Session 4 and 5 notebook

The notebook has been written during the session
please watch the video on "Course Materials" section of iLearn for the full description

April 23, 2020

1 Naive Bayes spam filtering

Consider that you are given a data set of text messages which are labeled with ham or spam. We will use a training sample with ~4000 text messages, but first let's consider a few examples to get familiar with the naive Bayes idea.

Class	Message	Bag of words
Spam	Send us your password	send, password
Ham	I will send you the letter	send, letter
Ham	I wrote a letter	write, letter

We want to compute P(Spam | Bag of words). Last session, we learned from Bayes' rule:

$$P(\text{Spam} \mid \text{Bag of words}) = \frac{P(\text{Bag of words} \mid \text{Spam})P(\text{Spam})}{P(\text{Bag of words} \mid \text{Spam})P(\text{Spam}) + P(\text{Bag of words} \mid \text{Ham})P(\text{Ham})}$$

P(word | spam) and P(word | ham) can be estimated from the training sample. To avoid zero probabilities, we consider the initial value of 1 for the number of occurence of a word. Note that the priors are P(ham)=$\frac{2}{3}$ and P(spam)=$\frac{1}{3}$.

Spam	Ham	word	Spam(i=1)	Ham(i=1)
$\frac{1}{2}$	$\frac{1}{4}$	send	$\frac{1+1}{2+4}$	$\frac{1+1}{4+4}$
$\frac{1}{2}$	$\frac{0}{4}$	password	$\frac{1+1}{2+4}$	$\frac{0}{2}$
$\frac{2}{4}$	letter	$\frac{0+1}{2+4}$	$\frac{2+1}{4+4}$	
$\frac{0}{2}$	$\frac{1}{4}$	write	$\frac{0+1}{2+4}$	$\frac{1+1}{4+4}$

Now, consider a new text message *"write your password in the password box"*. We don't have the word *"box"* in our training sample, so the safe choice would be to remove this from the bag of words and make decision based on on the other two words, *"write"* and *"password"*. *"password"* occured twice.

$$P(\text{spam} \mid \text{write,password,password})$$
$$= \frac{P(\text{write} \mid \text{spam})P(\text{password} \mid \text{spam})P(\text{password} \mid \text{spam})P(\text{spam})}{P(\text{write} \mid \text{Spam})P(\text{password} \mid \text{Spam})P(\text{password} \mid \text{spam})P(\text{Spam}) + P(\text{write} \mid \text{ham})P(\text{password} \mid \text{ham})P(\text{password} \mid \text{ham})P(\text{ham})}$$

$$P(\text{spam} \mid \text{write,password,password}) = \frac{\frac{1}{6} \times \frac{2}{6} \times \frac{2}{6} \times \frac{1}{3}}{\frac{1}{6} \times \frac{2}{6} \times \frac{2}{6} \times \frac{1}{3} + \frac{2}{8} \times \frac{1}{8} \times \frac{1}{8} \times \frac{2}{3}} \sim 70\%$$

and $P(\text{ham} \mid \text{write,password,password}) = 1 - P(\text{spam} \mid \text{write,password,password}) = 30\%$, so we classify this email as a spam message. This was just a demonsteration of the naive Bayes method. Let's use a large data set to build a model and evaluate its performance.

[122]:
```
import numpy as np
```

[123]:
```
import pandas as pd
from collections import Counter
```

NLTK (Natural Language Toolkit) is a set of libraries for Natural Language Processing (NLP)

[124]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /Users/nima/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[124]:
```
True
```

Stop words are the most common words in a language which don't carry much information. We will filter them before NLP

[125]:
```
stopwords=nltk.corpus.stopwords.words('english')
print(stopwords[:5])
```

```
['i', 'me', 'my', 'myself', 'we']
```

A word can have many variations with the same meaning. So, we will use stem package to normalize the words.

[126]:
```
from nltk.stem import PorterStemmer
Ps=PorterStemmer()
Ps.stem('cook'),Ps.stem('cooking'),Ps.stem('cooked')
```

[126]:
```
('cook', 'cook', 'cook')
```

We also need to remove punctuations, they are not informative in our classification.

[127]:
```
import string
punctuations=string.punctuation
print(punctuations)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Let's load the data:

[128]:
```
data=pd.read_csv('spam.csv')
data.head()
```

```
[128]:   Class                                               Text
       0   ham  Go until jurong point, crazy.. Available only ...
       1   ham                      Ok lar... Joking wif u oni...
       2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
       3   ham  U dun say so early hor... U c already then say...
       4   ham  Nah I don't think he goes to usf, he lives aro...
```

Change categorical data into numbers which can be processed in the code

```
[129]: data['Class_code']=pd.get_dummies(data.Class,drop_first=True)
       data.head()
```

```
[129]:   Class                                               Text  Class_code
       0   ham  Go until jurong point, crazy.. Available only ...           0
       1   ham                      Ok lar... Joking wif u oni...           0
       2  spam  Free entry in 2 a wkly comp to win FA Cup fina...           1
       3   ham  U dun say so early hor... U c already then say...           0
       4   ham  Nah I don't think he goes to usf, he lives aro...           0
```

```
[130]: def train_test_split(dataframe,test_size=0.3,rs=None):
           """A function which takes pandas dataframe and split it to train and test␣
        ↪samples"""
           dataframe_test=dataframe.sample(frac=test_size,random_state=rs)
           dataframe_train=dataframe.loc[dataframe.index.difference(dataframe_test.
        ↪index)]

           return (dataframe_train.reset_index(drop=True),dataframe_test.
        ↪reset_index(drop=True))
```

```
[131]: data_train,data_test=train_test_split(data,test_size=0.3,rs=4)
```

```
[132]: data_train.head()
```

```
[132]:   Class                                               Text  Class_code
       0   ham  Go until jurong point, crazy.. Available only ...           0
       1   ham                      Ok lar... Joking wif u oni...           0
       2   ham  U dun say so early hor... U c already then say...           0
       3  spam  FreeMsg Hey there darling it's been 3 week's n...           1
       4   ham  Even my brother is not like to speak with me. ...           0
```

```
[133]: data_test.head()
```

```
[133]:   Class                                               Text  Class_code
       0   ham                      No problem. Talk to you later           0
       1   ham  No idea, I guess we'll work that out an hour a...           0
       2   ham  Em, its olowoyey@ usc.edu have a great time in...           0
       3   ham                 I'm in a movie... Collect car oredi...           0
```

```
4   ham  Sorry man, accidentally left my phone on silen...          0
```

Cleaning up one of the text messages as an example:

[134]:
```
message=data_train.Text[46]
print(message)
```

```
Your gonna have to pick up a $1 burger for yourself on your way home. I can't
even move. Pain is killing me.
```

[135]:
```
#convert to lower case
message=message.lower()
print(message)
```

```
your gonna have to pick up a $1 burger for yourself on your way home. i can't
even move. pain is killing me.
```

[136]:
```
message=''.join([x for x in message if x not in punctuations])
print(message)
```

```
your gonna have to pick up a 1 burger for yourself on your way home i cant even
move pain is killing me
```

[137]:
```
message=[x for x in message.split() if x not in stopwords]
print(message)
```

```
['gonna', 'pick', '1', 'burger', 'way', 'home', 'cant', 'even', 'move', 'pain',
'killing']
```

[138]:
```
message=[Ps.stem(x) for x in message]
print(message)
```

```
['gonna', 'pick', '1', 'burger', 'way', 'home', 'cant', 'even', 'move', 'pain',
'kill']
```

[139]:
```
print(Counter(message))
```

```
Counter({'gonna': 1, 'pick': 1, '1': 1, 'burger': 1, 'way': 1, 'home': 1,
'cant': 1, 'even': 1, 'move': 1, 'pain': 1, 'kill': 1})
```

Now put them together in a function

[140]:
```
def clean_message(message):
    """a function to clean up message and return a dict with bag of their␣
 ↪occurence rate"""
    message=message.lower()
    message=''.join([x for x in message if x not in punctuations])
    message=[x for x in message.split() if x not in stopwords]
    message=[Ps.stem(x) for x in message]
    return(Counter(message))
```

[141]:
```
print(data_train.Text[80])
print(clean_message(data_train.Text[80]))
```

```
What is the plural of the noun research?
Counter({'plural': 1, 'noun': 1, 'research': 1})
```

Apply the function to all the data set

[142]:
```
data_train['bag_of_words']=data_train['Text'].apply(clean_message)
data_train.head()
```

[142]:
```
  Class                                               Text  Class_code  \
0   ham  Go until jurong point, crazy.. Available only ...           0
1   ham                      Ok lar... Joking wif u oni...           0
2   ham  U dun say so early hor... U c already then say...           0
3  spam  FreeMsg Hey there darling it's been 3 week's n...           1
4   ham  Even my brother is not like to speak with me. ...           0

                                        bag_of_words
0  {'go': 1, 'jurong': 1, 'point': 1, 'crazi': 1,...
1  {'ok': 1, 'lar': 1, 'joke': 1, 'wif': 1, 'u': ...
2  {'u': 2, 'dun': 1, 'say': 2, 'earli': 1, 'hor'...
3  {'freemsg': 1, 'hey': 1, 'darl': 1, '3': 1, 'w...
4  {'even': 1, 'brother': 1, 'like': 2, 'speak': ...
```

[143]:
```
bows=data_train.bag_of_words
```

[144]:
```
bows_ham=data_train[data_train.Class_code==0].bag_of_words
bows_spam=data_train[data_train.Class_code==1].bag_of_words
```

[145]:
```
words=list(set().union(*bows))
```

[146]:
```
number_of_occurence_ham={key:1 for key in words}
for word in words:
    for bow in bows_ham:
        if word in bow.keys():
            number_of_occurence_ham[word]+=bow[word]
```

[147]:
```
number_of_occurence_ham['soon']
```

[147]:
```
42
```

[148]:
```
number_of_occurence_spam={key:1 for key in words}
for word in words:
    for bow in bows_spam:
        if word in bow.keys():
                number_of_occurence_spam[word]+=bow[word]
```

[149]:

```
number_of_occurence_spam['free']
```

[149]:

```
143
```

Probability of a word given that the text is ham/spam

[150]:

```
P_word_h={}
P_word_s={}
for key in number_of_occurence_ham:
    P_word_h[key]=number_of_occurence_ham[key]/sum(number_of_occurence_ham.
 ↪values())
for key in number_of_occurence_spam:
    P_word_s[key]=number_of_occurence_spam[key]/sum(number_of_occurence_spam.
 ↪values())
```

Finding the priors

[151]:

```
P_h=bows_ham.size/bows.size
P_s=bows_spam.size/bows.size
```

[152]:

```
print(P_s)
print(P_h)
```

```
0.1310116086235489
0.8689883913764511
```

Define the main classifier function

[153]:

```
def classifier(document):
    document_bag_of_words=clean_message(document)
    P_document_h=1
    P_document_s=1
    for key in document_bag_of_words:
        if key in words:
            P_document_h=P_document_h*P_word_h[key]
            P_document_s=P_document_s*P_word_s[key]
    P_document_h=P_document_h*P_h
    P_document_s=P_document_s*P_s

    Pr_doc_h_normalized=P_document_h/(P_document_h+P_document_s)

    if Pr_doc_h_normalized>0.5:
        return 0
    else:
        return 1
classifier=np.vectorize(classifier)
```

[154]:

```
classifier('congratulations! you won $500')
```

[154]:

```
array(1)
```

[155]:

```
classifier("Let's apply this model to the test sample")
```

[155]:

```
array(0)
```

[156]:

```
prediction=classifier(data_test.Text.values)
```

[157]:

```
T=data_test.Class_code
```

[160]:

```
TP,TN,FP,FN=0,0,0,0
for i in range(len(T)):
    if T[i]==1:
        if prediction[i]==1:
            TP+=1
        if prediction[i]==0:
            FN+=1
    if T[i]==0:
        if prediction[i]==1:
            FP+=1
        if prediction[i]==0:
            TN+=1
```

Confusion matrix

[161]:

```
print(np.array([[TP,FP],[FN,TN]]))
```

```
[[ 158    9]
 [  21 1363]]
```

[164]:

```
precision=TP/(TP+FP)
print("precision=",precision)
```

```
precision= 0.9461077844311377
```

[165]:

```
recall=TP/(TP+FN)
print("recall=",recall)
```

```
recall= 0.88268156424581
```

[166]:

```
F1_score=2*precision*recall/(precision+recall)
print("F1_score=",F1_score)
```

```
F1_score= 0.9132947976878613
```

[167]:

```
accuracy=(TP+TN)/(TP+FP+FN+TN)
print("accuracy=",accuracy)
```

```
accuracy= 0.9806576402321083
```

[ ]:

```

```