STAT 391 Homework 1

Chongyi Xu
University of Washington
STAT 391 Spring 2018
chongyix@uw.edu

---

1. Problem 1- Practice with Probability

a Estimate $\theta = (\theta_0\ \theta_1\ \ldots\ \theta_4)$

```
observations = 0
counter = {0:0, 1:0, 2:0, 3:0, 4:0}
for line in open(r'C:\Users\johnn\Documents\UW\SchoolWorks
\\2018Spring\STAT391\HW1\hw2-little-amazon.dat').readlines():
    line = int(line.rstrip())
    counter[line] = counter[line] + 1
    observations = observations + 1

theta = [counter[0]/observations, counter[1]/observations, \
           counter[2]/observations, counter[3]/observations,
              counter[4]/observations]
print(theta)
```

As the result, I got my θ to be

$$\theta = (0.149, 0.396, 0.049, 0.255, 0.151)$$

And the sufficient statistics are the counts for each title, which is Table1

b A customer buys 3 books. What is the probability that he buys "War and Peace", "Harry Potter", "Probability" in this order? Assign the event that a customer buys the i^{th} book as E_i, then we are looking for

Table 1: Sufficient Statistics for Books

Book ID	Book Title	Count
0	War and Peace	149
1	Harry Potter & the Deathly Hallows	396
2	Winnie the Pooh	49
3	Get rich NOW	255
4	Probability	151

the probability that $P(E_0) \cdot P(E_1) \cdot P(E_4)$ since the book that every time that customer gets is an independent random

$$P(E_0) \cdot P(E_1) \cdot P(E_4) = 0.149 * 0.396 * 0.151 \approx 0.008910$$

And getting these three books has $2 * 3 = 6$ combinations, and we are only looking for one of those, thus

$$P = \binom{6}{1} \cdot (P(E_0) \cdot P(E_1) \cdot P(E_4)) \approx 0.001485$$

Therefore, the probability that the customer buys "War and Peace", "Harry Potter", "Probability" in this order is 0.001484934.

c A customer buys 4 books. What is the probability that she buys only non-fiction, that is, $N = 3, 4$? Denote the event that the customer buys 4 books and she buys only non-fiction as E. Then

$$P(E) = (P(E_3) + P(E_4))^4 = (0.255 + 0.151)^4 \approx 0.02717$$

d A customer buys 2 "Probability" books and 3 fiction (i.e 0 or 1 or 2) books. What is the probability of this event? Denote the event that the customer buys 2 "Probability" books and 3 fiction (i.e 0 or 1 or 2) books as E. Then

$$\begin{aligned} P(E) &= P(E_4)^2 * (P(E_0) + P(E_1) + P(E_2))^3 \\ &= 0.151^2 * (0.149 + 0.396 + 0.049)^3 \\ &\approx 0.004779 \end{aligned}$$

e A customer buys n books. What is the probability that he buys at least one "Probability"? Denote the event that the customer buys at least one "Probability" among n books he bought

$$P(E) = 1 - P(\neg E) = 1 - (1 - 0.151)^n$$

Table 2: Possible Results of Prediction

Outcome	Prediction	Result
H	H	1
H	T	0
T	H	0
T	T	1

2. Problem 2 Practice with probability

A man claims to have extrasensory perception. As a test, a fair coin is flipped 10 times, and the man is asked to predict the outcome in advance. He gets 7 out of 10 correct.

a Let Y refer to the number of correct tests, and denote the outcomes of the 10 individual tests with the random variables $X_1, X_2, \cdots X_10$.What are the distributions of each X_i? What is the relationship between $X_{1:10}$ and Y? For each random variable X_i, it has a Bernoulli distribution to be either 1 (for correct test) or 0 (for incorrect test). The relationship between X_i, $1 \leq i \leq 10$ and Y is

$$Y = \sum_{i=1}^{10} X_i$$

b What is the probability that he would have done at least this well if he had no ESP, i.e. if his guesses were essentially random? The event he would have done at least this well can be rewrite as $Y \geq 7$, denoting Y as the number of correct tests. And since Y is the sum of Bernoulli random variables, it will also be a Bernoulli random variable. From the result table above, we can conclude that the probability to get a head in a single trial is $p = 0.5$.

$$\begin{aligned} P(Y \geq 7) &= P(Y = 7) + P(Y = 8) + P(Y = 9) + P(Y = 10) \\ &= \binom{10}{7} p^7 (1-p)^3 + \binom{10}{8} p^8 (1-p)^2 \\ &+ \binom{10}{9} p^9 (1-p)^1 + \binom{10}{10} p^1 0 (1-p)^0 \\ &\approx 0.1719 \end{aligned}$$

c Suppose the test is changed - now, the coin is flipped until the man makes an incorrect guess. He guesses the first two correctly, but guesses the third wrong. What is the probability of this experimental outcome (again, assuming no ESP)? Denote the event that guessing first 2 correct and the third wrong to be E. Then,

$$P(E) = 0.5^2(1-0.5)^1 = 0.125$$

d Assume both tests were planned beforehand. What is the probability that both of these tests turned out the way they did? In other words, the plan was to first flip the coin 10 times and count how many times the man is correct (Y) then to continue flipping until the man makes the next mistake, at flip 10 + Z. You are asked the probability that Y = 7 and Z = 3.

$$\begin{aligned} P(\{Y=7\} \cup \{Z=3\}) &= P(Y=7) \cdot P(Z=3) \\ &= \binom{10}{7} p^7(1-p)^3 \cdot p^2(1-p)^1 \\ &\approx 0.01465 \end{aligned}$$

3. Problem 3

The basic procedure that I will do to the data and test sentences is the following.

- Use the given .dat files to get the probability of each letter for every languages.
- Eliminate all spaces and punctuation and turn all letters to lower case.
- Get the sufficient statistics for each test case.
- Compute the log-likelihood of each sentence in every language and print it out.
- Output the best guess.

According to the steps above, I have my code setup like the following.

```
import math

def langReader(file):
    '''
    langReader is used to read in files and compute for the
        probability
    for each letter

    Args:
        file: The name of the file containing the letter and
            probability.
    Returns:
        A dictionary containing letters as keys and probability as
            values
    '''
    pLang = {}
    for line in open(file):
        el = line.split(' ')
        letter = el[1].lower()
        pLang[letter] = float(el[2])/1000
    return pLang

def LetterCounter(testString):
    '''
    Counts for the number of each letter in the test string
        (sufficient statistics)

    Args:
        testString: The string to count
    Returns:
        A dictionary containing letters as keys and count number as
            values
    '''
    counter = {}
    for char in testString:
        try:
            counter[char] = counter[char] + 1
        except KeyError:
            counter[char] = 1
```

```
    return counter

def computeP(counter, probMap):
    '''
    Compute for the probability of with given letter counter and
        language probability map

    Args:
        counter: The letter counter of the test string.
        probMap: The probability map of the test language.
    Returns:
        A integer telling P(sentence)
    '''
    p = 1
    for letter in counter:
        try:
            p = p * probMap[letter]**counter[letter]
        except KeyError:
            print("The letter ", letter, " is not in this language")
    return p

def MaxLoglikelihood(testStr, langDict):
    '''
    Find the maximum log-likelihood according to the testStr and
        output the guess.

    Args:
        testStr: The string to test
        langDict: The dictionary for all languages.
    Output:
        The log-likelihood for each language and the best guess.
    '''
    print("Considering the sentence: ", testStr)
    testStr = ''.join(e for e in testStr if e.isalnum()).lower()
    counter = LetterCounter(testStr)
    best = [-float('inf'), '']
    for lang in langDict:
        ll = math.log(computeP(counter, langDict[lang]), 2)
        print("The log-likelihood for ", lang, " is ", ll)
        if best[0] < ll:
```

```
            best = [ll, lang]
    print("And as the result, the best guess is ", best[1], " with
        likelihood ", best[0], "\n")


# Initialize the probability map
path =
    r'C:\Users\johnn\Documents\UW\SchoolWorks\2018Spring\STAT391\HW1'

english = langReader(path + r'\english.dat')
french = langReader(path + r'\french.dat')
german = langReader(path + r'\german.dat')
spanish = langReader(path + r'\spanish.dat')

langs = {'English': english, 'French': french, 'German': german,
    'Spanish': spanish}
```

After finishing the setup, I put in the test sentences to test the maximum log likelihood.

```
# Case 1
MaxLogLikelihood("La verite vaut bien qu'on passe quelques annees
    sans la trouver.", langs)

# Case 2
MaxLogLikelihood("Chi po, non vo; chi vo, non po; chi sa, non fa;
    chi fa, non sa; \
e cosi, male il mondo va.", langs)

# Case 3:
MaxLogLikelihood("``Las cuentas, claras, y el chocolate, espeso''",
    langs)

# Case 4:
MaxLogLikelihood("`` Wir finden in den Buchern immer nur uns
    selbst. Komisch, \
    dass dann allemal die Freude gross ist und wir den Autor zum
        Genie erklaren.''", langs)
```

And the result I got is

```
Considering the sentence:  La verite vaut bien qu'on passe quelques
annees sans la trouver.
The log-likelihood for  English  is  -230.52920056694558
The log-likelihood for  French  is  -205.3554094970059
The log-likelihood for  German  is  -231.9774010063021
The log-likelihood for  Spanish  is  -213.83097995386774
And as the result, the best guess is  French  with likelihood
-205.3554094970059

Considering the sentence:  Chi po, non vo; chi vo, non po; chi sa,
non fa; chi fa, non sa; e cosi, male il mondo va.
The log-likelihood for  English  is  -247.7703137809151
The log-likelihood for  French  is  -258.7161923111603
The log-likelihood for  German  is  -265.64806883676596
The log-likelihood for  Spanish  is  -251.3309572579251
And as the result, the best guess is  English  with likelihood
-247.7703137809151

Considering the sentence:  ``Las cuentas, claras, y el chocolate,
espeso''
The log-likelihood for  English  is  -139.21322419404666
The log-likelihood for  French  is  -139.41869690833923
The log-likelihood for  German  is  -152.31264967462116
The log-likelihood for  Spanish  is  -133.70777498915248
And as the result, the best guess is  Spanish  with likelihood
-133.70777498915248

Considering the sentence:  `` Wir finden in den Buchern immer nur
uns selbst. Komisch,  dass dann allemal die Freude gross ist und
wir den Autor zum Genie erklaren.''
The log-likelihood for  English  is  -460.7764886241497
The log-likelihood for  French  is  -469.229214130381
The log-likelihood for  German  is  -437.88100125420345
The log-likelihood for  Spanish  is  -476.175646501842
And as the result, the best guess is  German  with likelihood
-437.88100125420345
```

which is pretty impressive since for each sentence, it successfully detect the language correctly and even for the Italian one, it gives me a result that is best guess among the four languages. As we can see from the result output, with longer sentence, the log-likelihood is more negative and generally, the difference of likelihood between the best guess and second best guess is larger. The probability model being used in this task tells that the language is produced by combining differnet letters and there are some letters that will be used much more often rather than some seldomly used ones.

4. Problem 4 - ML Estimation

a Estimate $\theta_a, \cdots, \theta_z$ the parameters of the Lincoln-English language model from the above texts (available in the files lincoln_text.txt).

Basically same setup as Problem 3, but this time, we need to build our own model, "lincol_English". And in the given lincoln_text.txt, there might be some missing letters (not all 26 letters of alphabet are present),s o we need to compare the result letters from the txt file and the letters in english.dat to see if there is any letter missing. If so, make the count of the letter to be 0.1 and add it to the total. By doing this, we are assuming that the missing letter will appear once every 10 times the same length of text, which could be a reasonable solution to the missing letters.

```
# Read in lincoln_text.txt
txt = ''
for line in open(path + r'\lincoln_text.txt'):
    txt = txt + line
txt = ''.join(e for e in txt if e.isalnum()).lower()
counter = LetterCounter(txt)
length = len(txt)

# Since there might be some missing letters
lincolnEng = {}
for letter in english:
    if letter not in counter:
```

```
        counter[letter] = 0.1
        length = length + 0.1
    lincolnEng[letter] = counter[letter] / length
print(lincolnEng)
langs['Lincoln-English'] = lincolnEng
```

And the θ found is the following

```
{'a': 0.07108239095315025, 'b': 0.015347334410339256,
'c': 0.020193861066235864, 'd': 0.029079159935379646,
'e': 0.11793214862681745, 'f': 0.02665589660743134,
'g': 0.021809369951534735, 'h': 0.05492730210016155,
'i': 0.07673667205169628, 'j': 0.0024232633279483036,
'k': 0.004038772213247173, 'l': 0.045234248788368334,
'm': 0.024232633279483037, 'n': 0.06704361873990307,
'o': 0.08239095315024232, 'p': 0.01615508885298869,
'q': 0.0008077544426494346, 'r': 0.08077544426494346,
's': 0.05977382875605816, 't': 0.0888529886914378,
'u': 0.03715670436187399, 'v': 0.01050080775444265,
'w': 0.018578352180936994, 'x': 8.076892011953801e-05,
'y': 0.027461432840642924, 'z': 0.0008076892011953801}
```

b Decide if the text

```
``Fourscore and seven years ago our fathers brought forth on
this continent a new nation,conceived in liberty and dedicated
to the proposition that all men are created equal.''
```

- Lincoln's Gettysburg Address on November 19, 1863.
is written in English, Spanish, German, French or Lincoln-English by the Maximum Likelihood method used in Problem 3.

Doing the same procedure as I did in Problem 3

```
MaxLogLikelihood("Fourscore and seven years ago our fathers
    brought forth on this \
continent a new nation, conceived in liberty and dedicated to
    the proposition that \
```

```
all men are created equal.", langs)
```

And the result is

```
Considering the sentence:  Fourscore and seven years ago our
fathers brought forth on this continent a new nation, conceived
in liberty and dedicated to the proposition that all men are
created equal.
The log-likelihood for  English  is  -582.2285420232441
The log-likelihood for  French  is  -610.0146289552811
The log-likelihood for  German  is  -615.2114218430331
The log-likelihood for  Spanish  is  -604.2956254104148
The log-likelihood for  Lincoln-English  is  -587.2208264009482
And as the result, the best guess is  English  with likelihood
-582.2285420232441
```