# HOMEWORK 4

$AKASHSHARMA$
9081731771

**Instructions:** Although this is a programming homework, you only need to hand in a pdf answer file. There is no need to submit the latex source or any code. You can choose any programming language, as long as you implement the algorithm from scratch.
Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Please check Piazza for updates about the homework.

## 1 Best Prediction Under 0-1 Loss (14 pts)

Suppose the world generates a single observation $x \sim$ multinomial$(\theta)$, where the parameter vector $\theta = (\theta_1, \dots, \theta_k)$ with $\theta_i \geq 0$ and $\sum_{i=1}^k \theta_i = 1$. Note $x \in \{1, \dots, k\}$. You know $\theta$ and want to predict $x$. Call your prediction $\hat{x}$. What is your expected 0-1 loss:

$$\mathbb{E}\mathbb{1}[\hat{x} \neq x]$$

using the following two prediction strategies respectively? Prove your answer.
Strategy 1: $\hat{x} \in \arg\max_x \theta_x$, the outcome with the highest probability.
$x_j \in \arg\max_x \theta_j$
When $\hat{x} = x_j$, there is no Loss. Here, $\mathbb{E}0[\hat{x} = x_j]$ = $0(\theta_j)$.
When $\hat{x} \neq x_j$, the probabaility is $1 - \theta_j$. 1 loss, that is, $\mathbb{E}1[\hat{x} \neq x_j]$ = $1(1 - \theta_j)$.
0-1 loss is given by,

$$\mathbb{E}\mathbb{1}[\hat{x} \neq x] = 0(\theta_j) + 1(1 - \theta_j)$$

Hence, $\mathbb{E}\mathbb{1}[\hat{x} \neq x] = 1 - \theta_j$
Strategy 2: You mimic the world by generating a prediction $\hat{x} \sim$ multinomial$(\theta)$. (Hint: your randomness and the world's randomness are independent)
When $\hat{x} = x$ there is no loss, while when they are unequal the loss is 1. Let the maximum likelihood estimate for $x_j$ be $\theta_j$. The estimate will be same for testing and training data, and as both the randomness are independent, $p(\hat{x} = x)$ is equal to $\sum_{i=1}^k \theta_j^2$.
The loss in this case will be $0(\sum_{i=1}^k \theta_j^2) = 0$.
When $\hat{x} \neq x$, $p(\hat{x} \neq x)$ will be $1 - \sum_{i=1}^k \theta_j^2$ and the loss will be $1(1 - \sum_{i=1}^k \theta_j^2)$.
0-1 loss is:

$$\mathbb{E}\mathbb{1}[\hat{x} \neq x] = 1 - \sum_{i=1}^k \theta_j^2$$

## 2 Best Prediction Under Different Misclassification Losses (12 pts)

Like in the previous question, the world generates a single observation $x \sim$ multinomial$(\theta)$. Let $c_{ij} \geq 0$ denote the loss you incur, if $x = i$ but you predict $\hat{x} = j$, for $i, j \in \{1, \dots, k\}$. $c_{ii} = 0$ for all $i$. This is a way to generalize different costs on false positives vs false negatives from binary classification to multi-class classification. You want to minimize your expected loss:

$$\mathbb{E}c_{x\hat{x}}.$$

Derive your optimal prediction $\hat{x}$.

Let $p_j$ = probability of getting $(x = j)$
$\theta_i$ = probability of getting $(x = i)$

Loss $\mathbb{E}c_{x\hat{x}} = \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij}\theta_i p_j$ Let, $\sum_{i=1}^{k} c_{ij}\theta_i$ be equal to $t_j$

Minimising the above expected loss. Let the value be j = n, where $\mathbb{E}c_{x\hat{x}}$ is minimum.
$\mathbb{E}c_{x\hat{x}} = \sum_{j \neq n} t_j p_j + t_n p_n$. Putting $p_n = 1 - \sum_{j \neq n} p_j$ in the above
$\mathbb{E}c_{x\hat{x}} = t_n + \sum_{j \neq n} p_j(t_j - t_n)$.
We can minimise the above expression, by setting $p_j = 1$ for $j = n$ and $p_j = 0$ for all $j \neq n$
Therefore, minimum $\mathbb{E}c_{x\hat{x}} = \sum_{i=1}^{k} c_{in}\theta_i$.
So, the optimal value of $\hat{x} = n\epsilon \arg\min_j \sum_{i=1}^{k} c_{ij}\theta_i$

## 3 Language Identification with Naive Bayes (8 pts each)

Implement a character-based Naive Bayes classifier that classifies a document as English, Spanish, or Japanese - all written with the 26 lower case characters and space.
The dataset is languageID.tgz, unpack it. This dataset consists of 60 documents in English, Spanish and Japanese. The correct class label is the first character of the filename: $y \in \{e, j, s\}$.
We will be using a character-based multinomial Naïve Bayes model. You need to view each document as a bag of characters, including space. We have made sure that there are only 27 different types of printable characters (a to z, and space) – there may be additional control characters such as new-line, please ignore those. Your vocabulary will be these 27 character types. (Note: not word types!)

1. Use files 0.txt to 9.txt in each language as the training data. Estimate the prior probabilities $\hat{p}(y = e)$, $\hat{p}(y = j)$, $\hat{p}(y = s)$ using add-1 smoothing. Give the formula for add-1 smoothing in this case.
   The Prior probabilities are below:
   $\hat{p}(y = e) = 10 + 1/30 + 3 = 1/3$
   $\hat{p}(y = j) = 10 + 1/30 + 3 = 1/3$
   $\hat{p}(y = s) = 10 + 1/30 + 3 = 1/3$
   The formula for add-1 smoothing in our case is: $\hat{p}(y = c) = \frac{n_c + \alpha}{n + c_t\alpha}$ where $\alpha$ is 1 for add-1 smoothing. Here, $n_c$ is the number of documents with label c, n is the total number of documents, and $c_t$ is the total number of classes.

   Print the prior probabilities. (Hint: Store all probabilities here and below in $\log()$ internally to avoid underflow. This also means you need to do arithmetic in log-space. But answer questions with probability, not log probability.)

2. Using the same training data, estimate the class conditional probability (multinomial parameter) for English

   $$\theta_{c,e} := \hat{p}(c \mid y = e)$$

   for $c \in \{a, \ldots, z, space\}$. Again use add-1 smoothing. Give the formula for add-1 smoothing in this case. Print $\theta_e$ which is a vector with 27 elements.

   The value of $\theta_e$ is below:
   $(0.060147893833355344, 0.011158061534339852, 0.021523834675821998, 0.02198600290505744,$
   $0.10530833223293279, 0.018948897398653311, 0.017496368678198863, 0.047207183414762974,$
   $0.05539416347550508, 0.0014525287204542453, 0.0037633698666631454, 0.028984550376336987,$
   $0.020533474184603197, 0.057903076719926054, 0.06443945596197016, 0.01677010431797174,$
   $0.0005942162947312822, 0.053809586689554996, 0.06615608081341608, 0.08008715172322725,$
   $0.026673709230159777, 0.009309388617456754, 0.015515647695761256, 0.0011884325894625644,$
   $0.01386504687706325, 0.0006602403274792025, 0.17912320084510763)$

   Here, to do the 1-smoothing, we will add 1 in the numerator, and 27 in the denominator (total number of unique characters). The formula for add one smoothing is: $\theta_{c,e} := \frac{n_{c,e} + \alpha}{n + c_t\alpha}$, where $\alpha$ is 1, $c_t$ is the total number of characters, and $n_{c,e}$ is the frequency of character c across all the documents with label e.

3. Print $\theta_j, \theta_s$.
   The value of $\theta_j$ is:
   (0.1316763247922921, 0.010891572994484396, 0.005515604272847867, 0.017244990574600293,
   0.060182922572086 85, 0.0039097954339174755, 0.01403337289673951, 0.03176708790057948 5,
   0.09697689031627452, 0.002373804370592753, 0.05739021154786009, 0.001466173287719053 3,

0.039796132095231446, 0.056692033791803396, 0.09111219716539831, 0.0009076310828736997,
0.00013963555121133841, 0.042798296446275225, 0.0421699364658242, 0.05697130489422607,
0.07058577113733157, 0.00027927110242267683, 0.019758430496404383, 6.981777560566921e-05,
0.014173008447950849, 0.007749773092229281, 0.12336800949521748)

The value of $\theta_s$ is:
(0.10450428245732947, 0.008256824203586173, 0.03752541746256701, 0.0397436687411424,
0.11374699611806026, 0.008626532750015405, 0.007209316655370016, 0.004559738739293857,
0.04984903567687473, 0.006654753835726169, 0.000308090455357693, 0.05292994023045166,
0.025817980158974674, 0.05416230205188243, 0.0724628751001294, 0.02427752788218621,
0.007702261383942325, 0.059276603610820135, 0.0657465031733317, 0.035615256639349314,
0.03370509581613162, 0.005915336742867706, 0.00012323618214307722, 0.002526341733933083,
0.007887115657156942, 0.0027111960071476986, 0.16815577053422884)

4. Treat e10.txt as a test document $x$. Represent $x$ as a bag-of-words count vector (Hint: the vocabulary has size 27). Print the bag-of-words vector $x$.
The bag-of-words count vector is shown below:
('a': 164, 'b': 32, 'c': 53, 'd': 57, 'e': 311, 'f': 55, 'g': 51, 'h': 140, 'i': 140, 'j': 3, 'k': 6, 'l': 85, 'm': 64, 'n': 139, 'o': 182, 'p': 53, 'q': 3, 'r': 141, 's': 186, 't': 225, 'u': 65, 'v': 31, 'w': 47, 'x': 4, 'y': 38, 'z': 2, ' ': 498)

5. Compute $\hat{p}(x \mid y)$ for $y = e, j, s$ under the multinomial model assumption, respectively. Use the formula

$$\hat{p}(x \mid y) = \prod_{i=1}^{d} \theta_{i,y}^{x_i}$$

where $x = (x_1, \ldots, x_d)$. Show the three values: $\hat{p}(x \mid y = e), \hat{p}(x \mid y = j), \hat{p}(x \mid y = s)$. Hint: you may notice that we omitted the multinomial coefficient. This is ok for classification because it is a constant w.r.t. $y$.
The values are below:
$\hat{p}(x \mid y = e) = 10^{-3405.644556038407}$
$\hat{p}(x \mid y = s) = 10^{-3670.8233803708918}$
$\hat{p}(x \mid y = j) = 10^{-3804.2107164507584}$

6. Use Bayes rule and your estimated prior and likelihood, compute the posterior $\hat{p}(y \mid x)$. Show the three values: $\hat{p}(y = e \mid x), \hat{p}(y = j \mid x), \hat{p}(y = s \mid x)$. Show the predicted class label of $x$.
In this case, the value of probability $\hat{p}(y = e \mid x)$ gives the maximum value, among the $\hat{p}(y = j \mid x)$ and $\hat{p}(y = s \mid x)$. So, the value predicted is e.
$\hat{p}(y = e \mid x) = \hat{p}(x \mid y = e) * p(y = e)/p(x)$
$p(y = e \mid x) = \frac{10^{-3406.1216772931266}}{10^{-3406.1216772931266} + 10^{-3671.300501625612} + 10^{-3804.6878377054786}} \approx 1$
$p(y = s \mid x) = \frac{10^{-3671.300501625612}}{10^{-3406.1216772931266} + 10^{-3671.300501625612} + 10^{-3804.6878377054786}} \approx 0$
$p(y = j \mid x) = \frac{10^{-3804.6878377054786}}{10^{-3406.1216772931266} + 10^{-3671.300501625612} + 10^{-3804.6878377054786}} \approx 0$

7. Evaluate the performance of your classifier on the test set (files 10.txt to 19.txt in three languages). Present the performance using a confusion matrix. A confusion matrix summarizes the types of errors your classifier makes, as shown in the table below. The columns are the true language a document is in, and the rows are the classified outcome of that document. The cells are the number of test documents in that situation. For example, the cell with row = English and column = Spanish contains the number of test documents that are really Spanish, but misclassified as English by your classifier.

| | English | Spanish | Japanese |
|---|---|---|---|
| English | | | |
| Spanish | | | |
| Japanese | | | |

The classifier gives 100 percent accuracy in this testset. So, the values of the confusion matrix below shows that the documents that were English were classified as English, and same goes for Spanish and Japanese. So, there are no cases of error (that is where a wrong language class is predicted).

| | English | Spanish | Japanese |
|---|---|---|---|
| English | 10 | 0 | 0 |
| Spanish | 0 | 10 | 0 |
| Japanese | 0 | 0 | 10 |

8. Take a test document. Arbitrarily shuffle the order of its characters so that the words (and spaces) are scrambled beyond human recognition. How does this shuffling affect your Naive Bayes classifier's prediction on this document? Explain the key mathematical step in the Naive Bayes model that justifies your answer.

$$\hat{p}(x \mid y) = \prod_{i=1}^{d} \theta_{i,y}^{x_i}$$

where $x = (x_1, \ldots, x_d)$

When we shuffle the characters, so that the words, and spaces are scrambled, it will not change the frequency of each of the characters. Our Naive Bayesian classifier is built on the character frequency, which will be unchanged. Computing $\theta_{i,y}$ in the above formula, only depends on the frequency and total number of characters, it will not change the prediction and hence the answer remains the same.

## 4 Weka (10 pts)

Perform multinomial Naive Bayes classification. Suggested key steps:

- We want you to experience Weka's TextDirectoryLoader. For this, you need to prepare our documents such that each character becomes a word. The easiest way is to insert a space between characters, but turn original space into the word "space". For example, the document "is the sun dying" should be turned into "i s space t h e space s u n space d y i n g". Then, create 3 subdirectories e, j, s and place each of the 20 documents in that language into the corresponding subdirectory.
- Find out how to ask Weka to use TextDirectoryLoader to load those 60 documents and use the subdirectory name as the class name. This happens in the Preprocess tab in Weka Explorer.
- Apply Filter: filters/unsupervised/Attribute/StringtoWordVec. Click to see options, set outputWordCounts to True (otherwise Weka uses binary absence/presence features, which is less interesting for our purpose). You should see @@class@@ and 27 features.
- Choose Edit... A big document by feature table will show up, where you should see word counts roughly in the range 1–100. Right click on @@class@@, select "Attribute as class". You may save it as an arff file. The tri-color histogram for different features is interesting.
- From the Classify tab, choose Classifier / bayes / NaiveBayesMultinomial
- Let Weka perform 10-fold cross validation. Report the resulting confusion matrix.

Here, the Weka accuracy is also 100 percent. All the English documents are predicted as English only, and same for Spanish, and Japanese. The confusion matrix is shown below.

| a | b | c | classified as |
|---|---|---|---|
| 20 | 0 | 0 | a = e |
| 0 | 20 | 0 | b = j |
| 0 | 0 | 20 | c = s |