HOMEWORK 4

>>NAME HERE<<
>>ID HERE<<

Instructions:

- Please submit your answers in a single pdf file. Preferably made using latex. No need to submit latex code. See piazza post @114 for some recommendations on how to write the answers.
- Submit code for programming exercises. You can use any programming language you like, but we highly recommend python and pytorch.
- Submit all the material on time.

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

Suppose the world generates a single observation $x \sim$ multinomial$(\boldsymbol{\theta})$, where the parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ with $\theta_i \geq 0$ and $\sum_{i=1}^{k} \theta_i = 1$. Note $x \in \{1, \ldots, k\}$. You know $\boldsymbol{\theta}$ 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. (5 pts)
Strategy 2: You mimic the world by generating a prediction $\hat{x} \sim$ multinomial$(\boldsymbol{\theta})$. (Hint: your randomness and the world's randomness are independent) (5pts)

2 Best Prediction Under Different Misclassification Losses (10 pts)

Like in the previous question, the world generates a single observation $x \sim$ multinomial$(\boldsymbol{\theta})$. Let $c_{ij} \geq 0$ denote the loss you incur, if $x = i$ but you predict $\hat{x} = j$, for $i, j \in \{1, \ldots, k\}$. $c_{ii} = 0$ 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 $\hat{x}$.

3 Best Prediction Under Online Learning (10 pts)

Here we consider a repeated game: In each round $t = 1, 2, \ldots, T$

- An adversary choose a real number $y_t \in [0, 1]$ and he keeps it secret;
- You try to guess the real number, choosing $x_t \in [0, 1]$;
- The adversary's number is revealed and you pay the squared difference $(x_t - y_t)^2$.

To simplify the game a bit, we assume that the adversary is drawing the numbers i.i.d. from some fixed distribution over $[0, 1]$. However, he is still free to decide which distribution at the beginning of the game. Suppose μ is the mean of the distribution he chose and σ^2 is the variance.

(i) If we knew the distribution, how would you decide the optimal strategy to minimize your payments in T rounds in expectation? Please derive your optimal strategy and give the payments in T rounds in expectation of your strategy. (5 pts)

(ii) However, usually we need to decide our strategy without knowing the distribution, so it's important to measure whether our strategy has a good performance. It is natural to benchmark our strategy with respect to the optimal one (you are asked to derive it in (i)). Please design a quantity to measure such benchmark. (5pts)

4 Language Identification with Naive Bayes (5 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!)
In the following questions, you may use the additive smoothing technique to smooth categorical data, in case that the estimated probability is zero. Given N data samples $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^N$, where $\boldsymbol{x}^{(i)} = [x_1^{(i)}, \dots, x_j^{(i)}, \dots, x_{M_i}^{(i)}]$ is a bag of characters, M_i is the total number of characters in $\boldsymbol{x}^{(i)}$, $x_j^{(i)} \in S, y^{(i)} \in L$ and we have $|S| = K_S, |L| = K_L$. Here S is the set of all character types, and L is the set of all classes of data labels. Then by the additive smoothing with parameter α, we can estimate the conditional probability as

$$P_\alpha(a_s \mid y = c_k) = \frac{(\sum_{i=1}^N \sum_{j=1}^{M_i} \mathbb{1}[x_j^{(i)} = a_s, y^{(i)} = c_k]) + \alpha}{(\sum_{b_s \in S} \sum_{i=1}^N \sum_{j=1}^{M_i} \mathbb{1}[x_j^{(i)} = b_s, y^{(i)} = c_k]) + K_S\alpha},$$

where $a_s \in S, c_k \in L$. Similarly, we can estimate the prior probability

$$P_\alpha(Y = c_k) = \frac{(\sum_{i=1}^N \mathbb{1}[y^{(i)} = c_k]) + \alpha}{N + K_L\alpha},$$

where $c_k \in L$ and N is the number of training samples.

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 the prior probabilities.

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 which is a vector with 27 elements.

3. Print θ_j, θ_s, the class conditional probabilities for Japanese and Spanish.

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.

5. For the x of e10.txt, 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 $x = (x_1, \dots, x_d)$. 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. Also, Store all probabilities here and below in $\log()$ internally to avoid underflow. This also means you need to do arithmetic in log-space.

6. For the x of e10.txt, 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.

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			

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.

5 Simple Feed-Forward Network (30 pts)

In this exercise, you will be 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.

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \boldsymbol{g}(\boldsymbol{W_2}\boldsymbol{\sigma}(\boldsymbol{W_1}\mathbf{x}))$$

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

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

- Derive backpropagation updates for the above neural network. (10 pts)
- 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 error and learning curve. (10 pts)
- 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 error and learning curve. (6 pts)
- Try different weight initializations a) all weights initialized to 0, and b) Intialize the weights randomly between -1 and 1. Report test error and learning curves for both. (You can use either of the implementations) (4 pts)

You should play with different hyperparameters like learning-rate, batch size etc. for your own learning. You only need to report reuslts 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)