HOMEWORK 4

>>Martin Diges<<
>>9080689699<<

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Late submissions may not be accepted. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. You can choose any programming language (i.e. python, R, or MATLAB). Please check Piazza for updates about the homework.

https://github.com/missingnoglitch0/cs760/tree/main/hw4

1 Best Prediction Under 0-1 Loss (10 pts)

Suppose the world generates a single observation $x \sim \text{multinomial}(\theta)$, 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 θ 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.

SOLUTION
From the definition of expected value, we have that

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

For a single draw from a multinomial disrtribution, $P(x = j) = \theta_j$. Thus, if we choose the $\hat{x}$ which satisfies Strategy 1,

$$\mathbb{E}[\mathbb{1}_{\hat{x} \neq x}] = 1 - \theta_{\hat{x}}$$

Strategy 2: You mimic the world by generating a prediction $\hat{x} \sim \text{multinomial}(\theta)$. (Hint: your randomness and the world's randomness are independent)

SOLUTION
Because our randomness and the world's randomness are independent, the probability that both variables will be observed with some value j is:

$$P(x = j \wedge \hat{x} = j) = P(x = j) * P(\hat{x} = j)$$

Thus, we have that

$$P(x = \hat{x}) = \sum_{j=1}^{k} P(x = j \wedge \hat{x} = j) = \sum_{j=1}^{k} P(x = j) * P(\hat{x} = j) = \sum_{j=1}^{k} \theta_j^2$$

From Strategy 1, we have that $\mathbb{E}[\mathbb{1}_{\hat{x} \neq x}] = P(x \neq \hat{x}) = 1 - P(x = \hat{x})$. Thus,

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

2 Best Prediction Under Different Misclassification Losses (6 pts)

Like in the previous question, the world generates a single observation $x \sim \text{multinomial}(\theta)$. 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 $\hat{x}$.

SOLUTION
From the definition of expected value and the knowledge that our randomness and that of the world are independent, we have that

$$\mathbb{E}[c_{x\hat{x}}] = \sum_{i=1}^{k}\sum_{j=1}^{k} c_{ij} * P(x = i \wedge \hat{x} = j) = \sum_{i=1}^{k}\sum_{j=1}^{k} c_{ij} * P(x = i)P(\hat{x} = j)$$

We can extract the probability with respect to i from the inner summation to obtain

$$\sum_{i=1}^{k} P(x = i) \sum_{j=1}^{k} c_{ij} * P(\hat{x} = j)$$

Given that we are making one optimal prediction $\hat{x}$, we can fix some $\bar{x} = \hat{x}$ which will be our prediction. Thus, $P(\hat{x} = \bar{x}) = 1$, $P(\hat{x} \neq \bar{x}) = 0$. We can update our expression above to get

$$\sum_{i=1}^{k} P(x = i) \left(c_{i\bar{x}} * P(\hat{x} = \bar{x}) + \sum_{j \in \{1,\dots,k\}/\{\bar{x}\}} c_{ij} * P(\hat{x} = j) \right)$$

$$\sum_{i=1}^{k} P(x = i) \left(c_{i\bar{x}} * 1 + \sum_{j \in \{1,\dots,k\}/\{\bar{x}\}} c_{ij} * 0 \right)$$

$$\sum_{i=1}^{k} P(x = i)\,(c_{i\bar{x}})$$

Thus, we will choose an $\bar{x} = argmin_{\bar{x}} \sum_{i=1}^{k} P(x = i)\,(c_{i\bar{x}})$.
Expressed verbally, we choose the $\bar{x}$ which has the lowest sum of $c_{i\bar{x}}$ over the possible values of i, weighting the values of i by their probability of being generated.

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: $y \in \{e, j, s\}$. (Note: here each file is a document in corresponding language, and it is regarded as one data.)
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 additive smoothing with parameter $\frac{1}{2}$. Give the formula for additive smoothing with parameter $\frac{1}{2}$ in this case. Print and include in final report 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.)

   SOLUTION

   Additive smoothing: $\hat{p}(y = l) = \frac{b_l+\alpha}{\sum_k b_k+\alpha K}$, where b_l is the count for observations of class l, α is our smoothing parameter, and K is the number of classes (in this case, the number of languages). Once we make the necessary substitutions, we have

   $$\theta_l := \hat{p}(y = l) = \frac{b_l + 0.5}{\sum_k b_k + 0.5 * 3}$$

   After collecting observations and applying smoothing, we have $\hat{p}(y = e) = \hat{p}(y = j) = \hat{p}(y = s) = \frac{1}{3}$

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

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

   where c_i is the i-th character. That is, $c_1 = a, \dots, c_{26} = z, c_{27} = space$. Again use additive smoothing with parameter $\frac{1}{2}$. Give the formula for additive smoothing with parameter $\frac{1}{2}$ in this case. Print θ_e and include in final report which is a vector with 27 elements.

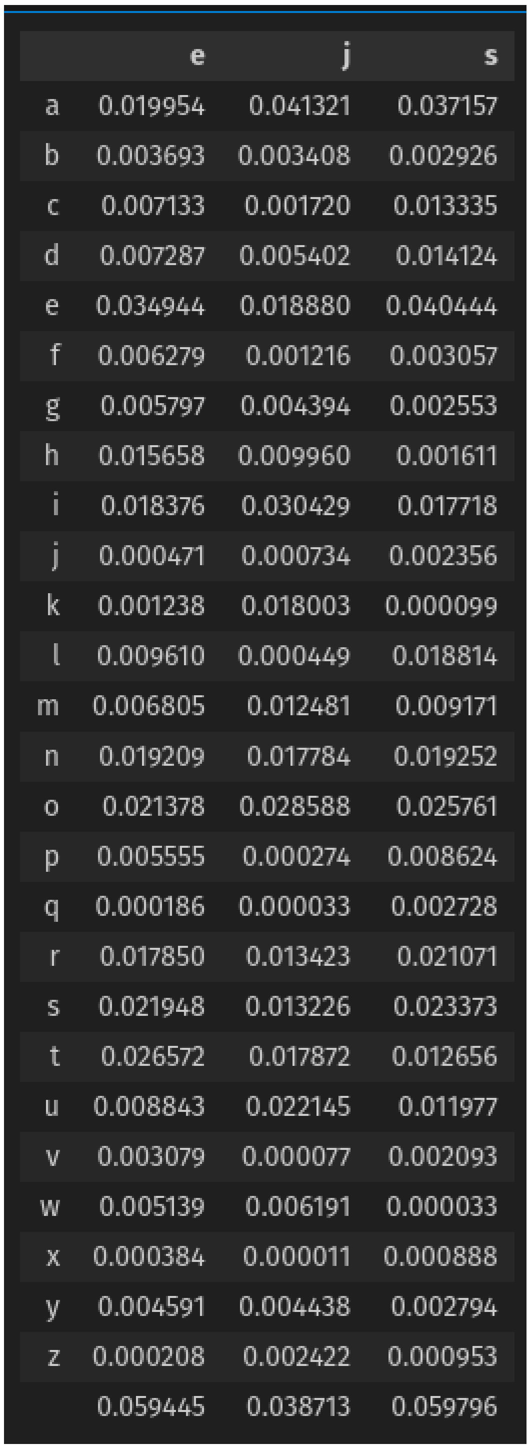
   SOLUTION

   $$\theta_{i,e} := \hat{p}(c_i \mid y = e) = \frac{b_{i,e} + \alpha}{\sum_k b_{k,e} + \alpha * K_c}$$

   where K_c is the number of character classes $c_1 \dots c_{27}$, $b_{i,e}$ is the count of occurrences of the character c_i across all documents belonging to language class e (English). As before, α is our smoothing parameter. Once we make the necessary substitutions, we have

   $$\theta_{i,e} := \hat{p}(c_i \mid y = e) = \frac{b_{i,e} + 0.5}{\sum_k b_{k,e} + 0.5 * 27}$$

   For θ_e, please see the first column vector in the table below (column e).



	e	j	s
a	0.019954	0.041321	0.037157
b	0.003693	0.003408	0.002926
c	0.007133	0.001720	0.013335
d	0.007287	0.005402	0.014124
e	0.034944	0.018880	0.040444
f	0.006279	0.001216	0.003057
g	0.005797	0.004394	0.002553
h	0.015658	0.009960	0.001611
i	0.018376	0.030429	0.017718
j	0.000471	0.000734	0.002356
k	0.001238	0.018003	0.000099
l	0.009610	0.000449	0.018814
m	0.006805	0.012481	0.009171
n	0.019209	0.017784	0.019252
o	0.021378	0.028588	0.025761
p	0.005555	0.000274	0.008624
q	0.000186	0.000033	0.002728
r	0.017850	0.013423	0.021071
s	0.021948	0.013226	0.023373
t	0.026572	0.017872	0.012656
u	0.008843	0.022145	0.011977
v	0.003079	0.000077	0.002093
w	0.005139	0.006191	0.000033
x	0.000384	0.000011	0.000888
y	0.004591	0.004438	0.002794
z	0.000208	0.002422	0.000953
	0.059445	0.038713	0.059796

3. Print θ_j, θ_s and include in final report the class conditional probabilities for Japanese and Spanish.

   SOLUTION

   Please see the j and s columns in the preceding table for vectors θ_j and θ_s.

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 and include in final report.

   SOLUTION

   $x =< 164, 32, 53, 57, 311, 55, 51, 140, 140, 3, 6, 85, 64, 139, 182, 53, 3, 141, 186, 225, 65, 31, 47, 4, 38, 2, 498 >$

   Where each entry in the ith index of this vector denotes the count of character c_i within document e10.txt

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.

SOLUTION

We note that $\log \hat{p}(x \mid y) = \sum_{i=1}^{d} \log \theta_{i,y}^{x_i} = \sum_{i=1}^{d} x_i \log \theta_{i,y}$

$\hat{p}(x \mid y = e) = exp(-10904.720647602408)$
$\hat{p}(x \mid y = j) = exp(-11989.469322201065)$
$\hat{p}(x \mid y = s) = exp(-11338.321401996694)$

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.

SOLUTION

According to Bayes' rule, $\hat{p}(y \mid x) = \frac{\hat{p}(x|y)p(y)}{p(x)}$. We can express $p(x)$ as $p(x) = \prod_{i=1}^{n} p(c_i)$, where n is the number of characters $\in x$ and $p(c_i)$ is the probability of observing a character c_i across all documents. It follows that $\log p(x) = \sum_{i=1}^{n} \log(p(c_i))$.

We can write our first expression in terms of log probabilities:

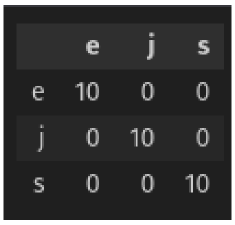
$$\log \hat{p}(y \mid x) = \log \frac{\hat{p}(x \mid y)p(y)}{p(x)} = \log \hat{p}(x \mid y) + \log p(y) - \log p(x)$$

$\hat{p}(y = e \mid x) = exp(-2904.6029864563616)$
$\hat{p}(y = j \mid x) = exp(-3989.3516610550187)$
$\hat{p}(y = s \mid x) = exp(-3338.2037408506485)$

The predicted class label for x will be e.

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.

SOLUTION



	e	j	s
e	10	0	0
j	0	10	0
s	0	0	10

8. If you take a test document and 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.

SOLUTION

Shuffling the order of a test document's characters arbitrarily should not affect the Naive Bayes classifier's prediction. This is because the Naive Bayes model assumes the conditional independence of features. For our model, this means that the probability of one character being present in the input text is assumed to tell us nothing about the probabilities of other characters appearing in the text. In short, the order of characters is not taken into account in our model and thus does not affect its predictions.

4 Simple Feed-Forward Network (20pts)

In this exercise, you will derive, implement back-propagation for a simple neural network and compare your output with some standard library's output. Consider the following 3-layer neural network.

$$\hat{y} = f(x) = g(W_2\sigma(W_1x))$$

Suppose $x \in \mathbb{R}^d$, $W_1 \in \mathbb{R}^{d_1 \times d}$, and $W_2 \in \mathbb{R}^{k \times d_1}$ i.e. $f : \mathbb{R}^d \to \mathbb{R}^k$, Let $\sigma(z) = [\sigma(z_1), ..., \sigma(z_n)]$ for any $z \in \mathbb{R}^n$ where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid (logistic) activation function and $g(z_i) = \frac{exp(z_i)}{\sum_{i=1}^k exp(z_i)}$ is the softmax function. Suppose the true pair is (x, y) where $y \in \{0, 1\}^k$ with exactly one of the entries equal to 1, and you are working with the cross-entropy loss function given below,

$$L(x, y) = -\sum_{i=1}^{k} y\text{log}(\hat{y})$$

1. Derive backpropagation updates for the above neural network. (5 pts)
2. Implement it in NumPy or PyTorch using basic linear algebra operations. (e.g. You are not allowed to use auto-grad, built-in optimizer, model, etc. in this step. You can use library functions for data loading, processing, etc.). Evaluate your implementation on MNIST dataset, report test errors and learning curve. (10 pts)
3. Implement the same network in PyTorch (or any other framework). You can use all the features of the framework e.g. auto-grad etc. Evaluate it on MNIST dataset, report test errors, and learning curve. (2 pts)
4. Try different weight initialization a) all weights initialized to 0, and b) initialize the weights randomly between -1 and 1. Report test error and learning curves for both. (You can use either of the implementations) (3 pts)

You should play with different hyperparameters like learning rate, batch size, etc. for your own learning. You only need to report results for any particular setting of hyperparameters. You should mention the values of those along with the results. Use $d_1 = 300$, $d_2 = 200$. For optimization use SGD (Stochastic gradient descent) without momentum, with some batch size say 32, 64, etc. MNIST can be obtained from here (https://pytorch.org/vision/stable/datasets.html)