CIS519: Applied Machine Learning **Fall 2018**

Homework 4

Handed Out: Shruti Sinha *Due: 1 late day*

Note: Submitted by:Shruti Sinha(1 late day taken)
I collaborated with Hemanth and Andres.
Please scroll down to the report and theory section to look at my observations and solutions.

1. **Gradescope**: You must select the appropriate pages on Gradescope. Gradescope makes this easy for you: before you submit, it asks you to associate pages with the homework questions. Failing to do so will get you points off, which cannot be argued against after the fact. Gradescope may prompt you with a warning to select your cover page, please ignore this warning. Failure to submit to Gradescope will result in a 0.
2. **Canvas**: You must submit your homework to Canvas as well. Failure to do so will get you points off, which cannot be argued against after the fact.
3. **LaTeX**: Handwritten solutions or solutions not typeset in Latex will not be accepted.
4. **Solutions**: Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear.
5. **Outside Resources**: Finally, you are not allowed to use *any* material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class. If you're unsure if something violates our policy, please ask.

---

1 Document Classification [60 points]

In this problem, you will implement several text classification systems using the naive Bayes algorithm and semi-supervised Learning. In addition to the algorithmic aspects, you will also get a taste of data preprocessing and feature extraction needed to do text classification.

Please note that we are leaving some of the implementation details to you. In particular, you can decide how to represent the data internally, so that your implementation of the algorithm is as efficient as possible. **Note, however, that you are required to implement the algorithm yourself, and not use existing implementations, unless we specify it explicitly.** You should also **make your code as efficient as possible**. Sub-optimal code may fail tests.

1.1 Dataset

For the experiments, you will use the *20 newsgroups text* dataset[1] (you should download the .zip file provided on the course website). It consists of ∼18000 newsgroups posts on 20 topics. We have provided 2 splits of the data. In the first split, `20news-bydate`, we split the training and testing data by time, i.e. you will train a model on the data dated before a specified time and test that model on the data dated after that time. This split is realistic since in a real-world scenario, you will train your model on your current data and classify future incoming posts.

In the second split, `20news-random`, we have sampled the training/testing data uniformly at random.

1.2 Preprocessing [15 points]

In the first step, you need to preprocess all the documents and represent it in a form that can be used later by your classifier. We will use three ways to represent the data:

- Binary Bag of Words (B-BoW)
- Count Bag of Words (C-BoW)
- TF-IDF

We define these representations below. In each case, we define it as a matrix, where rows correspond to documents in the collection and columns correspond to words in the training data (details below).

However, since the vocabulary size is too large, it will not be feasible to store the whole matrix in memory. You should use the fact that this matrix is really sparse, and store the document representation as a dictionary mapping from word index to the appropriate value, as we define below. For example, `{'doc1': {'word1': val1, 'word2': val2,...},...}`

[1] `http://qwone.com/~jason/20Newsgroups/`

We would like you to do the preprocessing yourself, following the directions below. Do not use existing tokenization tools.

Note that we will be testing for the correctness of your code using space tokenization, so you must implement this. The provided function headers come with an `use_adv_tokenization` parameter, which when set to `False`, should indicate that your code is using simple space tokenization. You can also do advanced tokenization for further experimentation, but please only do advanced tokenization when the `use_adv_tokenization` parameter is set to `True`.

1.2.1 Binary Bag of Words (B-BoW) Model [5 points]

1. Extract a case-insensitive (that is, "Extract" will be represented as "extract") vocabulary set, $\mathcal{V}$, from the document collection $\mathcal{D}$ in the training data. Come up with a tokenization scheme - you can use simple space separation or more advanced Regex patterns to do this. You should lemmatize the tokens to extract the root word, and use a list of "stopwords" to ignore words like *the*, *a*, *an*, etc. **When reading files, make sure to include the `errors='ignore'` option**.

2. The set $\mathcal{V}$ of vocabulary extracted from the training data is now the set of features for your training. You will represent each document $d \in \mathcal{D}$ as a vector of all the tokens in $\mathcal{V}$ that appear in d. Specifically, you can think of representing the collection as a matrix $f[d, v]$, defined as follows:

$$\forall\ v \in \mathcal{V},\ \forall d \in \mathcal{D}, \qquad f[d, v] = \begin{cases} 1 & \text{if } v \in d \\ 0 & \text{else} \end{cases}$$

3. This should be a general function callable for any training data.

1.2.2 Count Bag of Words (C-BoW) Model [5 points]

1. The first part of vocabulary extraction is the same as above.

2. Instead of using just the binary presence, you will represent each document $d \in \mathcal{D}$ as a vector of all the tokens in $\mathcal{V}$ that appear in d, along with their counts. Specifically, you can think of representing the collection as a matrix $f[d, v]$, defined as follows:

$$f[d, v] = tf(d, v), \qquad \forall v \in \mathcal{V}, \qquad \forall d \in \mathcal{D},$$

   where, $tf(d, v)$ is the *Term-Frequency*, that is, number of times the word $v \in \mathcal{V}$ occurs in document d.

1.2.3 TF-IDF Model [5 points]

1. The first part of vocabulary extraction is the same as above.

2. Given the Document collection $\mathcal{D}$, calculate the Inverse Document Frequency (IDF) for each word in the vocabulary $\mathcal{V}$. The IDF of the word w is defined as the log (**use base 10**) of the multiplicative inverse of the fraction of documents in $\mathcal{D}$ that contain v. That is:
$$idf(v) = \log \frac{|\mathcal{D}|}{|\{d \in \mathcal{D}; v \in d\}|}$$

3. Similar to the representation above, you will represent each document $d \in \mathcal{D}$ as a vector of all the tokens in $\mathcal{V}$ that appear in d, along with their *tf idf* value. Specifically, you can think of representing the collection as a matrix $f[d, v]$, defined as follows:
$$f[d, v] = tf(d, v) \cdot idf(v, \mathcal{D}), \qquad \forall v \in \mathcal{V}, \qquad \forall d \in \mathcal{D},$$
where, $tf(.)$ is the *Term-Frequency*, and $idf(.)$ is the *Inverse Document-Frequency* as defined above.

1.3 Experiment 1 [10 points]

In this experiment, you will implement a simple (multiclass) Naive Bayes Classifier. That is, you will use the training documents to learn a model and then compute the most likely label among the 20 labels for a new document, using the Naive Bayes assumption. You will do it for all three document representations defined earlier. Please keep the labels as strings (the names of the subdirectories).

Note that, using Bayes rule, your prediction should be:
$$\hat{y}_d = \underset{y}{\text{argmax}} P(d|y) \cdot P(y),$$
where y ranges over the 20 candidate labels.

Since we are using the Naive Bayes model, that is, we assume the independence of the features (words in a document) given the label (document type), we get:
$$P(d|y) = \Pi_{v \in d} P(v|y)$$
where v is a word in document d and y is the label of document d.

The question is now how to estimate the coordinate-wise conditional probabilities $P(v|y)$. We have suggested to use three different representations of the document, but in all cases, estimate this conditional probability as:
$$P(v|y) = \frac{\sum_{\text{docs } d' \in \mathcal{D} \text{ of class } y} f[d', v]}{\sum_{w \in \mathcal{V}} \sum_{\text{docs } d' \in \mathcal{D} \text{ of class } y} f[d', w]}$$
Notice that in class we used the same estimation for the **B-BOW** model, and here we generalize it to the other two representations. Also recall that $\mathcal{D}$ refers to the training data, as defined above.

As discussed in class, estimating the probabilities needs to be smoothed. To do k**-laplace smoothing** (choose a suitable value of k), modify the above formula as follows: (You can think of 1-laplace smoothing as adding another document of class y which contains all the words in the vocabulary.)

$$P(v|y) = \frac{\sum_{\text{docs } d' \in \mathcal{D} \text{ of class } y} f[d', v] + k}{(\sum_{w \in \mathcal{V}} \sum_{\text{docs } d' \in \mathcal{D} \text{ of class } y} f[d', w]) + |\mathcal{V}|k}.$$

You will also need to estimate the prior probability of each label, as:

$$P(y) = \frac{\text{\# of documents in } \mathcal{D} \text{ with label } y}{|\mathcal{D}|}$$

In this experiment you will run the Naive Bayes algorithm for both datasets and all three the representations and provide an analysis of your results. Use *accuracy* to measure the performance.

Important Implementation Detail: Since we are multiplying probabilities, numerical underflow may occur while computing the resultant values. To avoid this, you should do your computation in **log space**. This means that you should take the log of the probability.

1.4 Experiment 2 [15 points]

In this experiment, you will use Semi-Supervised Learning to do document classification. The underlying algorithm you will use is Naive Bayes with the **B-BOW** representation, as defined in Experiment 1.

Follow the steps in Algorithm 1 below. You will split the training data into 2 parts, the Supervised (S) part, and the Unsupervised (U) part. You will use the S part as labeled and the U part as the unlabeled data. According to the Algorithm you will train on the labeled data S, predict on U, and then augment S with some of the (now labeled) examples from U.

You will use two separate criteria to determine how to choose the examples in U that will augment S in each round of the algorithm. You will also initialize the algorithm with three different sizes of S – 5%, 10% and 50% of the training data. Make sure that your algorithm always terminates.

You will use two options for `filter(.)` (line 10 of algorithm). In both cases, the decision will be made according to the probability Naive Bayes assigns to the winning label.

1. **Top-m:** Filter the top-m instances and augment them to the labeled data.
2. **Threshold:** Set a threshold. Augment those instances to the labeled data with confidence higher than this threshold. You have to be careful here to make sure that the program terminates. If the confidence is never higher than the threshold, then the classifier training will take forever to terminate. You can choose to terminate the classifier training if there are iterations where no confidence exceeded the threshold value.

Run the algorithm with both filtering techniques for both the datasets, and all three initializations, but only the representation **B-BOW**, and provide an analysis of your results. Use *accuracy* to measure the performance.

Algorithm Semi-Supervised Classifier

```
1: procedure SEMI-SUPERVISED CLASSIFIER
2:     $S_X, S_Y \leftarrow$ p% of Training data ($p = 5\%, 10\%, 50\%$ of the training data)
3:     $U_X \leftarrow$ X's of remaining 100-p% of Training data
4: loop:
5:     $model \leftarrow train\_naive\_bayes\_classifier(S_X, S_Y)$
6:     if $|U_X| = 0$ then
7:         return model
8:     end if
9:     $U_Y, P_Y \leftarrow predict(model, U_X)$
10:    $S_X^{new}, S_Y^{new} \leftarrow filter(U_X, U_Y, P_Y)$ (Choose a subset of the labeled U to augment S)
11:    $S_X \leftarrow S_X \cup S_X^{new}$
12:    $S_Y \leftarrow S_Y \cup S_Y^{new}$
13:    $U_X \leftarrow U_X \setminus S_X^{new}$
14:    goto loop
15: end procedure
```

1.5 (Optional: Extra Credit) Experiment 3 [5 points]

For experiment 2, initialize as suggested but, instead of the filtering, run EM. That is, for each data point in U, label it fractionally – label data point d with label l, that has weight $p(l|d)$. Then, add all the (weighted) examples to S. Now use Naive Bayes to learn again on the augmented data set (but now each data point has a weight! That is, when you compute $P(f[d, w]|y)$, rather than simply counting all the documents that have label y, now *all* the documents have "fractions" of the label y), and use the model to relabel it (again, fractionally, as defined above.) Iterate, and determine a stopping criteria.

1.6 What to Report: [20 points]

You have to submit the following items on **Canvas** and **Gradescope**:

1. `hw4.py` - well-commented code written in Python3.
2. `writeup.pdf` - For each of the above experiments, train it on both the datasets provided and report the accuracies achieved on the test data. Which test dataset achieved higher performance? Which representation model helped achieve a higher score? Which tokenization scheme(s) did you try for vocabulary extraction. How did they affect the performance? How did the stopwords removal affect your classifier performance? Are there any labels for which the classifier often make mistakes? Provide a detailed analysis of your results. For experiment 2, report the accuracy of your classifier on the test data for each iteration.
   SOLUTION:

   EXPERIMENT1

   The naive bayes classifier was first run for model of type B-BOW, with different values

of K, to determine the value of k which gave maximum training accuracy.
It was observed that with decreasing k, the test accuracy increased. Thus,k= 0.3 gave the best test accuracy for NEWS-BY-DATE data. k = 0.3 was chosen as the optimal smoothening parameter. The following test accuracies were observed for different k values on BY-DATE data.

k	test accuracy
0.3	59.61
0.5	57.12
1	52.58
2	44.96

Table 1: Different test accuracy for various k values for naïve Bayes

Next, naive bayes was tested on BY-DATE and RANDOM datasets for different representative models(B-BOW, C-BOW and TF-IDF) for k=0.3.
The following test accuracies were observed:

MODEL	TEST ACCURACY	LOW ACCURACY LABEL	ACCURACY(label)
B-BOW	59.61	comp.os.ms-windows.misc	08.12
C-BOW	58.08	comp.os.ms-windows.misc	12.69
TF-IDF	59.63	comp.os.ms-windows.misc	07.86
B-BOW(w/o stopW)	61.82	comp.os.ms-windows.misc	12.43
C-BOW(w/o stopW)	60.66	comp.os.ms-windows.misc	25.38
TF-IDF(w/o stopW)	60.31	comp.os.ms-windows.misc	07.86

Table 2: Testing accuracy for different models on BY-DATE Data

MODEL	TEST ACCURACY	LOW ACCURACY LABEL	ACCURACY(label)
B-BOW	67.03	comp.os.ms-windows.misc	32.99
C-BOW	65.71	talk.religion.misc	35.05
TF-IDF	66.78	talk.religion.misc	33.46

Table 3: Testing accuracy for different models on RANDOM Data

OBSERVATIONS: 1)It is observed that without stopwords, the test accuracy increases for each model representation.This is because, now irrelevant words are not included in the training dictionary.The inclusion of which would not be a true representation of the uniqueness of a document in terms of words in each document. For B-BOW we have 59.61 and B-BOW(w/o stopwords) we get 61.82
2)Among the three, B-BOW(67.03) performs only slightly better than TF-IDF(66.78) for

given value of k =0.3 on random data.
3)Simple space tokenization was only employed and the words were turned into their lowercase form, in order to identify same words with different cases as one. Converting into lowercase does improve the performance, as more words are identified correctly.
4)The random dataset performed better than by-date dataset. This is because for bydate, the model trained on previous data may not be able to capture the characteristics of the future data very well. Thus random dataset allows the model to be familiar with variance across samples,like different mix of words the model might encounter.
5)The classifier often makes mistakes as shown in the table above for the misc. documents: comp.os.ms-windows.misc and talk.religion.misc.The folders are consistently misclassified and receive the lowest accuracies.

EXPERIMENT2

The semi supervised model was tested on By-date and random dataset with two filter functions. For filter by threshold: I set threshold as -200.
For random dataset across different iterations the accuracies are as follows:
For $p = 0.50$ we get: 59.9, 54.6, 45.37
For $p = 0.10$ we get: 39.17, 39.16, 21.90
For $p = 0.05$ we get: 35.14, 34.9, 15.34
For by-date dataset:
For $p = 0.50$ we get: 51.99, 48.23, 39.16
For $p = 0.10$ we get: 32.21, 30.08, 16.44
For $p = 0.05$ we get: 29.61, 27.03, 12.99
For the filter by mtop, I run my experiments for m = 1500,on by date and random dataset:
For m = 1500
For by-date dataset across different iterations the accuracies are as follows:
For $p = 0.50$ we get:52.76, 47.88, 35.90, 24.15
For $p = 0.10$ we get:51.11, 49.16, 46.77, 43.89, 39.45, 30.29, 21.23
For $p = 0.05$ we get:45.98, 45.63, 43.97, 38.55, 34.41, 27.03, 19.54, 11.7
For random dataset:m=1500
For $p = 0.50$ we get:60.66, 54.91, 38.19, 29.16
For $p = 0.10$ we get:59.51, 56.79, 54.77, 52.24, 45.55, 37.99, 27.07
For $p = 0.05$ we get:51.17, 51.18, 46.69, 41.40, 37.34, 31.85, 27.76, 15.30

OBSERVATIONS
1)It is observed that with every iteration, testing accuracy decreases as mislabeled documents are added to the supervised set. Thus with every iteration the accuracy decreases.
2)Moreover, the random data set has comparatively higher accuracies than bydate dataset.
3)By setting a high threshold, more proportion of documents are disregarded by the filter function:filter by threshold, thus requiring less number of iterations.
4)For filter by mtop, more the value of m, lesser iterations are required for semi supervised model, as for every iteration a fixed large number of unsupervised documents are added to the supervised set.
5)The greater the p value, lesser the computational time.

6)Also, a higher p value,here 50, has a higher range of test accuracy(60-29 for random), while a lower value has more iterations but lesser range of accuracy(51.17-15.30) as observed for random dataset.

2 Theory [40 points]

2.1 Multivariate Exponential naïve Bayes [30 points]

In this question, we consider the problem of classifying piazza posts (Y) into two categories: student posts (A), and instructor posts (B). For every post, we have two attributes: number of words (X_1), and number of mathematical symbols (X_2). We assume that each attribute (X_i, $i = 1, 2$) is related to a post category (A/B) via an Exponential distribution[2] with a particular mean ($\lambda^{-1}_{A;i}/\lambda^{-1}_{B;i}$). That is

$$Pr[X_i = x|Y = A] = e^{-x\lambda_{A;i}}\lambda_{A;i} \quad \text{and} \quad Pr[X_i = x|Y = B] = e^{-x\lambda_{B;i}}\lambda_{B;i} \text{ for } i = 1, 2$$

X_1	X_2	Y
0	3	A
4	8	A
2	4	A
6	2	B
3	5	B
2	1	B
5	4	B

Table 4: Dataset for Exponential naïve Bayes

Assume that the given data in Table 4 is generated by a Exponential naïve Bayes model. You will use this data to develop a naïve Bayes predictor over the Exponential distribution.

$\Pr(Y=A) = \frac{3}{7}$	$\Pr(Y=B) = \frac{4}{7}$
$\lambda_{A;1} = \frac{1}{2}$	$\lambda_{B;1} = \frac{1}{4}$
$\lambda_{A;2} = \frac{1}{5}$	$\lambda_{B;2} = \frac{1}{3}$

Table 5: Parameters for Exponential naïve Bayes

1. **[10 points]** Compute the prior probabilities and parameter values, i.e., fill out Table 5. Please show all the intermediate steps and results. [Hint: Use MLE to compute the λ's].
   Solution:Under the Naive Bayes independence assumption,conditioned on Y=A,X1 is generated independent of X2. Moreover, the probability of X1 given Y=A/B, follows

[2] http://en.wikipedia.org/wiki/Exponential_distribution

an Exponential distribution(similar for X2).We use this the independence assumption to compute MLE for λ's. To compute $\lambda_{A;i}$ we only look at those values of X1/X2 where Y=A. To compute MLE, we take the log likelihood of the data(say for n examples):

$$L(x, \lambda) = \sum_{k=1}^{n} \log Pr[X_i = x_k | Y = A]$$

We now find the maximum log likelihood:
$\frac{\partial L(x,\lambda)}{\partial \lambda} = 0$

$$\frac{\partial (n \times \log \lambda - \lambda \sum_{k=1}^{n} x_k)}{\partial \lambda} = 0$$

$$\frac{n}{\lambda} - \sum_{k=1}^{n} x_k = 0$$

$$\lambda = \frac{n}{\sum_{k=1}^{n} x_k}$$

Thus, we use the formula for λ to calculate $\lambda_{A;1}$,$\lambda_{A;2}$,$\lambda_{B;1}$, and$\lambda_{B;2}$.
For $\lambda_{A;1}$ we look when Y=A, the values of X1 are 0,4,2. The formula thus yields $\lambda_{A;1} = \frac{3}{0+4+2} = \frac{1}{2}$. Similarly other values are calculated.
Also, since the dataset contains 3A and 4B labels so the prior probabilities are:
$\Pr(Y{=}A) = \frac{3}{7}$ and $\Pr(Y{=}B) = \frac{4}{7}$
The parameter and probability values are filled up accordingly in Table2.

2. **[10 points]** Based on the parameter values from Table 5, compute

$$\frac{\Pr(X_1{=}2, X_2{=}3 \mid Y{=}A)}{\Pr(X_1{=}2, X_2{=}3 \mid Y{=}B)}$$

Solution:We know by the naive bayes independence assumption:

$$\frac{\Pr(X_1{=}x_1, X_2{=}x_2 \mid Y{=}A)}{\Pr(X_1{=}x_1, X_2{=}x_2 \mid Y{=}B)} = \frac{\Pr(X_1{=}x_1 \mid Y{=}A) \Pr(X_2{=}x_2 \mid Y{=}A)}{\Pr(X_1{=}x_1 \mid Y{=}B) \Pr(X_2{=}x_2 \mid Y{=}B)}$$

$$\frac{\Pr(X_1{=}x_1 \mid Y{=}A) \Pr(X_2{=}x_2 \mid Y{=}A)}{\Pr(X_1{=}x_1 \mid Y{=}B) \Pr(X_2{=}x_2 \mid Y{=}B)} = \frac{e^{-x_1 \lambda_{A;1}} \lambda_{A;1} \times e^{-x_2 \lambda_{A;2}} \lambda_{A;2}}{e^{-x_1 \lambda_{B;1}} \lambda_{B;1} \times e^{-x_2 \lambda_{B;2}} \lambda_{B;2}}$$

Plugging in the values of x1 and x2.

$$\frac{e^{-x_1 \lambda_{A;1}} \lambda_{A;1} \times e^{-x_2 \lambda_{A;2}} \lambda_{A;2}}{e^{-x_1 \lambda_{B;1}} \lambda_{B;1} \times e^{-x_2 \lambda_{B;2}} \lambda_{B;2}} = \frac{e^{-2 \times \frac{1}{2}} \times \frac{1}{2} \times e^{-3 \times \frac{1}{5}} \times \frac{1}{5}}{e^{-2 \times \frac{1}{4}} \times \frac{1}{4} \times e^{-3 \times \frac{1}{3}} \times \frac{1}{3}} = \frac{6}{5} e^{\frac{-1}{10}} = 1.085$$

3. **[5 points]** Derive an algebraic expression for the Exponential naïve Bayes predictor for Y in terms of the parameters estimated from the data.
Given $X_1 = x_1$ and $X_2 = x_2$, we predict A if:

$$\frac{\Pr(Y{=}A \mid X_1{=}x_1, X_2{=}x_2)}{\Pr(Y{=}B \mid X_1{=}x_1, X_2{=}x_2)} > 1$$

Else, we predict B.

$$\frac{\Pr(X_1 = x_1, X_2 = x_2 \mid Y = A)\Pr(Y = A)}{\Pr(X_1 = x_1, X_2 = x_2 \mid Y = B)\Pr(Y = B)} = \frac{\Pr(X_1 = x_1 \mid Y = A)\Pr(X_2 = x_2 \mid Y = A)\Pr(Y = A)}{\Pr(X_1 = x_1 \mid Y = B)\Pr(X_2 = x_2 \mid Y = B)\Pr(Y = B)} > 1$$

$$\frac{e^{-x_1\lambda_{A;1}}\lambda_{A;1} \times e^{-x_2\lambda_{A;2}}\lambda_{A;2}\Pr(Y = A)}{e^{-x_1\lambda_{B;1}}\lambda_{B;1} \times e^{-x_2\lambda_{B;2}}\lambda_{B;2}\Pr(Y = B)} > 1$$

In cases where we have more than two classes (say Y=A,B,C,D), we could also do the argmax for each label and determine the final label.

4. [**5 points**] Use the parameters estimated from the data given in Table 4 to create a Exponential naïve Bayes classifier. What will the classifier predict as the value of Y, given the data point: $X_1 = 2, X_2 = 3$?
Solution: We plug in the values into the above formula to get:

$$\frac{e^{-x_1\lambda_{A;1}}\lambda_{A;1} \times e^{-x_2\lambda_{A;2}}\lambda_{A;2}\Pr(Y = A)}{e^{-x_1\lambda_{B;1}}\lambda_{B;1} \times e^{-x_2\lambda_{B;2}}\lambda_{B;2}\Pr(Y = B)} = \frac{e^{-2\times\frac{1}{2}} \times \frac{1}{2} \times e^{-3\times\frac{1}{5}} \times \frac{1}{5} \times \frac{3}{7}}{e^{-2\times\frac{1}{4}} \times \frac{1}{4} \times e^{-3\times\frac{1}{3}} \times \frac{1}{3} \times \frac{4}{7}} = 0.814 < 1$$

Since $0.814 < 1$ we predict B label.

2.2 Coin Toss [10 points]

Consider the following way to generate a series of Heads and Tails. For each element in the series, first a coin is tossed. If it comes up as a T, it is shown to the user. On the other hand, if the coin toss comes up as an H, then the coin is tossed the second time, and the outcome of this toss is shown to the user.
Assume that the probability of a coin toss coming up as an H is p (and hence, the probability of it coming up as a T is $1 - p$). Suppose you see a sequence $TTHTHHTHTT$ generated based on the scheme given above. What is the most likely value of p?
(Assume a Bernoulli model to compute probability of the coin toss sequence.)

Solution: According to the given sequence of generating Heads and Tails, the probability of getting Heads is p^2.
This is because, H can be shown to the user only if it comes up in the second toss (H for the first toss and H for the second toss too). The probability of getting a tail is $(1 - p)$ in the first toss and $p(1 - p)$ in the second toss(first toss H and second toss T,HT).
Thus the probability of getting a Tails is $(1 - p) + p(1 - p) = 1 - p^2$. To get the most likely value of p we do the following:
Assuming Bernoulli distribution, if we have a sequence S of length n which contains k Heads,

$$\Pr(S) = (p^2)^k(1 - p^2)^{n-k}$$

Taking log,

$$L = \log \Pr(S) = 2k \log p + (n-k)\log(1-p^2)$$

We take the derivative

$$\frac{\partial L}{\partial p} = \frac{2k}{p} + \frac{(n-k)(-2p)}{1-p^2} = 0$$

Therefore,

$$2k = 2np^2$$

$$p = \sqrt{\frac{k}{n}}$$

For the given sequence, we have n as 10(sequence of heads and tails) and k, no of times heads appears is 4. Therefore we get p,

$$p = \sqrt{\frac{4}{10}} = 0.632$$