

## INSTRUCTIONS

- **Due:** Wednesday, February 9 2022 at 11:59 PM EDT.
- **Format:** Complete this pdf with your work and answers. Whether you edit the latex source, use a pdf annotator, or hand write / scan, make sure that your answers (tex'ed, typed, or handwritten) are within the dedicated regions for each question/part. If you do not follow this format, we may deduct points.
- **How to submit:** Submit a pdf with your answers on Gradescope. Log in and click on our class 10-315, click on the appropriate *Written* assignment, and upload your pdf containing your answers. Don't forget to submit the associated *Programming* component on Gradescope if there is any programming required.
- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information.

Name	
Andrew ID	
Hours to complete (both written and programming)?	

## Q1. [16 pts] MLE

Consider the following distribution with parameters  $k$  and  $\alpha$ :

$$p(x \mid k, \alpha) = \begin{cases} \frac{\alpha k^\alpha}{x^{\alpha+1}} & x \in [k, \infty) \\ 0 & \text{otherwise} \end{cases}$$

We also have that  $k \in (0, \infty)$  and  $\alpha \in (0, \infty)$ .

This distribution is often used for modeling the distribution of wealth in society, fitting the trend that a large portion of wealth is held by a small fraction of the population. This is due to its nature as a skewed, heavy-tailed distribution (notice that the probability decays polynomially in value of the random variable  $x$ , unlike Gaussian, exponential, Laplace or Poisson distributions where the probability decays exponentially in the value of the random variable).

Suppose you have a dataset  $\mathcal{D}$  which contains  $N$  i.i.d samples  $x_1, x_2, \dots, x_N$  drawn from the above distribution.

- (a) [4 pts] Derive the likelihood  $L(k, \alpha; \mathcal{D})$  and log-likelihood  $\ell(k, \alpha; \mathcal{D})$ .

$\ell(k, \alpha; \mathcal{D})$ :

- (b) [8 pts] Give the MLE for the parameter  $\alpha$ , assuming the parameter  $k$  is fixed.

$\hat{\alpha}_{MLE}$ :

(c) [4 pts] Next, give the MLE for the parameter  $k$ . Here  $\alpha$  may be any fixed value or set to its MLE.

*Hint:* You may be tempted to conclude that MLE of  $k$  is infinity, but when  $k = \infty$ , what happens to  $p(x | k, \alpha)$ ?

$\hat{k}_{MLE}$ :

## Q2. [10 pts] MAP

We've seen in lecture how generative classifiers like Naive Bayes use information about the distributions of the variables in the training data to derive an optimal classifier. MLE is commonly used for estimating the parameters of these distributions since it is generally easy to compute and behaves well asymptotically (i.e. with lots of data). However, using MLE can be problematic when we don't have enough data to get good parameter estimates. When this happens, we need some other method of estimating parameters that doesn't require lots of training data. This is where MAP estimation can be useful since MAP combines a prior assumption about the underlying distribution with the given training data to estimate parameters.

**(a)** [2 pts] MAP definition.

Let  $D$  be some dataset with rows we assume are conditionally independent and are taken from some distribution with parameter  $\theta$ . By definition, we know that  $\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} P(\theta \mid D)$ . Use this definition to derive the equivalent definition  $\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} P(D \mid \theta)P(\theta)$ , justifying each step.

**Solution:**

**(b)** [8 pts] MAP Estimation for Bernoulli random variables.

Consider a dataset  $D$  composed of the outcomes of independent coin flips using an unbiased coin. Taking each coin flip to be a  $\operatorname{Ber}(\theta)$  random variable, we can use MAP to estimate the coin's bias  $\theta$ . We'll use as a prior  $\theta \sim \beta(x, y)$ . Here, the prior represents pseudo-observations; we haven't observed these outcomes, but they reflect our prior beliefs about the bias of the coin.

Our goal is to consider what happens as we vary (1) the size of  $D$  and (2) the correctness of our prior. Compute both the MAP and MLE estimates of  $\theta$  under the following contexts: [You can use the MAP and MLE expressions from lecture slides.]

**(i)** [1 pt]  $D = \{1 H, 4 T\}$ ,  $\theta \sim \beta(3, 3)$

**Solution:**

**(ii)** [1 pt]  $D = \{8 H, 4 T\}$ ,  $\theta \sim \beta(3, 3)$

**Solution:**

- (iii) [1 pt]  $D = \{16 H, 10 T\}$ ,  $\theta \sim \beta(3, 3)$

**Solution:**

- (iv) [1 pt]  $D = \{16 H, 10 T\}$ ,  $\theta \sim \beta(2, 4)$

**Solution:**

- (v) [1 pt]  $D = \{40 H, 40 T\}$ ,  $\theta \sim \beta(2, 4)$

**Solution:**

- (vi) [3 pts] Give a short (2-3 sentence) summary of your findings regarding how (1) the size of the dataset and (2) our choice of prior affects the estimate for  $\theta$ , given that the coin is actually fair. For (iv) and (v) consider the effect an incorrect prior has on our MLE and MAP estimate for  $\theta$ .

**Solution:**

### Q3. [26 pts] Naive Bayes

Consider a simple learning problem of determining whether Alice and Bob will go to hiking, where  $\mathbf{Y} : Hike \in \{T, F\}$  given the weather conditions  $\mathbf{X}_1 : Sunny \in \{T, F\}$ , and  $\mathbf{X}_2 : Windy \in \{T, F\}$  by a Naive Bayes classifier. Using training data, we estimated the parameters

- $P(Hike = T) = 0.5$
- $P(Sunny = T|Hike = T) = 0.8$
- $P(Sunny = T|Hike = F) = 0.7$
- $P(Windy = T|Hike = T) = 0.4$
- $P(Windy = T|Hike = F) = 0.5$

Assume that the *true* distribution of  $\mathbf{X}_1$ ,  $\mathbf{X}_2$ , and  $\mathbf{Y}$  satisfies the Naive Bayes assumption of conditional independence with the above parameters.

- (a) [2 pts] Assume  $\mathbf{X}_1 : Sunny$  and  $\mathbf{X}_2 : Windy$  are truly independent given *Hike*. Write down the Naive Bayes decision rule for this problem using *both*  $\mathbf{X}_1$  and  $\mathbf{X}_2$  as features.

**Solution:**

- (b) [8 pts] Given the decision rule above, write down  $\mathbf{P}(\mathbf{X}_1, \mathbf{X}_2|\mathbf{Y})$  and the Naive Bayes decision for each setting of the weather conditions in the table below:

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{Y}$	$\mathbf{P}(\mathbf{X}_1, \mathbf{X}_2 \mathbf{Y})$	$\mathbf{f}(\mathbf{X}_1, \mathbf{X}_2)$
F	F	F		
F	F	T		
F	T	F		
F	T	T		
T	F	F		
T	F	T		
T	T	F		
T	T	T		

Table 1:  $\mathbf{Y}$  is the true decision, while  $f(X_1, X_2)$  is the decision made by Naive Bayes classifier.

- (c) [2 pts] What is the estimated error rate i.e.  $\mathbf{P}(\mathbf{f}(\mathbf{X}_1, \mathbf{X}_2) \neq \mathbf{Y})$  for the Naive Bayes classifier using these two features?

**Solution:**

Next, suppose we gather more information about weather conditions and introduce a new feature denoting  $\mathbf{X}_3$  :  $Rainy \in \{T, F\}$ . Assume that each day the weather can be **either** *Rainy* **or** *Sunny*. That is, it can not be both *Sunny* **and** *Rainy* (similarly, it can not be not *Sunny* **and** not *Rainy*).

(d) [2 pts] In the above new case, are any of the Naive Bayes assumptions violated? Why or why not?

**Solution:**



- (e) [8 pts] Given the decision rule above, write down  $\mathbf{P}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3|\mathbf{Y})$ ,  $\mathbf{P}(\mathbf{X}_1|\mathbf{Y})\mathbf{P}(\mathbf{X}_2|\mathbf{Y})\mathbf{P}(\mathbf{X}_3|\mathbf{Y})$  and the Naive Bayes decision for each setting of the weather conditions in the table below. Notice that when calculating the Naive Bayes prediction  $f(X_1, X_2, X_3)$ , any violations of the Naive Bayes assumption are ignored.

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{Y}$	$\mathbf{P}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3 \mathbf{Y})$	$\mathbf{P}(\mathbf{X}_1 \mathbf{Y})\mathbf{P}(\mathbf{X}_2 \mathbf{Y})\mathbf{P}(\mathbf{X}_3 \mathbf{Y})$	$\mathbf{f}(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$
F	F	F	F			
F	F	F	T			
F	F	T	F			
F	F	T	T			
F	T	F	F			
F	T	F	T			
F	T	T	F			
F	T	T	T			
T	F	F	F			
T	F	F	T			
T	F	T	F			
T	F	T	T			
T	T	F	F			
T	T	F	T			
T	T	T	F			
T	T	T	T			

Table 2:  $\mathbf{Y}$  is the true decision, while  $f(X_1, X_2, X_3)$  is the decision made by Naive Bayes classifier.

- (f) [4 pts] What is the estimated error rate when the Naive Bayes classifier uses all *three* features? Does the performance of Naive Bayes improve by observing the new feature *Rainy*? Explain why or why not.

**Solution:**

## Q4. [40 pts] Programming

This part of the assignment will have you implement a Naive Bayes classifier. You will submit your completed `naive_bayes.py` file to Gradescope, where we will run your code against a suite of tests. Your grade will be automatically determined from the testing results. Since you get immediate feedback after submitting your code and you are allowed to submit as many different versions as you like (without any penalty), it's easy for you to check your code as you go.

Our autograder requires that you write your code using Python 3.6.9 and Numpy 1.17.0. Otherwise, when running your program on Gradescope, it may produce a result different from the result produced on your local computer. **Please do not include print statements outside the provided functions, as this may crash the autograder.**

The file `hw1data.pkl` contains data regarding words used in articles from The Economist and articles from The Onion. This programming assignment is focused on identifying which words are characteristic of which articles. You can load the pickle file into Python using `pickle`. After loading the data, you will see that there are 5 variables: `Vocabulary`, `XTrain`, `yTrain`, `XTest`, and `yTest`.

- **Vocabulary** is a  $V \times 1$  dimensional array that contains every word appearing in the documents. When we refer to the  $j^{\text{th}}$  word, we mean `Vocabulary[j,0]`.
- **XTrain** is a  $n \times V$  dimensional matrix describing the  $n$  documents used for training your Naive Bayes classifier. The entry `XTrain[i,j]` is 1 if word  $j$  appears in the  $i^{\text{th}}$  training document and 0 otherwise.
- **yTrain** is a  $n \times 1$  dimensional matrix containing the class labels for the training documents. `yTrain[i,0]` is 1 if the  $i^{\text{th}}$  document belongs to The Economist and 2 if it belongs to The Onion.
- Finally, **XTest** and **yTest** are the same as **XTrain** and **yTrain**, except instead of having  $n$  rows, they have  $m$  rows. This is the data you will test your classifier on and it should not be used for training.

### Logspace Arithmetic

When working with very large or very small numbers (such as probabilities), it is useful to work in *logspace* to avoid numerical precision issues. In logspace, we keep track of the logs of numbers, instead of the numbers themselves. For example, if  $p(x)$  and  $p(y)$  are probability values, instead of storing  $p(x)$  and  $p(y)$  and computing  $p(x) * p(y)$ , we work in log space by storing  $\log(p(x))$ ,  $\log(p(y))$ , and we can compute the log of the product,  $\log(p(x) * p(y))$ , by taking the sum in logspace:  $\log(p(x) * p(y)) = \log(p(x)) + \log(p(y))$ .

If we want the sum of two probabilities, it's a little trickier: if  $l(x) = \log p(x)$  and  $l(y) = \log p(y)$ , then  $\log(p(x) + p(y)) = \log(\exp(l(x)) + \exp(l(y)))$ . If we compute this expression naively we risk overflow or underflow. A good workaround is to factor out  $\exp(l(x))$  or  $\exp(l(y))$ , whichever is larger, before computing the sum.

### Training Naive Bayes

- [8 pts] Complete the function `D = NB_XGivenY(XTrain, yTrain, a=0.001, b=0.9)`. The output `D` is a  $2 \times V$  matrix, where for any word index  $w \in \{1, \dots, V\}$  and class index  $y \in \{1, 2\}$ , the entry `D[y-1,w-1]` is the MAP estimate of  $\theta_{yw} = P(X_w = 1 | Y = y)$  with a `Beta(1.001,1.9)` prior distribution. Here we define  $a = \alpha - 1$  and  $b = \beta - 1$  where  $\alpha, \beta$  are parameters of the Beta distribution. To help with numerical issues clip  $D$  to be in  $[10^{-5}, 1 - 10^{-5}]$  before this function returns it.
- [8 pts] Complete the function `p = NB_YPrior(yTrain)`. The output `p` is the MLE for  $\rho = P(Y = 1)$ .
- [8 pts] Complete the function `yHat = NB_Classify(D, p, X)`. The input `X` is an  $m \times V$  matrix containing  $m$  feature vectors (stored as its rows). The output `yHat` is a  $m \times 1$  matrix of predicted class labels, where `yHat[i]` is the predicted label for the  $i^{\text{th}}$  row of `X`. So, the output vector should take the form  $[[y_0], [y_1], \dots, [y_{m-1}]]$ . [Hint: In this function, you will want to use Logspace Arithmetic to avoid numerical problems.]

- (d) [2 pts] Complete the function `e = NB_ClassificationAccuracy(yHat, yTruth)` which measures the average number of times `yHat` agrees with `yTruth` as a performance metric for the Naive Bayes classifier.

### Questions

- (e) [4 pts] Train your classifier on the data contained in `XTrain` and `yTrain` by running

```
D = NB_XGivenY(XTrain, yTrain)
p = NB_YPrior(yTrain)
```

Use the learned classifier to predict the labels for the article feature vectors in `XTrain` and `XTest` by running

```
yHatTrain = NB_Classify(D, p, XTrain)
yHatTest = NB_Classify(D, p, XTest)
```

Use the function `NB_ClassificationAccuracy` to measure and report the training and testing accuracy by running

```
trainAcc = NB_ClassificationAccuracy(yHatTrain, yTrain)
testAcc = NB_ClassificationAccuracy(yHatTest, yTest)
```

How do the train and test accuracies compare? Which is likely to be more representative of the performance of the trained classifier on a new collection of articles?

**Solution:**

- (f) [5 pts] In this question we explore how the size of the training data set affects the test and train accuracy. For each value of  $m$  in  $\{100, 130, 160, \dots, 450\}$ , train your Naive Bayes classifier on the first  $m$  training examples (that is, use the data given by `XTrain[0:m]` and `yTrain[0:m]`). Plot the training and testing accuracy for each such value of  $m$ . The  $x$ -axis of your plot should be  $m$ , the  $y$ -axis should be accuracy, and there should be one curve for training accuracy and one curve for testing accuracy.

- Explain the general trend of both the curves.
- What would you expect to happen to the test accuracy of the classifier if the Naive Bayes assumption is satisfied and we have infinite training data?

**Solution:**

- (g) [5 pts] We're now going to compare Naive Bayes with logistic regression. **For the logistic regression components, you are encouraged/expected to use the code `logistic_regression.py` we provide in the handout.** To examine things in more detail and ensure reasonable runtime/convergence/etc., we're going to restrict our *base* dimension to  $d = 10$ . In other words, our train data matrix will now be  $450 \times 10$  and our test data matrix will now be  $153 \times 10$ . Just select the first 10 features from the vocabulary – ensure that you do *not* accidentally sort or shuffle the vocabulary before selecting these 10 features (for autograder purposes).

1. [2 pts] Baseline values (on this 10-feature dataset)

- (a) Naive Bayes: report the train accuracy and test accuracy of your learned classifier. Clearly label which is train and which is test. No other work required.

**Solution:**

- (b) Logistic regression: run gradient descent for 5000 epochs with learning rate 0.1, and report the final train accuracy and test accuracy. Clearly label which is train and which is test. No other work required.

**Solution:**

2. [2 pts] Now we will study what happens when we replicate features (specifically, we'll replicate 1 feature many times). We'll compare Naive Bayes to logistic regression. Please add 500 repeats of the feature at index 4 (0-indexed). Your train data matrix will be  $450 \times 510$  and your test data matrix will be  $153 \times 510$ .

- (a) Naive Bayes: recompute the train accuracy and test accuracy of your learned classifier. Clearly label which is train and which is test. No other work required.

**Solution:**

- (b) Logistic regression: run gradient descent for 5000 epochs with learning rate 0.1, and compute the final train accuracy and test accuracy. Clearly label which is train and which is test. No other work required.

**Solution:**

3. [1 pt] What do you notice about Naive Bayes vs. logistic regression? Please just write 1 sentence.

**Solution:**

### Collaboration Questions

After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found on the course site.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.

2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details?

3. Did you find or come across code that implements any part of this assignment ? If so, include full details.