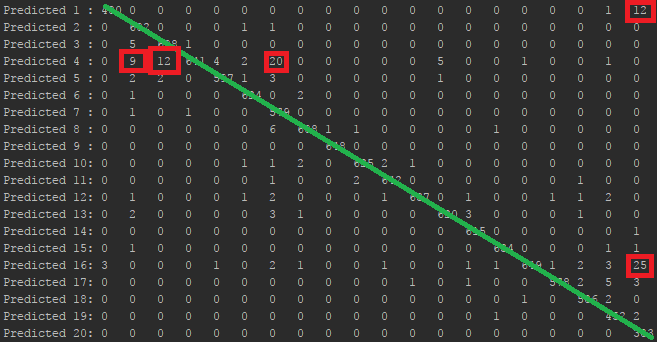
**25% Training, 75% Testing**



**75% Training, 25% Testing**



**100% Training, 100% Testing:**



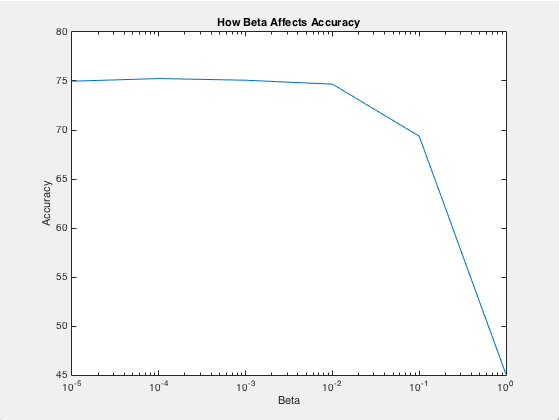
The numbers on the diagonal, overlaid in green, are those predictions Cij where i=j (meaning that the algorithm successfully categorized the document with its actual class).

**Question 3:**

As seen in the confusion matrix labelled above (for 100% Training, 100% Testing), we can see three separate instances of high inaccuracy when the algorithm predicts class 4, which is “comp.sys.ibm.pc.hardware”. The three instances of relatively high inaccuracy are when the actual inputs are of class “comp.graphics” (with 9 instances), “comp.os.ms-windows.misc” (with 12 instances), and most notably “misc.forsale” (with 20 instances). These misclassifications make sense, as the content of the documents are relatively similar: articles about “IBM PC hardware” likely have several words and phrases in common with computer graphics, Windows OS, and miscellaneous for sale listings or advertisements.

Additionally, we see two other significant instances where the algorithm misclassifies several documents; in Row 1, column 20, and Row 16, column 20. These correspond to the algorithm predicting articles about “alt.atheism” which are actually about “talk.religion.misc” (12 instances), and predicting “soc.religion.christian” that are actually about “talk.religion.misc” (25 instances) respectively.

**Question 4**



The above experiment of the effects that beta has on our classification of documents was done with training on 25% of our training.csv, and testing on the remaining 75%. As we can see from the above graph the impact of having a low beta (10-5), and a high beta (1) causes drops in accuracy. We found our best accuracy at around 10-4. When beta becomes small it starts to modify our data too much. This is because in our MLE equation the beta in the denominator multiplying against the number of unique words starts to take away that value, and has too large of an effect on the equation. When beta is 1 it no longer plays an effect on the equation, so it does not help us.

**Question 5**

In trying to come up with a ranking system for seeing how often the Naive Bayes algorithm relies on certain words, the obvious answer sprang to mind: entropy. In its essence, determining the entropy of all classes based on a single word at a time, we can see how “unstable” using that particular word for making a classification would be. For example, the most common word throughout all documents is “the”, unsurprisingly. When we calculate the entropy of this word across the classes we observe in our testing documents, “the” scores a whopping 4.9. This indicates that not only is “the” common in several, if not all, documents, but it also occurs frequently (almost 5 times per document) **within** these same documents. These are the sorts of words that Naive Bayes seeks to avoid: common (*within* documents), and widespread (*across* documents).

In a similar way, we also don’t want to rely heavily on words that occur rarely amongst documents, but also occur rarely **within** these same documents. Luckily, by modifying our entropy “filter” to only select those with high enough entropy to indicate that they are found in some documents, but not all, and not a single document, we can get a good selection of words. Namely, we aim to obtain words that occur more than roughly once in twenty documents (our total number of classes), but **also** have low entropy. Low entropy ensures the word is not shared by all classes, while also filtering out the most rare words (which have the lowest entropies). For example, “drporter” is found in no documents, so it would therefore have the lowest entropy, but needs to be filtered as it won’t help Naive Bayes in actually determining the likelihood of a specific class. By combining these methods, we guarantee obtaining words that are not spread amongst all classes (entropy), but also aren’t rare within the classes that they are found (filtering).

The process for ranking these elements is to 1. Sort the words by entropy in likelihood (MLE), but in *reverse order*. 2. Remove/ignore elements with entropy < 1/20 (removing typos and extremely rare words). 3. Select 100 elements from the front of the ordered set. The accuracy of this method was verified by viewing an explicit count of the total occurrences of given words, and ensuring that words being ranked highly were both occurring in very few classes (<4), and did not occur rarely throughout the dataset as a whole (<300).

**Question 6**

Ordered from 1-100 in descending rank (1 is higher ranking than 100).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1: mhz  2: mine  3: red  4: sale  5: states  6: christian  7: turn  8: certainly  9: goes  10: started  11: pay  12: consider  13: correct  14: deal  15: claim  16: server  17: via  18: os  19: package  20: haven | 21: hardware  22: night  23: netcom  24: sort  25: company  26: white  27: posting  28: sound  29: offer  30: advance  31: technology  32: cause  33: everyone  34: making  35: screen  36: top  37: programs  38: level  39: per  40: model | 41: center  42: cd  43: display  44: fine  45: similar  46: couple  47: fast  48: uiuc  49: important  50: difference  51: care  52: size  53: sell  54: although  55: uucp  56: oh  57: paul  58: wouldn  59: simply  60: national | 61: live  62: co  63: copy  64: contact  65: understand  66: later  67: went  68: ok  69: write  70: science  71: anybody  72: board  73: single  74: light  75: love  76: monitor  77: almost  78: nice  79: instead  80: mind | 81: anyway  82: area  83: including  84: unless  85: hi  86: agree  87: children  88: tried  89: open  90: black  91: side  92: whole  93: matter  94: saying  95: steve  96: color  97: often  98: guess  99: american  100: include |

**Question 7**

As we can see highlighted in the chart above, there are several words identified by our method that are inherently biased to the dataset at hand. While there are numerous words throughout this dataset that rank similarly (indicating that they are fairly unique, but not rare) and are what one might expect from an average article from a variety of sources, terms like “mhz”, “netcom”, or acronyms like “uucp” or “uiuc” are far from ordinary. Terms such as these -which are indicative of choices that the NB classifier would use to classify documents- are excellent illustrations of how such a powerful algorithm can be deeply influenced by the training data that is supplied to it. Given an even larger sampling of 200+ words, we have no doubts that similar instances of training/classification data bias would be identified.