

HOMEWORK 2

LINEAR REGRESSION, LOGISTIC REGRESSION, MLE/MAP AND NAIVE BAYES¹

CMU 10-701: MACHINE LEARNING (SPRING 2024)

piazza.com/cmu/spring2024/10701/home

OUT: Wednesday, Feb 7th, 2024

DUE: Monday, Feb 19th, 2024, 11:59pm

START HERE: Instructions

- **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 in our course syllabus for more information: <https://www.cs.cmu.edu/~hchai2/courses/10701/#Syllabus>
- **Late Submission Policy:** See the late homework policy here: <https://www.cs.cmu.edu/~hchai2/courses/10701/#Syllabus>
- **Submitting your work to Gradescope:** There will be two submission slots for this homework on Gradescope, the Written and the Programming:
 - For the written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using the written submission slot. Please use the provided template. The best way to format your homework is by using the Latex template released in the handout and writing your solutions in Latex. However submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Each derivation/proof should be completed in the boxes provided below the question, **you should not move or change the sizes of these boxes** as Gradescope is expecting your solved homework PDF to match the template on Gradescope. If you find you need more space than the box provides you should consider cutting your solution down to its relevant parts, if you see no way to do this, please add an additional page at the end of the homework and guide us there with a ‘See page xx for the rest of the solution’.

¹Compiled on Wednesday 7th February, 2024 at 20:20

- You are also required to upload your code, which you wrote to solve the final questions of this homework, to the Programming submission slot. Your code may be run by TAs so please make sure it is in a workable state.

Regrade requests can be made after the homework grades are released, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For \LaTeX users, use \blacksquare and \bullet for shaded boxes and circles, and don't change anything else. If an answer box is included for showing work, **you must show your work!**

1 Regularized Linear Regression [5 Points]

1. [5 Points] Consider the following linear regression model: for each data point in $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$,

$$y^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + \epsilon \text{ where } y^{(i)}, \epsilon \in \mathbb{R} \text{ and } \mathbf{w}, \mathbf{x}^{(i)} \in \mathbb{R}^{d+1}$$

In matrix notation, we can express this linear relationship for all data points as:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon} \text{ where } \mathbf{y}, \boldsymbol{\epsilon} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n \times (d+1)}, \text{ and } \mathbf{w} \in \mathbb{R}^{d+1}$$

Assuming the residuals are normal and i.i.d. ($\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$), we can write:

$$\mathbf{y} | \mathbf{X}, \mathbf{w} \sim \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I})$$

Now assume that we have a Gaussian prior on \mathbf{w} :

$$\mathbf{w} \sim \mathcal{N}\left(0, \frac{2\sigma^2}{\lambda} \mathbf{I}\right)$$

for some fixed $\lambda > 0$. Recall that in ridge regression, the optimal parameter vector is given by:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2.$$

Show that the solution to the MAP estimate $\mathbf{w}_{\text{MAP}}^*$ in this setting is the same as the one obtained from ridge regression.

(Hint: Start by writing down the expression for the negative log posterior and show that minimizing it gives the same solution as minimizing the OLS with Ridge regression.)

2 Convexity of Logistic Regression [18 points]

Consider a binary classification problem where the goal is to predict a label $y \in \{0, 1\}$, given an input $\mathbf{x} \in \mathbb{R}^d$. A method that you can use for this task is *logistic regression*. In logistic regression, we model the log-odds as an affine function of the data and find weights to maximize the likelihood of our data under the resulting model.

Recall that an *affine function* f takes the form $f(x) = w^\top x + c$ with $c \in \mathbb{R}$ and $w, x \in \mathbb{R}^n$. In other words, it is a linear function composed with a translation.

2.1 Convex Optimization

Recall that the log-likelihood for a logistic regression model can be written as

$$\mathcal{L}(w) = \log P(y|\mathbf{X}, w) = \sum_{i=1}^n [y_i w^\top x_i - \log(1 + \exp(w^\top x_i))].$$

Our goal is to find the weight vector w that maximizes this likelihood. Unfortunately, for this model, we cannot derive a closed-form solution with MLE. An alternative way to solve for w is to use gradient *ascent*, and update w step by step towards the optimal w . But we know gradient ascent will converge to the optimal solution w that maximizes the conditional log likelihood \mathcal{L} when \mathcal{L} is concave. In this question, you will prove that \mathcal{L} is indeed a concave function (and hence, the negative conditional log likelihood is a convex function).

1. [3 points] A real-valued function $f : S \rightarrow \mathcal{R}$ defined on a convex set S , is said to be *convex* if

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2), \forall x_1, x_2 \in S, \forall t \in [0, 1].$$

Show that a linear combination of n convex functions, f_1, f_2, \dots, f_n , $\sum_{i=1}^n a_i f_i(x)$ is also a convex function $\forall a_i \in \mathbb{R}^+$.

2. **[1 point]** Show that a linear combination of n concave functions, f_1, f_2, \dots, f_n , $\sum_{i=1}^n a_i f_i(x)$ is also a concave function $\forall a_i \in R^+$. Recall that if a function $f(x)$ is convex, then $-f(x)$ is concave. (You can use the result from part (1))

3. **[3 points]** Another property of twice differentiable convex functions is that the second derivative is non-negative. Using this property, show that $f(x) = \log(1 + \exp x)$ is a convex function. Note that this property is both sufficient and necessary. i.e. (if $f''(x)$ exists, then $f''(x) \geq 0 \iff f$ is convex)

4. **[3 points]** Let $f_i : \mathcal{S} \rightarrow \mathcal{R}$ for $i = 1, \dots, n$ be a set of convex functions. Is $f(x) = \max_i f_i(x)$ also convex? If yes, prove it. If not, provide a counterexample.

5. **[8 points]** Show that the log likelihood of *Logistic Regression* is a concave function. You may use the fact that if f and g are both convex, twice differentiable and g is non-decreasing, then $g \circ f$ is convex.

3 Naïve Bayes [24 Points]

Let $X = (x_1, x_2, \dots, x_d)$ denote a set of features and $y \in \{0, 1\}$ denote a binary label. Recall that naïve Bayes models the conditional label distribution $P(y \mid X)$ via the conditional distribution of features given the label $P(X \mid y)$:

$$P(y \mid X) \propto P(X \mid y)P(y)$$

1. **Multinomial Naïve Bayes** Suppose that each feature x_i takes values in the set $\{1, 2, \dots, K\}$. Further, suppose that the label distribution is Bernoulli, and the feature distribution conditioned on the label is multinomial.

- (a) **[2 points]** What is the total number of parameters of the model under the naïve Bayes assumption? For full credit you must show your work.

- (b) **[2 points]** What is the total number of parameters of the model without the naïve Bayes assumption? For full credit you must show your work.

- (c) **[4 points]** Suppose we change the set of values that y takes, so that $y \in \{0, 1, \dots, M-1\}$. How would your answers change in both cases (with and without the naïve Bayes assumption)? For full credit you must show your work.

2. **[8 points] Gaussian Naïve Bayes** Now suppose each feature is real-valued, with $x_i \in \mathbb{R}$, and $P(x_i \mid y = c) \sim \mathcal{N}(\mu_{i,c}, 1)$ for $i = 1, 2, \dots, d$ and $c = 0, 1$. Again, suppose that the label distribution is Bernoulli with $P(y = 1) = \pi$. Under the naïve Bayes assumption, show that the decision boundary $\{(x_1, x_2, \dots, x_d) : P(y = 0 \mid x_1, x_2, \dots, x_d) = P(y = 1 \mid x_1, x_2, \dots, x_d)\}$ is linear in x_1, x_2, \dots, x_d .

3. **MLE estimators can be biased** Given N independent observations drawn from a univariate Gaussian distribution $x^{(1)}, \dots, x^{(N)} \sim \mathcal{N}(\mu, \sigma^2)$, recall that the MLE of the mean and variance parameters are

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x^{(i)} \text{ and } \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \hat{\mu})^2.$$

- (a) **[6 points]** Prove that $\hat{\mu}$ is an *unbiased* estimator of μ (by showing $\mathbb{E}[\hat{\mu}] = \mu$) and that $\hat{\sigma}^2$ is a *biased* estimator of σ^2 (by showing $\mathbb{E}[\hat{\sigma}^2] \neq \sigma^2$).

- (b) **[2 points]** Based on your response to the previous question, propose an unbiased estimator of σ^2 .

4 Implementing Naïve Bayes [35 points]

In this question you will implement a naïve Bayes classifier for a text classification problem. You will be given a collection of emails, labeled either spam or non-spam. The goal is to learn a classifier that can distinguish between emails from each class.

We have pre-processed the emails so that they are easier to use in your experiments. We extracted the set of all words that occur in any of the emails. This set is called the vocabulary and we let V be the number of words in the vocabulary. For each email, we produced a feature vector $X = \langle X_0, \dots, X_{V-1} \rangle$, where X_i is equal to 1 if the i^{th} word appears in the email and 0 otherwise. Each email is also accompanied by a class label of either 0 for non-spam or 1 for spam.

When we apply the naïve Bayes classification algorithm, we make two assumptions about the data: first, we assume that our data is drawn iid from a joint probability distribution over the possible feature vectors X and the corresponding class labels Y ; second, we assume for each pair of features X_i and X_j with $i \neq j$ that X_i is conditionally independent of X_j given the class label Y . Under these assumptions, a natural classification rule is as follows: Given a new input X , predict the most probable class label \hat{Y} given X . Formally,

$$\hat{Y} = \underset{y}{\operatorname{argmax}} P(Y = y \mid X)$$

Using Bayes Rule and the naïve Bayes assumption, we can rewrite this classification rule as follows:

$$\begin{aligned} \hat{Y} &= \underset{y}{\operatorname{argmax}} \frac{P(X \mid Y = y)P(Y = y)}{P(X)} && \text{(Bayes Rule)} \\ &= \underset{y}{\operatorname{argmax}} P(X \mid Y = y)P(Y = y) && \text{(Denominator does not depend on } y\text{)} \\ &= \underset{y}{\operatorname{argmax}} P(X_1, \dots, X_V \mid Y = y) P(Y = y) \\ &= \underset{y}{\operatorname{argmax}} \left(\prod_{w=1}^V P(X_w \mid Y = y) \right) P(Y = y) && \text{(Conditional independence).} \end{aligned}$$

The advantage of the naïve Bayes assumption is that it allows us to represent the distribution $P(X \mid Y = y)$ using many fewer parameters than would otherwise be possible. Specifically, since all the random variables are binary, we only need one parameter to represent the distribution of X_w given Y for each $w \in \{1, \dots, V\}$ and $y \in \{0, 1\}$. This gives a total of $2V$ parameters. On the other hand, without the naïve Bayes assumption, it is not possible to factor the probability as above, and therefore we need one parameter for all but one of the 2^V possible feature vectors X and each class label $y \in \{0, 1\}$. This gives a total of $2(2^V - 1)$

parameters. The vocabulary for our data has $V \approx 29,000$ words. Under the naïve Bayes assumption, we require on the order of 58,000 parameters, while without it we need more than 10^{8000} !

Of course, since we don't know the true joint distribution over feature vectors X and class labels Y , we need to estimate the probabilities $P(X | Y = y)$ and $P(Y = y)$ from the training data. For each word index $w \in \{1, \dots, V\}$ and class label $y \in \{0, 1\}$, the distribution of X_w given $Y = y$ is a Bernoulli distribution with parameter θ_{yw} . In other words, there is some unknown number θ_{yw} such that

$$P(X_w = 1 | Y = y) = \theta_{yw} \quad \text{and} \quad P(X_w = 0 | Y = y) = 1 - \theta_{yw}$$

We believe that there is a non-zero (but maybe very small) probability that any word in the vocabulary can appear in an email labeled either spam or non-spam. To make sure that our estimated probabilities are always non-zero, we will impose a Beta(2,2) prior on θ_{yw} and compute the MAP estimate from the training data.

Similarly, the distribution of Y (when we consider it alone) is a Bernoulli distribution with parameter ρ . In other words, there is some unknown number ρ such that

$$P(Y = 1) = p \quad \text{and} \quad P(Y = 0) = 1 - p.$$

In this case, since we have many examples of both spam and non-spam emails, there is no risk of having zero-probability estimates, so we will instead use the MLE.

4.1 Program: Naïve Bayes

Please confine all code you write to a single, self-contained file titled `naive_bayes.py`. Submit this file to Gradescope under Homework 2 Programming.

The file `hw2data.pkl` contains the following elements:

- **Vocabulary:** A text file containing the words occurring in the emails. Each line in this file is a word (note that some of the words may look a little strange because we have run them through a stemming algorithm that tries to make words with common roots look the same. For example, “stemming” and “stemmed” would both become “stem”.) The line number (zero indexed; i.e the first word maps to 0) indicates the id of that word. The file contains 29356 words.
- **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^{th} training document and 0 otherwise.

- `yTrain` is a $n \times 1$ dimensional matrix containing the class labels for the training documents. `yTrain(i,1)` is 0 if the i^{th} document belongs to non-spam and 1 if it belongs to spam.
- `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.

Note that `XTrain` and `XTest` are stored in a sparse array format.

You are free to implement your Naive Bayes classifier any way you wish. However, we have provided the following recommended structure in the handout:

1. Complete the function `D = NB_XGivenY(XTrain, yTrain)`. The output `D` is a $2 \times V$ matrix, where for any word index $w \in \{1, \dots, V\}$ and class index $y \in \{0, 1\}$, the entry `D[y,w-1]` is the MAP estimate of $\theta_{yw} = P(X_w = 1|Y = y)$ with a Beta(2,2) prior distribution. **Note:** to help with numerical issues, you should clip `D` to be in $[10^{-5}, 1 - 10^{-5}]$.
2. Complete the function `p = NB_YPrior(yTrain)`. The output `p` is the MLE for $p = P(Y = 0)$.
3. 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 $m \times 1$ vector of predicted class labels, where `yHat[i]` is the predicted label for the i^{th} row of `X`. **Note:** in this function, you will want to use logspace arithmetic to avoid numerical problems (see the section below).
4. Complete the function `error = ClassificationError(yHat, yTruth)`.

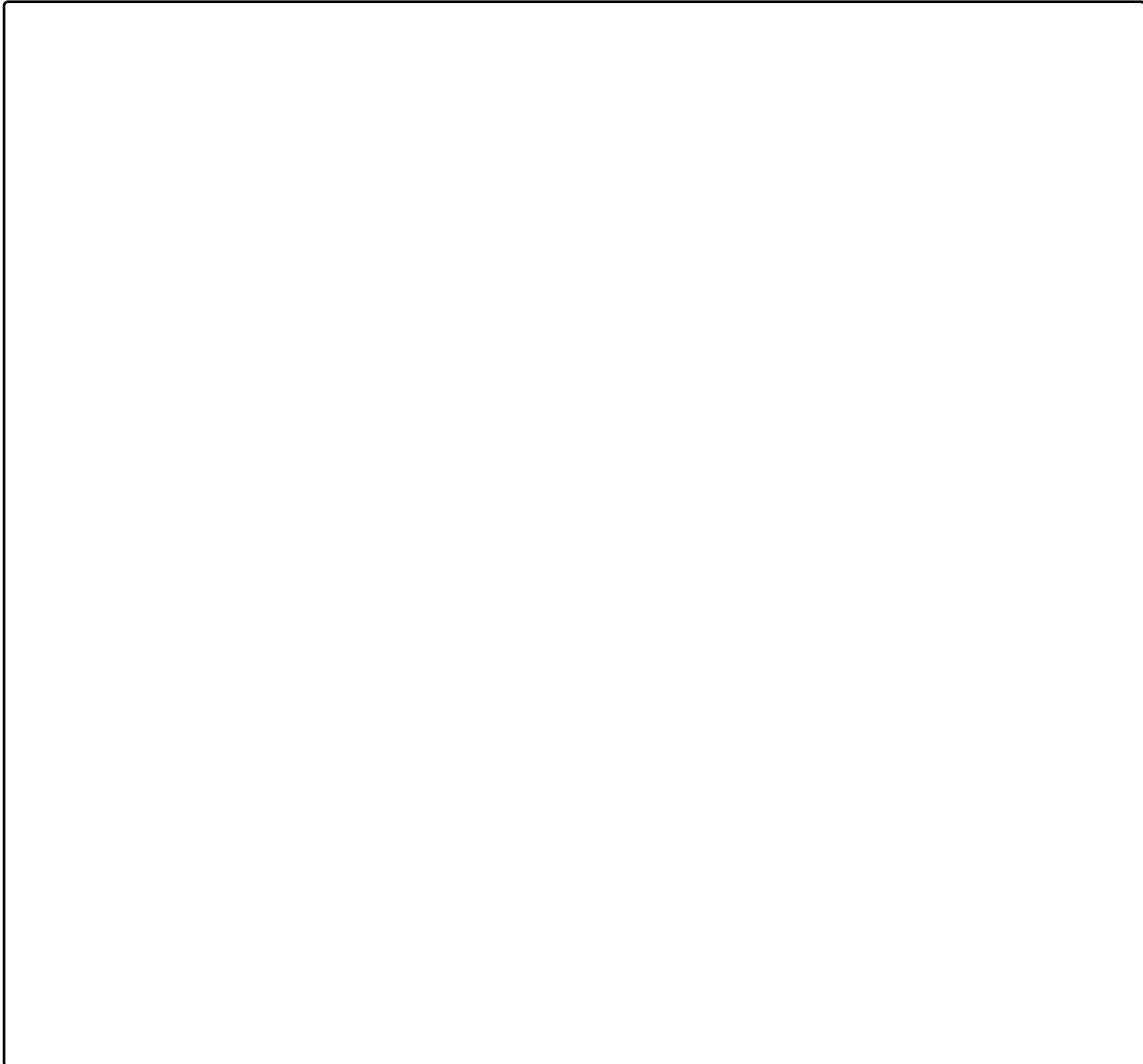
4.2 Logspace Arithmetic

When working with very large or very small numbers (such as probabilities), it is often 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, $\log(p(x) * p(y))$ as $\log(p(x) * p(y)) = \log(p(x)) + \log(p(y))$.

4.3 Empirical Questions

1. **[10 Points]** In this question we explore how the size of the training data set affects the test and train error. For each value of m in $\{400, 800, 1200, \dots, 4400\}$, 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]`).

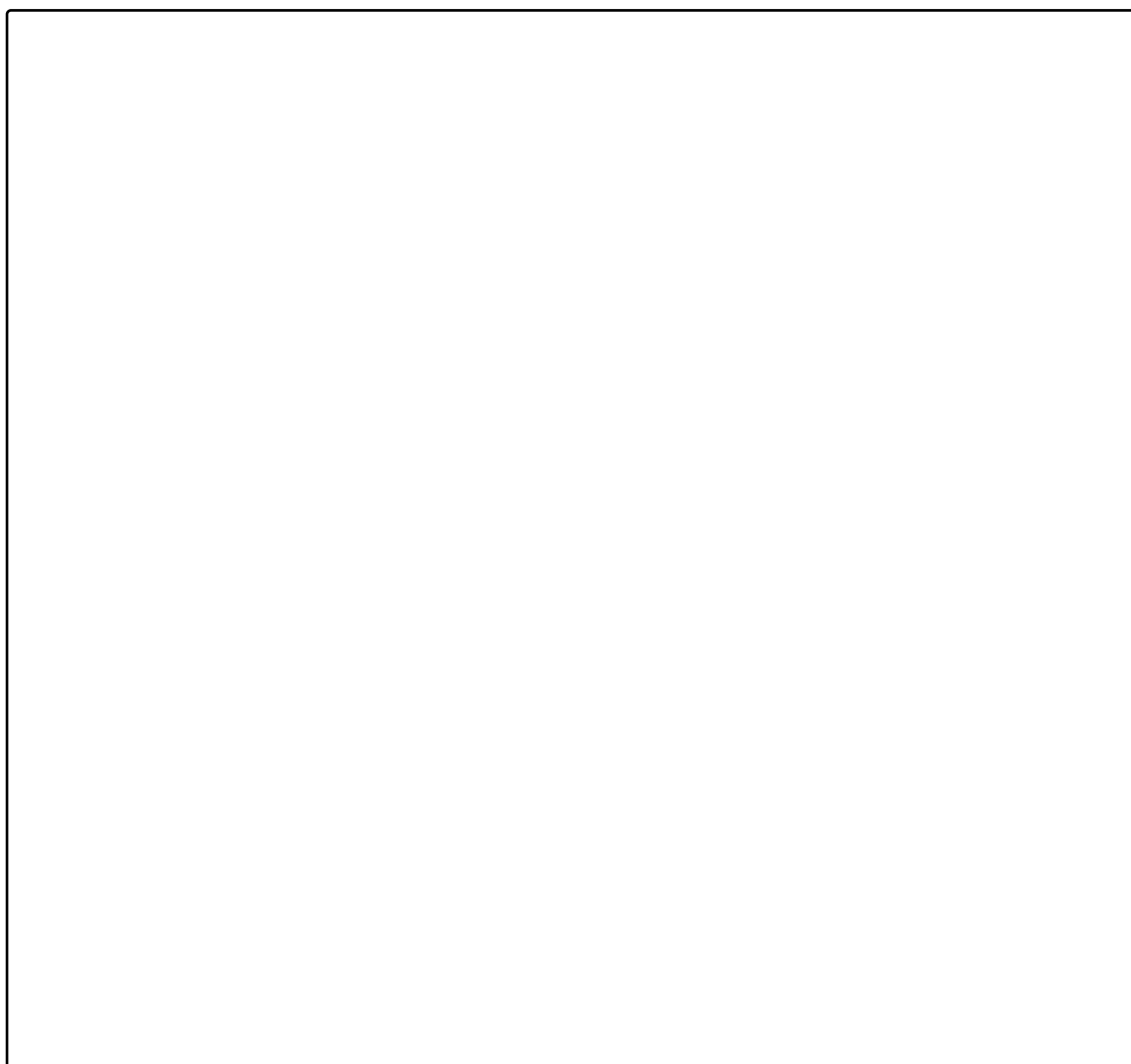
Plot the training and testing error for each such value of m . The x -axis of your plot should be m , the y -axis should be error, and there should be one curve for training error and one curve for testing error; both curves should be on the same graph (same axes). Then explain the general trend of both the training and testing error curves.



2. **[10 Points]** When estimating the class prior and attribute distributions, we currently add 2 observations to each outcome because multiplying by zero can be fatal. In this question, we explore how estimates change when *stronger* priors are added to the parameters.

To make sure that our estimated probabilities are always non-zero, we will impose a $\text{Beta}(\alpha, \beta)$ prior on θ_{yw} .

Plot the training and testing error for $\alpha \in [2, 5, 10, 25, 50, 100]$. Keep β fixed at 2. The x -axis of your plot should be α , the y -axis should be error, and there should be one curve for training error and one curve for testing error; both curves should be on the same graph (same axes). Then, explain the intuition behind the observed changes regarding varying levels of α .



3. Next, we will try to interpret the learned parameters. For this question, revert back to the standard Beta(2,2) prior. Train your classifier on the data contained in **XTrain** and **yTrain**. For each of the following criteria, fill in the table for each label $y \in \{0, 1\}$:

(Note that some of the words may look a little strange because we of the stemming algorithm. #1 should be the word that give the highest value and #5 the fifth highest):

- (a) **[5 Points]** Top five words that the model says are most likely to occur in a document from class y . That is, the top five words according to this metric:

$$P(X_w = 1|Y = y)$$

	Word #1	Word #2	Word #3	Word #4	Word #5
$Y = 0$					
$Y = 1$					

- (b) **[5 Points]** Top five words w according to this metric:

$$\frac{P(X_w = 1|Y = y)}{P(X_w = 1|Y \neq y)}.$$

	Word #1	Word #2	Word #3	Word #4	Word #5
$Y = 0$					
$Y = 1$					

- (c) **[5 Points]** Which list of words is more informative about the class y ? Briefly explain your reasoning.

5 Linear Regression for Time Series [30 Points]

In this problem you will explore fitting a linear regression to predict time series data using OLS and stochastic gradient descent.

5.1 Data Processing

In this problem, you will be using a temperature dataset which can be found in the file `temperature.csv`. This file contains timestamps (column labeled 'Date Time') at 30 minute intervals and corresponding temperature measurements (column labeled 'T (degC)') over eight years of data collection.

Your first task is to split this data into a train set and a test set. We want to use the first 6 years of data as our training set and the last 2 years as our test set. Since the dataset includes one measurement every 30 minutes (i.e. 2 measurements every hour), the training set should contain the first $2 * 24 * 365 * 6 = 105142$ samples and the test set should contain the last $2 * 24 * 365 * 2 = 35040$ samples (i.e. training and test comprises the entire dataset).

Our first task will be to train a model that predicts the temperature at a given time based on the measurements at the previous $D=10$ timesteps, so we use the value 10 for D below to set us up for this.

Concretely, we can write our training portion of the dataset as $X_{\text{train}} = \{x_1, x_2, \dots, x_T\}$ where each x_i is one temperature measurement and $T = 105142$ is the total number of training samples. In this setting, we will use the temperatures $x_1, x_2, x_3, \dots, x_D$ to predict the value at x_{D+1} ; temperatures x_2, x_3, \dots, x_{D+1} to predict the temperature at x_{D+2} ; and so on so that in general we use $x_i, x_{i+1}, \dots, x_{i+(D-1)}$ to predict the value of x_{i+D} .

You need to reformat the training portion of the dataset to follow this framework. You should create an `X_train` matrix that has D columns (i.e. the D consecutive timestamps) and $T - D$ rows. Note that data will be repeated in this matrix, since each row is shifted by only one timestep from the previous row and will therefore contain $D - 1$ of the same temperature values. You should also create a `y_train` vector that contains the target values we want to predict, i.e. $[x_{D+1}, x_{D+2}, \dots, x_T]$.

Similarly, we can define our test portion of our dataset as $X_{\text{test}} = \{x_{T+1}, x_{T+2}, \dots, x_{T+C}\}$ where C is the total number of the test samples. We will create an `X_test` matrix and a `y_test` vector following the same procedure as for the training dataset.

We have now created normalized datasets with 10 features and can use these 10 features to predict the target corresponding to each row.

5.2 Predicting Temperatures using Linear Regression

For this question, please submit all code you wrote in a single, self-contained file titled `time_series.py` on Gradescope.

1. **[4 points]** Fit an OLS linear regression model using `X_train` and `y_train` to find the OLS solution. Report the following values:
 - (a) the weights you learned (including the bias or intercept weight)
 - (b) the time taken to fit the model
 - (c) the MSE on `X_test`

As a reminder, you are not permitted to use libraries other than `numpy` in your implementations.

2. **[8 points]** Repeat the previous question for $D = \{50, 100, 500\}$. For these models, you should just report the weights on the first 10 features and the bias weight, along with the time required to fit each model and the MSE on `X_test`.

3. **[3 points]** Using the times taken to fit the models for $D = 10, 50, 100$, and 500 , estimate the time it would take to fit a model using features from an entire year's worth of data (i.e. $D = 2 * 24 * 365 = 17520$). Report how you estimated the time (i.e. what function did you fit?) and your time estimate in minutes.

4. **[10 points]** Based on our result from the previous part, we may conclude that using OLS will result in a very high computational cost. As an alternative, you will now learn the weights \mathbf{w} and bias b using **stochastic gradient descent**. We will use train and test datasets created for $D = 17520$, i.e. using the data from an entire year.

Your implementation of SGD should use a learning rate of $\eta = 1e - 10$ and run for 20 epochs. Initialize your weights to be uniformly distributed, with each value set to $\frac{1}{D}$ and your bias to be 1. Additionally, while normally you would shuffle or permute the training data points in each epoch of SGD, for this assignment, you should loop through the training dataset in chronological order (i.e., the original ordering of the dataset) without randomly shuffling the data points.

Again, please only use `numpy` to implement SGD from scratch. After using SGD to train a LR model that uses a year's worth of data for 20 epochs, please report the following values:

- (a) Train MSE
- (b) Test MSE
- (c) Total training time
- (d) First 10 items in your final weight vector.

5. [**5 points**] Now we can examine the weights learned by SGD. What takeaways can you observe? Are any features particularly informative in predicting the targets? Please give a 2-3 sentence response describing your observations. **Hint:** consider which features have the largest weights; what observations do those features correspond to?

6 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment?

Solution Yes / No.

- (b) If you answered ‘yes’, give full details (e.g. “Jane Doe explained to me what is asked in Question 3.4”)

Solution

2. (a) Did you give any help whatsoever to anyone in solving this assignment? **Solution**

Yes / No.

- (b) If you answered ‘yes’, give full details (e.g. “I pointed Joe Smith to section 2.3 since he didn’t know how to proceed with Question 2”)

Solution

3. (a) Did you find or come across code that implements any part of this assignment?

Solution Yes / No.

- (b) If you answered ‘yes’, give full details (book & page, URL & location within the page, etc.).

Solution