# Naive Bayes Classifier

October 28, 2019

## 1 Naive Bayes Classifier (Ham or Spam Emails)

#### 1.0.1 Creating the dictionary of ham and spam

[1]:
```python
import glob
import math
import os

ALPHA = 0.05
V = 200000

"""
Counts the number of words in each email and returns a dictionary of word␣
 ↪counts as well as the total count of words.
"""
def count_words(directory):
    words = {}
    count = 0
    for filename in glob.glob(directory):
        f = open(filename)
        for line in f.readlines():
            line = line.strip("\n")
            if line not in words:
                words[line] = 1
            else:
                words[line] += 1
            count += 1
    return words, count


ham = "./data/ham/*"
spam = "./data/spam/*"
dict_ham, total_ham = count_words(ham)
dict_spam, total_spam = count_words(spam)
total_words = total_ham + total_spam
```

### 1.0.2 Training the Model

[2]:
```python
"""
Calculates the probabilities necessary to implement the Naive Bayes classifier.
Adds the P(word|class) to the dictionary of words and returns P(class) and␣
↪P(unseen_word)
"""
def calculate_probabilities(total_words, total_class, words, is_ham=False):
    for word in words:
        curr_count = words[word] + ALPHA
        log_prob = math.log(curr_count/(total_class + (V * ALPHA)))
        words[word] = log_prob

    prob_class = math.log(total_class/total_words)
    prob_unseen_class = ALPHA/(total_class +(V*ALPHA))
    prob_unseen_class = math.log(prob_unseen_class)

    return prob_class, prob_unseen_class


prob_ham, prob_unseen_ham = calculate_probabilities(total_words, total_ham,␣
↪dict_ham, True)
prob_spam, prob_unseen_spam = calculate_probabilities(total_words, total_spam,␣
↪dict_spam)
```

### 1.0.3 Populating the truth table (for confusion matrix)

[3]:
```python
"""
Extracts the names of the files that should be "Spam".
Returns a set with the files that should be classified as "Spam."
"""
def populate_truth(directory):
    truth = set()

    for filename in glob.glob(directory):
        f = open(filename)
        for line in f.readlines():
            line = line.strip("\n")
            truth.add(line)
    return truth

truth_file = "./data/truthfile*"
truth_table = populate_truth(truth_file)
```

### 1.0.4 Running the Model with Test Data

[4]:
```python
"""
Runs the naive bayes model by adding to the total probability of ham and spam,␣
 ↪respectively.
Returns a dictionary containing a mapping of the file name to the model␣
 ↪classification and the truth.
Also returns the accuracy of the model.

"""

def run_model(directory, spam, prob_spam, prob_unseen_spam, ham, prob_ham,
                prob_unseen_ham, truth_table):
    classification = {}
    differences = 0
    total_email = 0

    for email in glob.glob(directory):
        total_prob_ham = 0
        total_prob_spam = 0
        f = open(email)

        for word in f.readlines():
            word = word.strip("\n")
            if word not in ham:
                total_prob_ham += prob_unseen_ham
            else:
                total_prob_ham += ham[word]

            if word not in spam:
                total_prob_spam += prob_unseen_spam
            else:
                total_prob_spam += spam[word]

        email = email.split("/")
        email = email[len(email) - 1].strip(".words")

        if email in truth_table:
            truth = "Spam"
        else:
            truth = "Ham"

        if total_prob_ham > total_prob_spam:
            generated = "Ham"
            classification[email] = {"classification": generated, "truth":␣
 ↪truth}
        else:
                generated = "Spam"
                classification[email] = {"classification": generated, "truth":␣
 ↪truth}

            if generated != truth:
                differences += 1
            total_email += 1

        return classification, 1 - (differences/total_email)
```

[5]:
```
test = "./data/test/*"
results, accuracy = run_model(test, dict_spam, prob_spam, prob_unseen_spam,
                                dict_ham, prob_ham, prob_unseen_ham,␣
 ↪truth_table)

print("\nAccuracy: " + str(accuracy) + "\n")

print("\n Results: \n")

for key, val in results.items():
    print(key, val)
```

```
Accuracy: 0.86


 Results:

89 {'classification': 'Ham', 'truth': 'Ham'}
74 {'classification': 'Spam', 'truth': 'Ham'}
31 {'classification': 'Spam', 'truth': 'Spam'}
49 {'classification': 'Ham', 'truth': 'Ham'}
90 {'classification': 'Ham', 'truth': 'Ham'}
28 {'classification': 'Spam', 'truth': 'Spam'}
50 {'classification': 'Spam', 'truth': 'Spam'}
15 {'classification': 'Spam', 'truth': 'Spam'}
9 {'classification': 'Ham', 'truth': 'Ham'}
100 {'classification': 'Ham', 'truth': 'Ham'}
52 {'classification': 'Ham', 'truth': 'Ham'}
17 {'classification': 'Spam', 'truth': 'Spam'}
92 {'classification': 'Ham', 'truth': 'Ham'}
76 {'classification': 'Spam', 'truth': 'Ham'}
33 {'classification': 'Ham', 'truth': 'Ham'}
72 {'classification': 'Spam', 'truth': 'Ham'}
37 {'classification': 'Spam', 'truth': 'Spam'}
56 {'classification': 'Ham', 'truth': 'Ham'}
13 {'classification': 'Spam', 'truth': 'Spam'}
```

```
96 {'classification': 'Ham', 'truth': 'Ham'}
69 {'classification': 'Spam', 'truth': 'Ham'}
94 {'classification': 'Ham', 'truth': 'Ham'}
54 {'classification': 'Ham', 'truth': 'Ham'}
11 {'classification': 'Ham', 'truth': 'Ham'}
70 {'classification': 'Spam', 'truth': 'Ham'}
35 {'classification': 'Spam', 'truth': 'Spam'}
93 {'classification': 'Ham', 'truth': 'Ham'}
16 {'classification': 'Ham', 'truth': 'Ham'}
53 {'classification': 'Ham', 'truth': 'Ham'}
8 {'classification': 'Ham', 'truth': 'Ham'}
32 {'classification': 'Spam', 'truth': 'Spam'}
77 {'classification': 'Ham', 'truth': 'Ham'}
48 {'classification': 'Ham', 'truth': 'Ham'}
88 {'classification': 'Spam', 'truth': 'Spam'}
30 {'classification': 'Ham', 'truth': 'Ham'}
75 {'classification': 'Spam', 'truth': 'Ham'}
14 {'classification': 'Spam', 'truth': 'Spam'}
51 {'classification': 'Spam', 'truth': 'Spam'}
91 {'classification': 'Ham', 'truth': 'Ham'}
29 {'classification': 'Ham', 'truth': 'Ham'}
10 {'classification': 'Spam', 'truth': 'Spam'}
55 {'classification': 'Spam', 'truth': 'Spam'}
68 {'classification': 'Spam', 'truth': 'Spam'}
95 {'classification': 'Ham', 'truth': 'Ham'}
34 {'classification': 'Ham', 'truth': 'Ham'}
71 {'classification': 'Spam', 'truth': 'Ham'}
36 {'classification': 'Spam', 'truth': 'Spam'}
73 {'classification': 'Ham', 'truth': 'Spam'}
97 {'classification': 'Ham', 'truth': 'Ham'}
12 {'classification': 'Ham', 'truth': 'Ham'}
57 {'classification': 'Ham', 'truth': 'Ham'}
66 {'classification': 'Spam', 'truth': 'Spam'}
23 {'classification': 'Spam', 'truth': 'Spam'}
42 {'classification': 'Spam', 'truth': 'Spam'}
82 {'classification': 'Spam', 'truth': 'Ham'}
38 {'classification': 'Spam', 'truth': 'Spam'}
80 {'classification': 'Ham', 'truth': 'Ham'}
40 {'classification': 'Spam', 'truth': 'Spam'}
64 {'classification': 'Ham', 'truth': 'Ham'}
21 {'classification': 'Spam', 'truth': 'Spam'}
99 {'classification': 'Ham', 'truth': 'Ham'}
2 {'classification': 'Ham', 'truth': 'Ham'}
59 {'classification': 'Ham', 'truth': 'Ham'}
60 {'classification': 'Ham', 'truth': 'Ham'}
25 {'classification': 'Spam', 'truth': 'Spam'}
18 {'classification': 'Spam', 'truth': 'Spam'}
6 {'classification': 'Ham', 'truth': 'Ham'}
```

```
84 {'classification': 'Spam', 'truth': 'Ham'}
79 {'classification': 'Ham', 'truth': 'Ham'}
44 {'classification': 'Spam', 'truth': 'Spam'}
46 {'classification': 'Spam', 'truth': 'Spam'}
86 {'classification': 'Ham', 'truth': 'Ham'}
4 {'classification': 'Ham', 'truth': 'Ham'}
62 {'classification': 'Ham', 'truth': 'Ham'}
27 {'classification': 'Ham', 'truth': 'Ham'}
41 {'classification': 'Ham', 'truth': 'Ham'}
39 {'classification': 'Ham', 'truth': 'Ham'}
81 {'classification': 'Spam', 'truth': 'Ham'}
58 {'classification': 'Spam', 'truth': 'Spam'}
3 {'classification': 'Ham', 'truth': 'Ham'}
20 {'classification': 'Spam', 'truth': 'Spam'}
65 {'classification': 'Spam', 'truth': 'Spam'}
98 {'classification': 'Ham', 'truth': 'Ham'}
22 {'classification': 'Ham', 'truth': 'Ham'}
67 {'classification': 'Ham', 'truth': 'Ham'}
1 {'classification': 'Spam', 'truth': 'Spam'}
83 {'classification': 'Spam', 'truth': 'Ham'}
43 {'classification': 'Spam', 'truth': 'Spam'}
87 {'classification': 'Spam', 'truth': 'Ham'}
47 {'classification': 'Ham', 'truth': 'Ham'}
26 {'classification': 'Ham', 'truth': 'Ham'}
63 {'classification': 'Spam', 'truth': 'Spam'}
5 {'classification': 'Spam', 'truth': 'Spam'}
7 {'classification': 'Ham', 'truth': 'Ham'}
19 {'classification': 'Spam', 'truth': 'Spam'}
24 {'classification': 'Spam', 'truth': 'Spam'}
61 {'classification': 'Ham', 'truth': 'Ham'}
45 {'classification': 'Spam', 'truth': 'Spam'}
85 {'classification': 'Spam', 'truth': 'Ham'}
78 {'classification': 'Ham', 'truth': 'Ham'}
```

#### 1.0.5 Calculating Precision, Recall, F-Score

[6]:
```
"""
After the classifier is run, obtains the metrics of precision, recall, and␣
 ↪f-score as well as the confusion matrix.
Returns confusion matrix and metrics as two separate dictionaries.
"""
def get_metrics(results):
    "Assuming Ham is positive and Spam is negative"
    positive = "Ham"
    negative = "Spam"
    confusion_matrix = {"TP": 0, "TN": 0, "FP": 0, "FN": 0}
    for val in results.values():
        if val["truth"] == positive:
            if val["classification"] == positive:
                confusion_matrix["TP"] += 1
            else:
                confusion_matrix["FN"] += 1

        else:
            if val["classification"] == negative:
                confusion_matrix["TN"] += 1
            else:
                confusion_matrix["FP"] += 1
    metrics = {}

    metrics["precision"] = confusion_matrix["TP"] / (confusion_matrix["TP"] +␣
 ↪confusion_matrix["FP"])
    metrics["recall"] = confusion_matrix["TP"] / (confusion_matrix["TP"] +␣
 ↪confusion_matrix["FN"])

    metrics["f_score"] = \
        (2 * metrics["recall"] * metrics["precision"]) / (metrics["precision"]␣
 ↪+ metrics["recall"])

    return confusion_matrix, metrics


print(get_metrics(results))
```

```
({'TP': 50, 'TN': 36, 'FP': 1, 'FN': 13}, {'precision': 0.9803921568627451,
'recall': 0.7936507936507936, 'f_score': 0.8771929824561403})
```