CSC 411: Assignment #3

Xiangyu Kong **Yun Lu**
kongxi16 **luyun5**

March 19, 2018

Problem 1

The dataset contains headlines including the word "Trump". The quality of the dataset is generally good since there is a large variety of vocabularies included.

By observing, we can see that the fake news' headlines are generally longer than the real news' headlines.

Except for "Donald" and "Trump", the most frequent words are "to", "for", etc, but they do not mean a lot. The top meaningful words in total are "Clinton", "election" and "president". This infers that most news are regarding the 2017 presidential election and especially between Donald Trump and Hilary Clinton.

The statistics for the three words are generated through part1 in fake.py and the results are given below in Listing 1. We can see that although "Clinton" has the largest word count, most of them appear in the fake news. Reports that include "election" are more likely to be real news.

Listing 1: statistic results

```
[clinton]:
        real: 83
        fake: 132
        total: 215
[election]:
        real: 87
        fake: 74
        total: 161
[president]:
        real: 66
        fake: 64
        total: 130
```

Problem 2

The Naive Bayes algorithm is implemented in naive_bayes in util.py. To tune the parameters m and $\hat{p}$, we try using naive bayes on different value of m and $\hat{p}$ and pick one that has the best performance on the validation set. The range of test values for m was from 1 to 10, and for $\hat{p}$ was from 0.05 to 0.95. The returned optimum values were $m = 1$ and $\hat{p} = 0.25$.

To deal with small multiplication, a small_product function was implemented in util.py. It takes in a list of small values and uses the fact that $\prod_{i=1}^{N} a_i = \exp(\sum_{i=1}^{N} \log(a_i))$ to compute the value of $p(a_1, a_2, \ldots, a_n) = p(a_1)p(a_2)\ldots p(a_1)$

The performance on training set, validation set and test set is given below:

Listing 2: Performance

```
train performance = 0.964597902098
validation performance = 0.842535787321
test performance = 0.881390593047
```

Problem 3

1. The results are produced by part3 in fake.py and are listed below.

   $P(c|word) = \frac{P(word|c) \times P(c)}{P(words)}$ where
   $P(c) = \frac{count(c)}{count(total)}$, $P(word|c) = \frac{count(word\ in\ c)}{count(c)}$ and $P(word) = \frac{count(word\ in\ c)}{count(total)}$

   $P(c|not\ word)$ follows with similar calculations.

   The most important presence for predicting a class means $P(c|word)$ must be high and the most important absence for predicting a class means $P(c|not\ word)$ must be high.

2. After removing the stop words like "to", "us", "in", and etc, we get the results in b.

3. The stop words are very likely to appear no matter what class the headline is, so including them will not mean a lot.

Listing 3: top results

```
a:
Real:
top 10 important presence:
        ['trump', 'donald', 'to', 'us', 'trumps', 'in', 'on',
        'of', 'says', 'for']
top 10 important absence:
        ['kommonsentsjane', 'lord', 'tired', 'miller', '270',
         'elegant', 'battleground', 'fingers', 'salbuch', 'cult']
Fake:
top 10 important presence:
        ['trump', 'to', 'the', 'donald', 'in', 'of', 'for', 'a', 'and', 'on']
top 10 important absence:
        ['hanging', 'marching', 'regional', 'hearin', 'piling',
        'jennett', 'loathing', 'deferred', 'decry', 'lgbt']

b:
Real:
top 10 important presence:
        ['trump', 'donald', 'trumps', 'says', 'election',
        'clinton', 'north', 'korea', 'ban', 'president']
Fake:
top 10 important presence:
        ['trump', 'donald', 'hillary', 'clinton', 'election',
        'just', 'new', 'president', 'obama', 'america']
```

Problem 4

The logistic regression is implemented in logistic_regression.py. The parameters α and λ are selected using the similar method as in Problem 2. The two parameters were assigned to various values and the optimum value is the pair where the validation set performs the best. The final $\alpha = 0.0001$ and $\lambda = 0.001$.
Using this pair of parameters, the final logistic regression model gives the learning curve as in Fig.1.

Figure 1: caption

Problem 5

For Naive Bayes, $\theta_0 = P(c)$, and $\theta_i = P(w_i, c)$ and $I_i(x) = \frac{1}{P(w_i)}$. The calculated result is $\frac{P(c|w_i)}{P(w_i)}$ but w_i does not matter here so when the calculated value is greater than a threshold, we can say that the word combination is likely to produce a real or fake headline.

For Logistic Regression, θ_is are the weights in the network. I_is are the identity function that indicates whether the word w_i is present. If the calculated output is greater than a specific threshold, then the output will be real.

Problem 6

1. The results for the top positive and negative weight words are given in the list below (a). Comparing to the top important presence and absence word list in Problem 3a, the two lists have a few common words, but Problem 3a's list contains more stop words while the list given below has less stop words.

2. The list of words that do not include stop words are given in b. Comparing to the top important presence and absence word list in Problem 3b, the two lists have a few common words, but they appear to be in different order. This shows that the two methods weigh the same word differently.

3. When using logistic regression, if the features are not normalized, some features may have way more weight than other features. If the feature is very important, the magnitude of the feature will be very large thus affect the overall performance and cost.

   For this question, using the magnitude of the logistic regression parameters to indicate importance of a feature makes sense since we are performing binary classification here. The magnitude of the feature only decides whether the news is real or not. If there are more than two classes, using the magnitude of the parameters to indicate importance would be a mistake.

Listing 4: Top positive and negative words

```
a:
Positive:
['trumps', 'australia', 'us', 'turnbull', 'tax', 'korea', 'ban',
'donald', 'debate', 'administration']
Negative:
['hillary', 'victory', 'breaking', 'just', 'watch', 'information',
'u', 'are', 'd', 'soros']
b:
Positive:
['trumps', 'australia', 'turnbull', 'tax', 'korea', 'ban', 'donald',
'debate', 'administration', 'says']
Negative:
['hillary', 'victory', 'breaking', 'just', 'watch', 'information', 'u',
'd', 'soros', 'won']
```

Problem 7

1. The relationship between the maximum depth of the decision tree and its performance is shown in Fig2a. To generate the graph, the depth was chosen to be from range(1, 300, 30). As we can see, the graph looks similar to the learning rate graph in gradient descent.

   The best performing depth was chosen to be 151 because it has the highest validation performance.

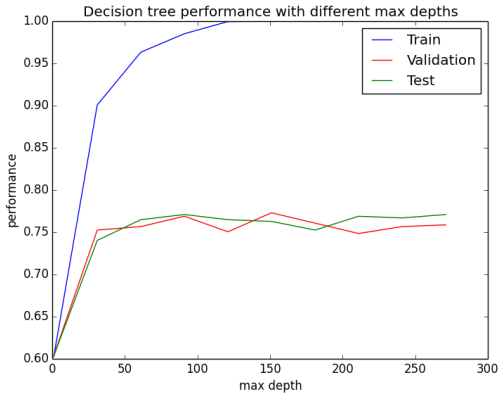
   Except for the max_depth, the criterion parameter was chosen to be 'entropy' while constructing the tree. This is to relate to the following questions.

2. The visualization of the best decision tree above is given in Fig2b.

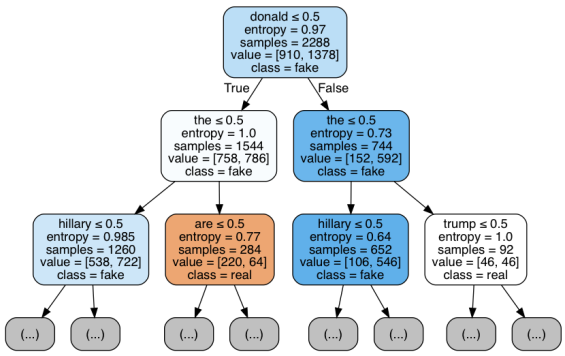
   We can see that all the words in the first two layer of the decision tree are in the list of important words in part 3.

3. The performance of three methods are given in the table below. We can see that Naive Bayes has the highest validation and test result. Decision Tree has the best training result (100%), but the lowest validation and test result. This is because it over-fits the data too much.

Method	Train	Validation	Test
Naive Bayes	96.46	84.25	88.14
Logistic Regression	97.90	84.05	84.66
Decision Tree	100	76.07	78.32



(a) Learning Curve



(b) Decision Tree

Problem 8

1. The first split of the tree is on the word x_i: "donald". Through the following calculations, we get that:

$$\begin{aligned}
H(Y) &= -P(Y = real)\log(P(Y = real)) + -P(Y = fake)\log(P(Y = fake)) = 0.967 \\
&= -\frac{910}{2288} \times \log(\frac{910}{2288}) - \frac{1378}{2288} \times \log(\frac{1378}{2288}) \\
&= 0.967 \\
H(Y|x_i) &= P(x_i = 0)H(Y|x_i = 0) + P(x_i = 1)H(Y|x_i = 1) \\
&= \frac{1544}{2288} \times 1 + \frac{744}{2288} \times 0.73 \\
&= 0.911 \\
I(Y; x_i) &= H(Y) - H(Y|x_i) = 0.056
\end{aligned}$$

2. The second split word x_j is chosen to "the". Through the following calculations, we get that:

$$\begin{aligned}
H(Y|x_j = 1) &= -(\frac{644}{1912} \times \log(\frac{644}{1912})) - (\frac{1268}{1912} \times \log(\frac{1268}{1912})) \\
&= 0.921 \\
H(Y|x_j = 0) &= -(\frac{266}{376} \times \log(\frac{266}{376})) - (\frac{110}{376} \times \log(\frac{110}{376})) \\
&= 0.871 \\
H(Y|x_j) &= \frac{1912}{2288} \times 0.921 + \frac{376}{2288} \times 0.871 \\
&= 0.913
\end{aligned}$$

   Then

$$I(Y, x_j) = 0.967 - 0.913 = 0.054$$

   This value is very close to the value we computed in 8(a), but it is still smaller. This is because these two words are very close and if this word has a larger information gain than x_i, this word should be the first word to split the tree.