CSC411: Project 2

Due on Sunday, February 18, 2018

Ying Yang

March 21, 2018

Part 1

Dataset description

Describe the datasets.
The dataset consists of healines of real news and headlines of fake news, each stored in a separate file. All headlines' punctuations and apostrophes are removed so that it only contain clean words.

We have a dataset of 3266 headlines
real news headlines: 1968
fake news headlines: 1298

3 examples of words that may be useful

We collected words that appear frequently in real and not frequently in fake news, and vice versa, since if a word appears frequently in a one and not frequent in the other, then the word is will strongly predict the headline. Among the results, we selected words that are meaningful (we didn't select words that are english grammar words, like "the", "and" , etc.)

examples of words:
word that appears frequent in real, and less frequent in fake: ['criticism', 10, 1]
words that appear frequent in fake, and less frequent in real: ['reporter', 15, 1] ['liberty', 12, 1]

Test scripts of the project:
For the following problems, test scripts are run simply by uncommenting out the part # in the *main*() function.

Listing 1: Test scripts

```
if __name__ == "__main__":
    # part1()
    # part2()
    # part3a()
    # part3b()
    # part4()
    # part5()
    # part6()
    # part7()
    # part8()
```

Part 2

Problem: Implement the Naive Bayes algorithm for predicting whether a headline is real or fake.

Implementation details: We used naive bayes probability to predict whether a headline is real or fake. For each of the headline, we used naive bayes to calculate P(real | headline) and P(fake | headline) and compare which one has greater probability. For example, if P(real | headline) > P(fake | headline), then we predict it to be real. We check if the prediction is correct using the corresponding label. We count the number of correct predictions for all headlines and divide it by the total number of headlines in each of training_set, validation_set and test_test to get the performance.

In particular, we used the formula:
P(real | headline) = likelihood_probability(headline) * prior_probability(real)
P(fake | headline) = likelihood_probability(headline) * prior_probability(fake)
= P(real | headline) = $\sum log(P(w_i|real))$+ log(prior_probability(real))
= P(fake | headline) = $\sum log(P(w_i|fake))$+ log(prior_probability(fake))

Step 1 : Calculate the likelihood of a headline.
(similar way for getting likelihood for fake news headline)

Step 1.1 We first calculate the total number of words that are in real and fake news in the dataset.

Step 1.2 We then maintained a table of real and fake words and its counts. (in python dictionary) For each word in the dataset, we store the word as key and P(word | real) and P(word | fake) as a list of value of the word.
P(word | c) = (count(word, c) + m * p_hat) / (count(c) + m) , c = real or fake
m and p_hat are the parameters that we will tune with validation set to get best performance.
We used validation set to tune the parameters m and p
Table of probability for each word (P(w | real), P(w | fake))

Listing 2: Make a table of probability for each word

```
probability_dict = {}
for word in probability_dict:
    # get real words probability
    if word in words_occurrences_real_table:
        real_word_likelihood = float(words_occurrences_real_table[word] + m * p_hat) /
         float(total_real_healines + m)
    else:
        real_word_likelihood = float(m * p_hat) / float(total_real_healines + m)
        probability_dict[word] = [real_word_likelihood]
    # get fake words probability
    if word in words_occurrences_fake_table:
        fake_word_likelihood = float(words_occurrences_fake_table[word] + m * p_hat) /
         float(total_fake_healines + m)
    else:
        fake_word_likelihood = float(m * p_hat) / float(total_fake_healines + m)
        probability_dict[word].append(fake_word_likelihood)
```

Step 1.3 Calculate likelihood of the healine. We do so by adding log(p(word | c)) if word appears in the

headline, log(1 - p(word | c)) if words doesn't appear in the headline, for each of the words in the table. This sum is the likelihood of the headline. We will called this likelihood_headline

Listing 3: Calculate likelihood for a headline

```
likelihood_headline_real = 0
likelihood_headline_fake = 0

for word in likelihood_table:
    if word in headline:
        likelihood_headline_real += math.log(likelihood_table[word][0])
        likelihood_headline_fake += math.log(likelihood_table[word][1])
    else:
        likelihood_headline_real += math.log(1 - likelihood_table[word][0])
        likelihood_headline_fake += math.log(1 - likelihood_table[word][1])
```

Step 2: Calculate prior probability of real or fake.
P(real) = total number of headlines real / total headlines P(fake) = total number of headlines fake / total headlines

Step 3: Calculate the posterior probability for a headline being real using likelihood_headline + log(prior(real)). We do similar operation for calculating posterior probability for a headline being fake.

Step 4: Count the accuracy.
We use the posterior probability for each headline being real or fake and compare which one is greater, and use the one that is greater to predict the whether a headline is real or fake. For example, if a headline's posterior probability for being real is greater than the posterior probability that of being fake, then we predict it is real. We iterate through all the headlines and check with the label of it. We increment correct guesses number if prediction is correct.

Listing 4: Calculate performance for a dataset

```
correct = 0
for i in range(len(dataset)):
    headline = dataset[i]
    if (predict_headline(likelihood_table, headline, dataset, dataset_label) ==
    "real" and dataset_label[i] == 1) or\
    (predict_headline(likelihood_table, headline, dataset, dataset_label) ==
     "fake" and dataset_label[i] == 0):
    correct += 1

return (float(correct) / len(dataset)) * 100
```

Step 5: We tune the parameters m and p_hat using validation set.
To do so, we tried with an array of numbers
m = [1.0, 0.5, 1e1,1e2,1e3,1e4] , p_hat = [0.9,3e-1,1e-1,3e-2,1e-2,3e-3]
After several tests and changing the values for p_hat and m, we found out that m=0.5 and p_hat = 0.1 are the best parameters.

Results:
training set performance: 97.8127734033%
validation set performance: 99.387755102%
test set performance: 98.0592441267%

Part 3

Part 3a

How we obtained the lists:
We have maintaine a table of probabilities for each words (P(w | real) and P(w | fake). For each of the word, we calculate store its posterior probability bydoing:

Presence predict real: P(real | w) = P(w | real) + log(prior_prob(real))
Presence predict fake: P(fake | w) = P(w | fake) + log(prior_prob(fake))
Absence predict real: P(real | w) = 1 - P(w | fake) + log(prior_prob(real))
Absence predict fake: P(fake | w) = 1 - P(w | fake) + log(prior_prob(fake))

Then we iterate through the table to find the top 10 highest values.

List the 10 words whose presence most strongly predicts that the news is real: ('travel')
('turnbull')
('ban')
('korea')
('north')
('says')
('us')
('trumps')
('trump')
('donald')

List the 10 words whose absence most strongly predicts that the news is real:
('if')
('are')
('you')
('clinton')
('and')
('just')
('is')
('a')
('hillary')
('the')

List the 10 words whose presence most strongly predicts that the news is fake:
('if')
('are')
('you')
('clinton')
('and')
('just')
('is')
('a')

('hillary')
('the')

List the 10 words whose absence most strongly predicts that the news is fake:
('travel')
('turnbull')
('ban')
('korea')
('north')
('says')
('us')
('trumps')
('trump')
('donald')

Compare the influence of presence vs absence of words on predicting whether the headline is real or fake news.

Part 3b

Write vectorized code that computes the gradient of the cost function with respect to the weights and biases of the network

10 non-stopwords that most strongly predict that the news is real: ('australia') ('wall') ('travel') ('turnbull') ('ban') ('korea') ('north') ('says') ('trumps') ('donald')
10 non-stopwords that most strongly predict that the news is fake: ('black') ('watch') ('win') ('obama') ('new') ('america') ('just') ('clinton') ('hillary') ('trump')

Part 3c

Why might it make sense to remove stop words when interpreting the model?
Because stop words usually doesn't have any meaning. They are only words that constructs syntactical sentences, but doesn't contribute to the meaning of the headline. Since we want to know which are the words that predict strongly, we need only consider words that make sense.

Why might it make sense to keep stop words?
It makes sense when sometimes a fake news might tend to use a set of non-stop english words to make up the title. For example, in real news, one might not use "seems", whereas in fake news one might use the word "seems" a lot when making up a title "that creates false statements", without proofs, such as "Trump seems to have intention to declare war". In this case, real news might not use this word "seems", since they tend to only use solid proofs, and we rarely see "seems" in titles with solid proofs.

Part 4

Input of the model is a 2D matrix, where row size is the number of headlines words, and column size is total headlines size. Output of the model is a n by 1 vector.
Learning rate: 0.1
Cost function: Cross-entropy
Regularization term: 3e-4,1e-3,3e-3,1e-2,3e-2,1e-1 were tested and 3e-3 was the optimal choice

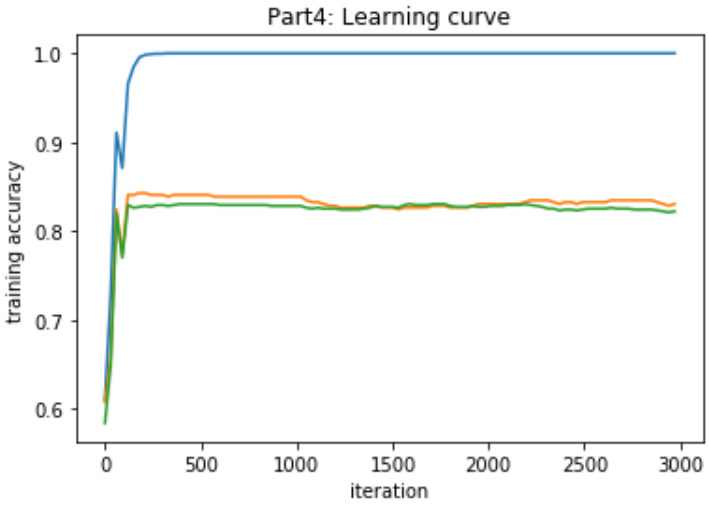

Figure 1: Learning curve

Accuracy of each set:
Training set accuracy 1.0
Validation set accuracy 0.830612244898
Test set accuracy 0.821246169561

The cost function of the model we use is:

$$-\sum_j (y_j log(\theta^T X_j) + (1 - y_j) log(1 - \theta^T X_j)) + \lambda \|\theta^2\|$$

As is stated, 3e-3 got the best performance.

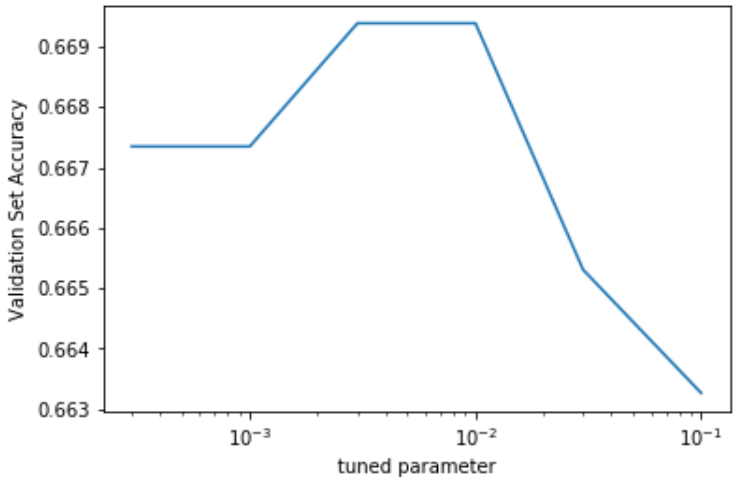

Figure 2: Validation set accuracy with respect to various regularization parameter

Part 5

Explain what are the θs and I's in the formula below

$\theta_0 + \theta_1 I_1(x) + \theta_2 I_2(x) + \theta_k I_k(x) >$ thr

k represents the total number of words that appear in the headline (only unique words in each of the headline)

Naive Bayes

$I_i(x) = 1$ if word appears in the headline, and 0 if it doesn't appear in the headline.
$\theta_i = \log \frac{P(x=1|y=real)}{P(x=1|y=fake)}$ - $\log \frac{P(x=0|y=real)}{P(x=0|y=fake)}$

Logistic regression

I is a k * 1 vector of 1 or 0s. $I_i(x) = 1$ if word appears in the headline, and 0 if it doesn't appear in the headline.
θ_i represent how important is a feature contributing to the final prediction of a headline being real or fake. It is the weights matrix.

¡¡¡¡¡¡¡ HEAD

Part 6

=======

Problem 9

Part 6 ***Part 6a)*** Top 10 positive θs:
trumps 9.95558870346

australia 6.85314372598

keating 6.57406540041

korea 6.35918006046

turnbull 5.95575911495

ban 5.41752108204

debate 5.30531752256

accept 5.30511003546

north 5.26097515049

tax 5.24804824411

Top 10 negative θs:
intensified -9.41612740596

indictment -8.01573155234

outsiders -7.31768911487

ww -6.84589755987

dismissing -6.41391572951

victoire -6.36644007845

totalement -6.26605371406

clipse -6.22355120767

dit -6.06041099787

livre -6.03790764251

compare with 3a:
Some common words between top 10 positive thetas and top 10 words whose presence most strongly predicts that the news is fake.

Some common words between top 10 negative thetas and top 10 words whose presence most strongly predicts that the news is real.

Similarity is because of common sense of definitions of two types of the ten words, and difference is because of assumption is probability-based, it can vary to some degree of instability.

Part 6b)

Top 10 positive theta without stop words:

trumps 9.95558870346

australia 6.85314372598

keating 6.57406540041

korea 6.35918006046

turnbull 5.95575911495

ban 5.41752108204

debate 5.30531752256

accept 5.30511003546

north 5.26097515049

tax 5.24804824411

Top 10 negative theta without stop words:

autistic -9.41612740596

breaking -8.01573155234

watch -7.31768911487

happened -6.84589755987

voter -6.41391572951

3 -6.26605371406

propaganda -6.22355120767

comment -6.06041099787

peacenik -6.03790764251

entire -5.91200525043

Compare with 3b:
Some common words between top 10 positive thetas and top 10 words whose presence most strongly predicts that the news is fake without stop words.
Some common words between top 10 negative thetas and top 10 words whose presence most strongly predicts that the news is real.
Part 6c) Without normalization, words with small magnitude would have more effects on output, or in other words, are more important than they are supposed to be in parameter numeric values. This would have bad influence because parameters value do not make that much sense any more as normalized ones because there are some bias produced by magnitude issues.
It is reasonable to use the magnitude in this problem because it is 0-1 classification problem.

¿¿¿¿¿¿¿ 2fcfde4e247687ab95fee9fe80aee69547f77eeb

Part 7

Part 8