Homework 2

Decision Trees, Logistic Regression, Linear Regression, and Optimization

CMU 10-701: Machine Learning (Spring 2021)

piazza.com/cmu/spring2021/10701
OUT: February 19, 2021

DUE: Thursday, March 04, 2021 11:59pm
TAs: Stefani Karp, Jeffrey Huang

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 on the course site for more information: https://www.cs.cmu.edu/~aarti/Class/10701_Spring21/index.html.
- **Extension Policy:** See the homework extension policy here: https://www.cs.cmu.edu/~aarti/Class/10701_Spring21/index.html.
- **Submitting your work:**
  - All portions of the assignments should be submitted to Gradescope (https://gradescope.com/).
  - **Programming:** We will autograde your Python code in Gradescope. After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). We recommend debugging your implementation on your local machine (or the linux servers) and making sure your code is running correctly before any submission. Our autograder requires that you write your code using Python 3.6.9 and Numpy 1.17.0.
  - **Written questions:** You must type the answers in the provided .tex file. Handwritten solutions will not be accepted. Make sure to answer each question in the provided box. DO NOT change the size of the boxes, because it may mess up the autograder. Upon submission, make sure to label each question using the template provided by Gradescope. Please make sure to assign ALL pages corresponding to each question.

1 Warm-Up [16 pts]

1. **[2 points]** For the following decision boundary (where points that satisfy the inequality are the positive class and the rest are the negative class), select all types of models that can perform perfect classification according to this boundary. In the problem below, all points in $\mathbb{R}^2$ are under consideration.

   $|x|, |y| \leq 1$

   ○ K-NN

   ● Decision Tree

   ○ Logistic Regression

   ○ Linear Regression

2. **[2 points] True or False:** Naive Bayes assumes that all features x_i are independent of all other features.

   ○ True

   ● False

3. **[2 points] True or False:** In unregularized logistic regression on separable data, checking for 0 change in the weights is always a good stopping criterion.

   ○ True

   ● False

4. **[4 points] Short Answer:** Under what condition will Gaussian Naive Bayes be guaranteed to find a linear decision boundary for any training set?

The decision boundary is $P(Y=1|X=x) = P(Y=0|X=x)$
$\frac{P(X=x|Y=1)P(Y=1)}{P(X=x|Y=0)P(Y=0)} = \sqrt{\frac{|\Sigma_0|}{|\Sigma_1|}} \exp(-\frac{(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)}{2} + \frac{(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)}{2})\frac{\theta}{1-\theta}$
We can expand $-(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + (x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)$
$= -(x^T\Sigma_1^{-1}x - 2x^T\Sigma_1^{-1}\mu_1 + \mu_1^T\Sigma_1\mu_1) + (x^T\Sigma_0^{-1}x - 2x^T\Sigma_0^{-1}\mu_0 + \mu_0^T\Sigma_0\mu_0)$
We can tell from their dimensions that $\mu_1^T\Sigma_1\mu_1$ and $\mu_0^T\Sigma_0\mu_0$ are constant.

If we want to have a linear decision boundary, then we need to have $\Sigma_1 = \Sigma_0$,
so the above equation can be written as: $x^T\Sigma^{-1}(\mu_1 - \mu_0) + c$
We know that $\frac{\theta}{1-\theta}$ is also a constant, so the decision boundary is $c_2 \exp(\frac{x^T\Sigma^{-1}(\mu_1-\mu_0)+c_1}{2}) = 1$
which is $\frac{x^T\Sigma^{-1}(\mu_1-\mu_0)+c_1}{2} = \ln(\frac{1}{c_2})$, and this is linear with respect to x

5. **[6 points]** We are given a set of 8 points

$$(1,1),(2,4),(3,9),(4,16),(5,25),(6,34),(7,43),(8,w)$$

We find that the line $y = mx + h$ best fits this set of 8 points with respect to mean squared loss, for some real numbers m and h, and it has been determined that $m = 4$. What is the value of w? Show your work.

We can write $A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \end{bmatrix}, Y = \begin{bmatrix} 1 \\ 4 \\ 9 \\ 16 \\ 25 \\ 34 \\ 43 \\ w \end{bmatrix}$

(The first column of A represents the bias term)

We know that $\hat{\beta} = (A^T A)^{-1} A^T Y$

We can then find that $\hat{\beta} = \begin{bmatrix} \frac{-7w+54}{28} \\ \frac{7w+272}{84} \end{bmatrix}$

We now know that $\frac{7w+272}{84} = m = 4$, so $w = \frac{64}{7}$

2 Information Gain & Decision Trees [9 pts]

1. Consider the following data with the following features:

A	B	Y
0	0	1
0	1	0
0	2	0
1	0	1
1	1	0
1	2	1

For all problems in this section, do not show your work.

(a) **[2 points]** What is the entropy of Y, $H(Y)$, in bits? Note that the unit "bits" implies that you need to use log base 2 in your calculations. (Please round your answer to the third decimal place, e.g. 0.123)

1

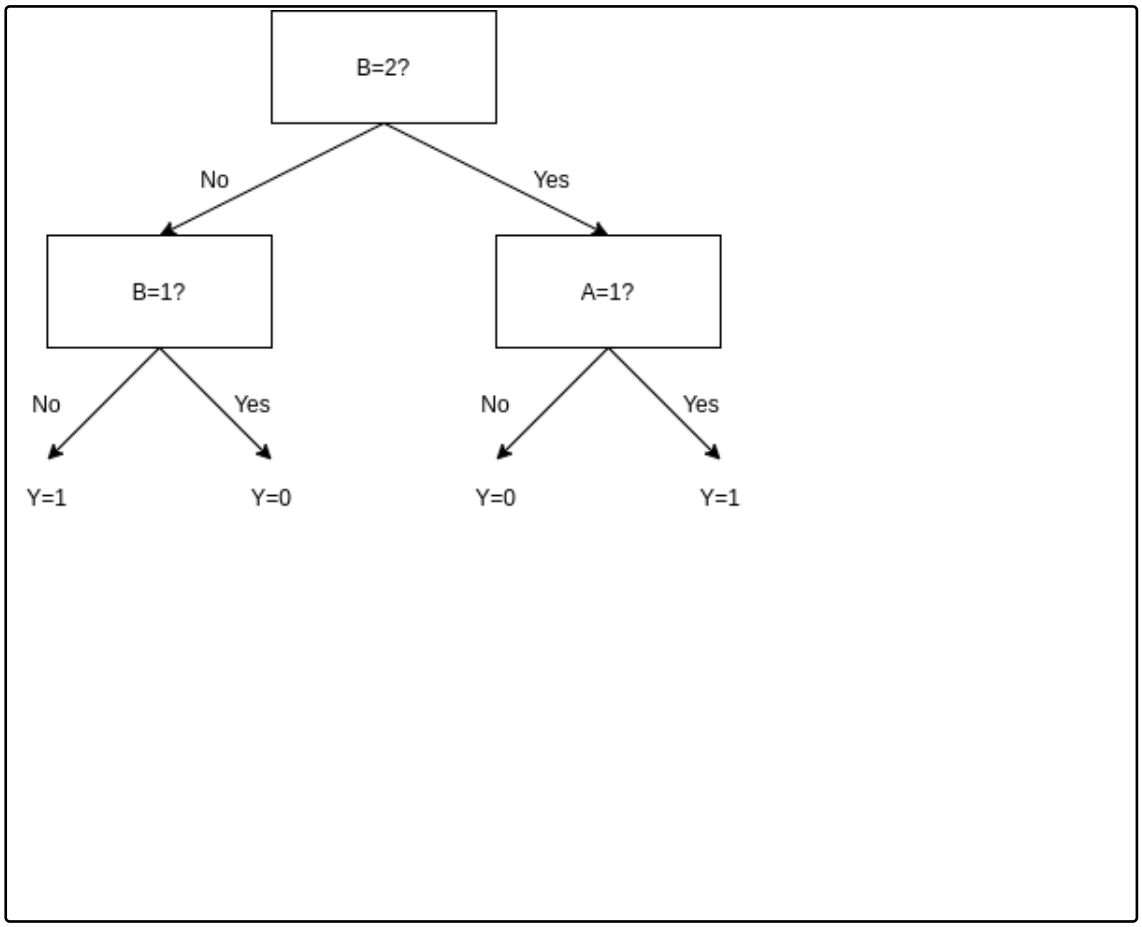
(b) **[2 points]** Compute the expected information gain of Y upon learning A, $I(Y; A)$. (Please round your answer to the third decimal place, e.g. 0.123)

0.082

(c) **[2 points]** Compute the expected information gain of Y upon learning B, $I(Y; B)$. (Please round your answer to the third decimal place, e.g. 0.123)

0.667

(d) **[3 points]** Apply the greedy decision tree algorithm using information gain as the splitting criterion until the above data are perfectly classified and draw the tree below. Assume that each split may only be based on one feature from the table of data above. Label the non-leaf nodes with which feature the tree will split on, the edges with the value of the attribute, and the leaf nodes with the classification decision. If two different variables give the same information gain, pick the variable which comes first in the alphabet. Your final tree should not perform any unnecessary splits (ones where nothing is gained).



3 Linear Regression [28 points]

In this section, we'll be solving some toy (i.e., small synthetic dataset) linear regression problems, in both closed-form and via iterative optimization (gradient descent, specifically).

Consider $X \in \mathbb{R}^{n \times d}$ (for small values of n and d to be defined below), $y \in \mathbb{R}^n$, and $w \in \mathbb{R}^d$. For simplicity, we will not use a bias term in this problem, but please note that it's typically a very good idea to have a bias term in practice. The objective function we are trying to minimize is:

$$f(w) = \|Xw - y\|_2^2 + \lambda \|w\|_2^2.$$

This is a just a *vectorized* way of computing the square loss over an n-item dataset.

Some of the following questions require **only writing**, and *some* require **writing + code** (each sub-question below specifies exactly which type it is).

We are providing you with `linear_regression.py` as a template, which includes a skeleton gradient descent implementation and incomplete functions corresponding to Parts 3-5 below (you should put your code for Parts 3-5 in the template functions named accordingly; do **not** add your code to this pdf). We will not be running automated tests, but you will be required to submit your completed `linear_regression.py` to Gradescope (for academic integrity purposes).

The only package you need/are allowed to use is `numpy`; for consistency among submitted solutions, please run your code using `Python 3.6.9` and `numpy 1.17.0`.

1. **[7 points, writing only]** Show that a closed-form solution for w is $w = (X^\top X + \lambda I)^{-1} X^\top y$ (assuming the inversion is valid).

$f(w) = \|Xw - y\|_2^2 + \lambda \|w\|_2^2$.
$= (Xw - y)^T (Xw - y) + \lambda w^T w$
$= ((Xw)^T - y^T)(Xw - y) + \lambda w^T w$
$= w^T X^T X w - w^T X^T y - y^T X w + y^T y + \lambda w^T w$
$= w^T X^T X w - 2 y^T X w + y^T y + \lambda w^T w$
According to the lecture,
$\frac{\partial}{\partial w} w^T X^T X w = (X^T X + (X^T X)^T) w = 2 X^T X w$
$\frac{\partial}{\partial w} y^T X w = (y^T X)^T = X^T y$
we also know that $\frac{\partial}{\partial w} y^T y = 0$

We can see that $\lambda w^T w = w^T \lambda I w$
now we can get $\frac{\partial}{\partial w} w^T \lambda I w = (\lambda I + (\lambda I)^T) w = 2 \lambda I w$
Therefore, $\frac{\partial}{\partial w} w^T X^T X w - 2 y^T X w + y^T y + \lambda w^T w = 2 X^T X w - 2 X^T y + 2 \lambda I w$
We can set this to 0, so $X^T X w + \lambda I w = X^T y$
$(X^T X + \lambda I) w = X^T y$
Therefore, $w = (X^T X + \lambda I)^{-1} X^T y$

2. **[1 point, writing only]** It turns out that we have another closed-form solution for w as well: $w = X^\top(XX^\top + \lambda I)^{-1}y$ (assuming the inversion is valid). This is less commonly seen/used in linear regression, but it will likely become relevant later in the course, so it's good to keep in mind.

   What is the shape of the matrix we're inverting here, vs. the matrix we inverted in Part 1?

   The shape of the matrix we are inverting here is $n \times n$
   The shape of the matrix we inverted in Part 1 is $d \times d$

3. **[9 points]** Consider the following design matrix `X = [[1,1],[2,3],[3,3]]`. In the notation above, $n = 3, d = 2$. Consider `y = [[1],[3],[3]]`. In what follows, we will try various ways to solve this linear regression problem.

   (a) **[1 point, writing + code]** Attempt to use the closed-form solution in Part 1 to find w, with $\lambda = 0$. Report w or explain what happened if you ran into a problem.

   $w = \begin{bmatrix} -3.1086 \times 10^{-15} \\ 1 \end{bmatrix} \simeq w = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

   (b) **[1 point, writing + code]** Attempt to use the closed-form solution in Part 2 to find w, with $\lambda = 0$. Report w or explain what happened if you ran into a problem.

   XX^T is a singular matrix, so the inversion is not valid.

   (c) **[3 points, writing + code]** Now try (a) and (b) again with ℓ_2 regularization at 3 different values of λ: $1, 10^{-3}, 10^{-5}$.

   What values of w do you find for (a)? Round to 4 decimal places.

   $\lambda = 1$:

   $w = \begin{bmatrix} 0.3636 \\ 0.6591 \end{bmatrix}$

   $\lambda = 10^{-3}$:

   $w = \begin{bmatrix} 1.5947 \times 10^{-3} \\ 0.9986 \end{bmatrix}$

$\lambda = 10^{-5}$:

$w = \begin{bmatrix} 1.5999 \times 10^{-5} \\ 9.9999 \times 10^{-1} \end{bmatrix}$

What values of w do you find for (b)? Round to 4 decimal places.

$\lambda = 1$:

$w = \begin{bmatrix} 0.3636 \\ 0.6591 \end{bmatrix}$

$\lambda = 10^{-3}$:

$w = \begin{bmatrix} 1.5947 \times 10^{-3} \\ 0.9986 \end{bmatrix}$

$\lambda = 10^{-5}$:

$w = \begin{bmatrix} 1.5999 \times 10^{-5} \\ 9.9999 \times 10^{-1} \end{bmatrix}$

What do you notice about this sequence of w values?

The w values get closer to the closed form solution $w = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ as λ decreases

(d) **[3 points, writing + code]** Now use gradient descent to find w (for $\lambda = 0$). What values of w do you find using a learning rate of 0.01 and 10 iterations, 100 iterations, 1000 iterations, 10000 iterations? Round to 4 decimal places.

10 iterations:

$w = \begin{bmatrix} 0.4646 \\ 0.6023 \end{bmatrix}$

100 iterations:

$w = \begin{bmatrix} 0.2675 \\ 0.7711 \end{bmatrix}$

1000 iterations:

$w = \begin{bmatrix} 1.0687 \times 10^{-3} \\ 0.9991 \end{bmatrix}$

10000 iterations:

$w = \begin{bmatrix} 3.9147 * 10^{-15} \\ 1 \end{bmatrix}$

What do you notice about this sequence of w values?

The w values get closer to the closed form solution $w = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ as λ decreases

(e) **[1 point, writing only]** In one sentence, compare the effect of ℓ_2 regularization in part (c) with the effect of *early stopping* in gradient descent (i.e., running gradient descent for a limited number of iterations) in part (d).

ℓ_2 regularization and *early stopping* have similar effect

4. **[2 points]** Consider the following design matrix `X = [[1,2,3],[1,3,3]]` and `Y = [[1],[3]]`. This trivially has a solution: we have 2 equations in 3 unknowns; it is *overparameterized*. It is nevertheless interesting to see how our different approaches behave on this problem.

(a) **[1 point, writing + code]** Attempt to use the closed-form solution in Part 1 to find w. Report w or explain what happened if you ran into a problem. How has this changed from 3(a)?

X^TX is a singular matrix, so the inversion is not valid.
In 3(a), X^TX is not a singular matrix, so we could calculate the inversion and get a closed form solution

(b) **[1 point, writing + code]** Attempt to use the closed-form solution in Part 2 to find w. Report w or explain what happened if you ran into a problem. How has this changed from 3(b)?

$w = \begin{bmatrix} -0.3 \\ 2 \\ -0.9 \end{bmatrix}$
In 3(b), XX^T is a singular matrix, so the inversion is invalid, but in this question, XX^T is not a singular matrix, so we could get a closed form solution.

5. **[9 points]** We're now going to examine a slightly different phenomenon - that of conditioning. We'll focus on gradient descent for the remainder of the problem.

We'll consider 2 different X matrices:
`X1 = [[100,0.001],[200,0.001],[-200,0.0005]]` and
`X2 = [[100,100],[200,100],[-200,100]]`.

We'll use the same Y for both: `Y = [[1],[1],[1]]`.

Complete each of the following questions for *both* $X1$ and $X2$, and add 1 sentence of comparison in each part below your results. Put each answer in the labeled box.

(a) **[3 points, writing + code]** Run gradient descent for 50 iterations with a learning rate of 0.01 and no regularization ($\lambda = 0$). For both $X1$ and $X2$: **a)** What is your final loss value? **b)** What is your final w?

$X1$ loss:

inf

$X1$ w:

$w = \begin{bmatrix} -6.2707 \times 10^{159} \\ -1.3935 \times 10^{154} \end{bmatrix}$

$X2$ loss:

inf

$X2$ w:

$w = \begin{bmatrix} -2.1817 \times 10^{160} \\ -3.5404 \times 10^{159} \end{bmatrix}$

1-sentence comparison:

For both $X1$ and $X2$, the losses are extremely large, and w are extremely small (extremely large in magnitude but negative)

(b) **[3 points, writing + code]** Recall that the Hessian of a function is its matrix of 2nd-order partial derivatives. For the linear regression objective function above (with $\lambda = 0$), the Hessian is: $2X^\top X$. Use `np.linalg.eig` to compute the eigenvalues of the Hessian for $X1$ (first box) and $X2$ (second box).

$X1$:

1.8000×10^5 and 3.6111×10^{-6}

$X2$:

1.8325×10^5 and 5.6754×10^4

1-sentence comparison:

The two eigenvalues of Hessian of $X1$ are very different in magnitude, but the two eigenvalues of Hessian of $X2$ are relatively similar in magnitude

(c) **[3 points, writing + code]** To guarantee convergence of gradient descent, we need to use a learning rate $\leq 1/\lambda_{\max}$. Try setting the learning rate to $1/(2 \times 10^5)$ and rerunning for 50 iterations. For both $X1$ and $X2$: **a)** What loss and **b)** w do you find?

$X1$ loss:

2.8889

$X1$ w:

$$w = \begin{bmatrix} 1.1111 \times 10^{-3} \\ 1.1414 \times 10^{-6} \end{bmatrix}$$

$X2$ loss:

8.8470×10^{-15}

$X2$ w:

$$w = \begin{bmatrix} 8.9439 \times 10^{-11} \\ 1.0000 \times 10^{-2} \end{bmatrix}$$

The Hessian is called ill-conditioned if the ratio of its largest eigenvalue to its smallest eigenvalue is large. Which, if either, of $X1, X2$ yields an ill-conditioned Hessian, and which of $X1, X2$ seems to converge most slowly with the newer learning rate?

$X1$ yields an ill-conditioned Hessian.
$X1$ converges more slowly.

4 Logistic Regression [47 pts]

4.1 Logistic Regression Theory [17 points]

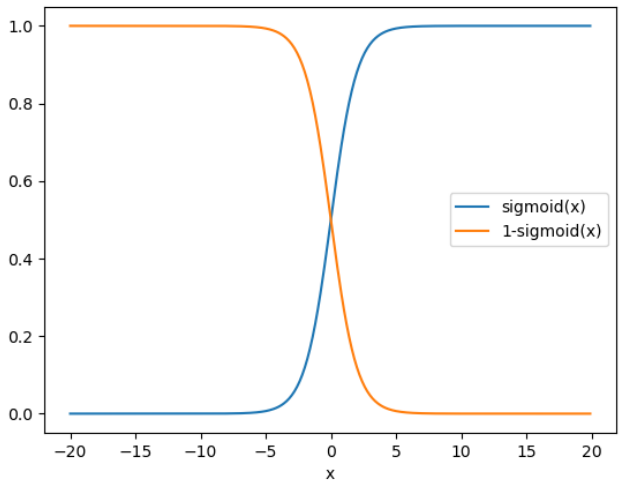
1. [**5 points**] Recall that the **sigmoid function** is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

This is a really important function to understand well!

a. Plot $\sigma(x)$ and $1 - \sigma(x)$ on the same plot. Restrict the domain to $x \in [-20, 20]$. Make sure to label which curve is $\sigma(x)$ and which curve is $1 - \sigma(x)$.
b. What relationship between $\sigma(x)$ and $1 - \sigma(x)$ do you notice? (Beyond the fact that the second is one minus the first :) .)
c. Use this insight to prove a relationship between $\sigma(x)$ and $\sigma(-x)$. Show your work and write your final answer as $\sigma(-x)$ in terms of $\sigma(x)$.

If you solved it correctly, you'll have noticed a very helpful relationship that comes up all the time in logistic regression. Internalizing this is of value!



We can see that $\sigma(x)$ and $1 - \sigma(x)$ are symmetric with respect to $x = 0$

If $f_1(x)$ and $f_2(x)$ are symmetric with respect to $x = 0$,
then $f_1(-x) = f_2(x)$,
so in our case, $\sigma(-x) = 1 - \sigma(x)$

2. [**5 points**] Compute $\sigma'(x)$. What does it simplify to? Write your final answer only in terms of $\sigma(x)$. Simplifying is an important part of this problem, because this simplified form is very helpful to know!

We know that $\sigma(x) = (1 + e^{-x})^{-1}$
Therefore,
$\sigma'(x) = -(1 + e^{-x})^{-2} \cdot e^{-x} \cdot (-1)$
$= (1 + e^{-x})^{-2} \cdot e^{-x}$
$= (1 + e^{-x})^{-1} \cdot (1 + e^{-x})^{-1} \cdot e^{-x}$
$= \frac{1}{1+e^{-x}} \cdot \frac{1+e^{-x}-1}{1+e^{-x}}$
$= \frac{1}{1+e^{-x}} \cdot (1 - \frac{1}{1+e^{-x}})$
$= \sigma(x) \cdot (1 - \sigma(x))$

3. [**7 points**] This question focuses on the following: in logistic regression, we sometimes use the relabeling $\{-1, 1\}$ instead of $\{0, 1\}$. We want you to be comfortable seeing both forms and moving fluidly between them.

   When our class labels are in $\{0, 1\}$, we can cleverly express $P(Y = y|X = x; \theta)$ as

   $$P(Y = 1|X = x; \theta)^y \cdot P(Y = 0|X = x; \theta)^{1-y}.$$

   This is the conditional probability for a *single* example.

   It yields the following conditional *negative* log likelihood over an n-item dataset:

   $$\ell(\theta) = \sum_{i=1}^{n} \left(-\log \left[P(Y_i = 1|X_i = x_i; \theta)^{y_i} \cdot P(Y_i = 0|X_i = x_i; \theta)^{1-y_i} \right] \right). \quad (1)$$

   (Think about what standard, but important, independence assumptions were needed to express the joint conditional probability over the full n-item dataset as a product of

per-i probabilities. You do not have to write anything down for this, but it's important to think about it and not completely take it for granted.)

a. Simplify the ith term of (1) as much as possible when $y_i = 1$. Recall that, in logistic regression, we model $P(Y_i = 1|X_i = x_i; \theta)$, where $\theta = (w, b)$, as $\sigma(w^T x_i + b)$. Your answer should be in terms of x_i, w, b.

The ith term is $-\log((\sigma(w^T x_i + b))^1 * 1) = -\log(\sigma(w^T x_i + b))$
$= -\log((1 + e^{-(w^T x_i + b)})^{-1})$
$= \log(1 + e^{-(w^T x_i + b)})$

b. Simplify the ith term of (1) as much as possible when $y_i = 0$. Recall that, in logistic regression, we model $P(Y_i = 1|X_i = x_i; \theta)$, where $\theta = (w, b)$, as $\sigma(w^T x_i + b)$. Your answer should be in terms of x_i, w, b.

The ith term is $-\log(1 * (1 - \sigma(w^T x_i + b))^1) = -\log(1 - \sigma(w^T x_i + b))$
$= -\log(\sigma(-(w^T x_i + b)))$
$= -\log((1 + e^{w^T x_i + b})^{-1})$
$= \log(1 + e^{w^T x_i + b})$

c. Suppose you mapped all appearances of label 0 to label -1 and retained label 1 as is. In light of (a) and (b), what does (1) simplify to? Maximum simplification is part of the credit here. Your answer should be in terms of x_i, y_i, w, b.

$P(Y_i = 1|X_i = x_i; \theta) = \sigma(w^T x_i + b)$
$P(Y_i = -1|X_i = x_i; \theta) = 1 - P(Y_i = 1|X_i = x_i; \theta) = 1 - \sigma(w^T x_i + b) = \sigma(-(w^T x_i + b))$
Therefore, we can let $P(Y_i = y_i|X_i = x_i; \theta) = \sigma(y_i \cdot (w^T x_i + b))$
Thus, $\ell(\theta) = \sum_{i=1}^{n} \left(-\log\left[\sigma(y_i \cdot (w^T x_i + b))\right]\right)$
$= \sum_{i=1}^{n} -\log((1 + e^{-y_i \cdot (w^T x_i + b)})^{-1})$
$= \sum_{i=1}^{n} \log(1 + e^{-y_i \cdot (w^T x_i + b)})$

For your understanding, can you see why it might be desirable to relabel as $\{-1, 1\}$? You do not have to write/submit anything here. Just think about it.

d. If you relabel as $\{-1, 1\}$ and write $\ell(\theta)$ as in part (c), have you changed the value of $\ell(\theta)$ at all, compared to (1) with $\{0, 1\}$ labels? Justify.

No, we did not change the value of $\ell(\theta)$ at all.
Previously, when $y_i = 0$, the ith term of $\ell(\theta)$ is $-\log(1 - \sigma(w^T x_i + b))$
Now, when $y_i = -1$, the ith term of $\ell(\theta)$ is $-\log(\sigma(-(w^T x_i + b))) = -\log(1 - \sigma(w^T x_i + b))$
Since they are equal, the value of $\ell(\theta)$ does not change

4.2 Logistic Regression Implementation [30 points]

Implementation instructions.

1-Sentence Overview: You will be training a logistic regression model on the Homework 1 dataset by running gradient descent and then answering the written questions below.

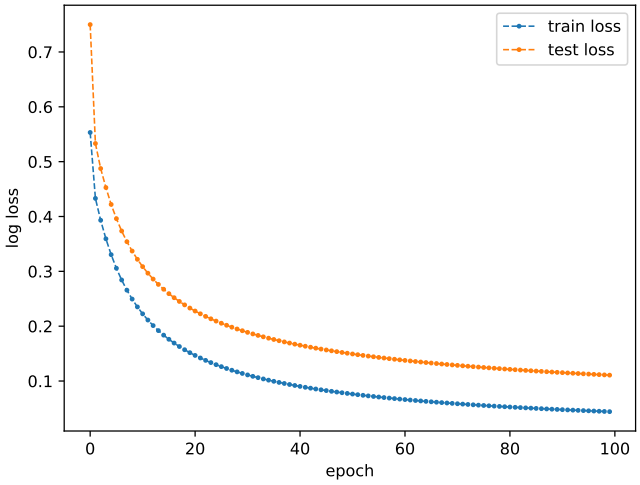
Details: You will fill in the `logistic_regression.py` template and submit your complete file to Gradescope, where we will run your code against a suite of tests. Your grade will be automatically determined from the testing results. Since you get immediate feedback after submitting your code and you are allowed to submit as many different versions as you like (without any penalty), it is 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.**

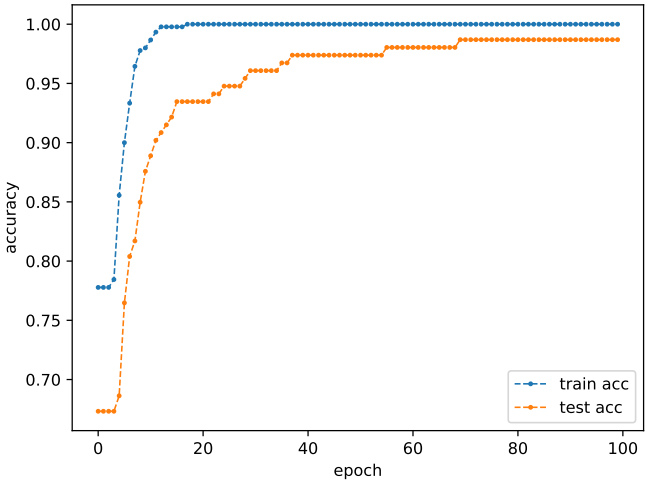
Everything to complete is clearly marked in `logistic_regression.py` with the identifier `##### ADD YOUR CODE HERE`. Extensive function descriptions are also provided in the code. Do not otherwise change the functionality of any of the provided code (though you are welcome to replace it if you can guarantee that the functionality won't change).

In this problem, **20 points will come from Gradescope's automated tests**, and **10 points will come from answering the following written questions** using your code.

[10 points] Using your code.

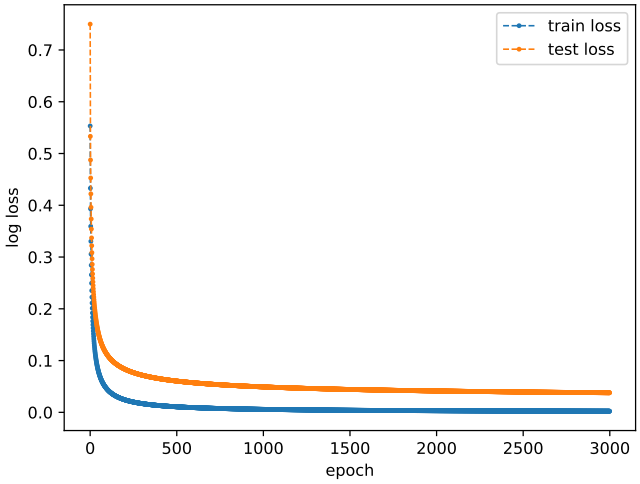
1. **[2.5 points]** Run gradient descent for 100 epochs with a learning rate of 0.1 (no explicit regularization). **a)** Plot train loss and test loss on one plot and **b.)** plot train accuracy and test accuracy on *another separate* plot. (PLEASE use the provided `_plot` function in the `logistic_regression.py` template.) **c.)** How does your final test accuracy compare to the Naive Bayes test accuracy in Homework 1?

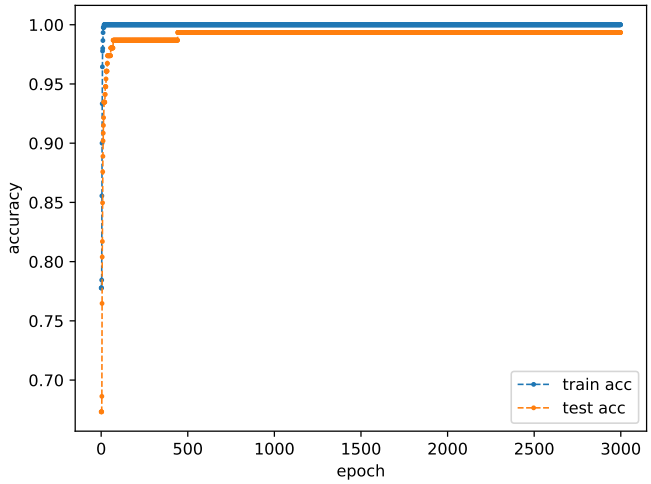




The final logistic regression test accuracy is 0.9869.
The final Naive Bayes test accuracy is 0.9477.
Therefore, logistic regression test accuracy is higher than the Naive Bayes test accuracy

2. [**2.5 points**] If you did this correctly, you might wonder if training longer could squeeze out a bit more test accuracy. Rerun for 3000 epochs and produce *new* versions of the **a.)** loss plot and **b.)** accuracy plot from the previous part. Our implementation took approx. 3 minutes; if yours is taking considerably longer (on your machine), you probably haven't sufficiently vectorized your code. **c.)** Did your test accuracy improve?





The final test accuracy is 0.9935, slightly better than the test accuracy we got from training for 100 epoches.

3. **[5 points]** We're now going to study what happens when we replicate features (specifically, we'll replicate 1 feature many times). We'll compare logistic regression to naive Bayes from Homework 1. **For the naive Bayes components, you are encouraged/expected to use your code from Homework 1; you do not need to resubmit the naive Bayes code.** 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 450×10 and our test data matrix will now be 153×10. 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).

   (a) **[2 points]** Baseline values (on this 10-feature dataset)

      i. 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.

      Train Acc: 0.8333
      Test Acc: 0.7974

      ii. 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.

      Train Acc: 0.8422
      Test Acc: 0.8039

   (b) **[2 points]** Now add 500 repeats of the feature at index 4 (0-indexed). Your train data matrix will be 450×510 and your test data matrix will be 153×510.

      i. 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.

      Train Acc: 0.7689
      Test Acc: 0.6405

      ii. 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.

Train Acc: 0.8422
Test Acc: 0.8039

(c) [**1 point**] What do you notice about naive Bayes vs. logistic regression? Please just write 1 sentence.

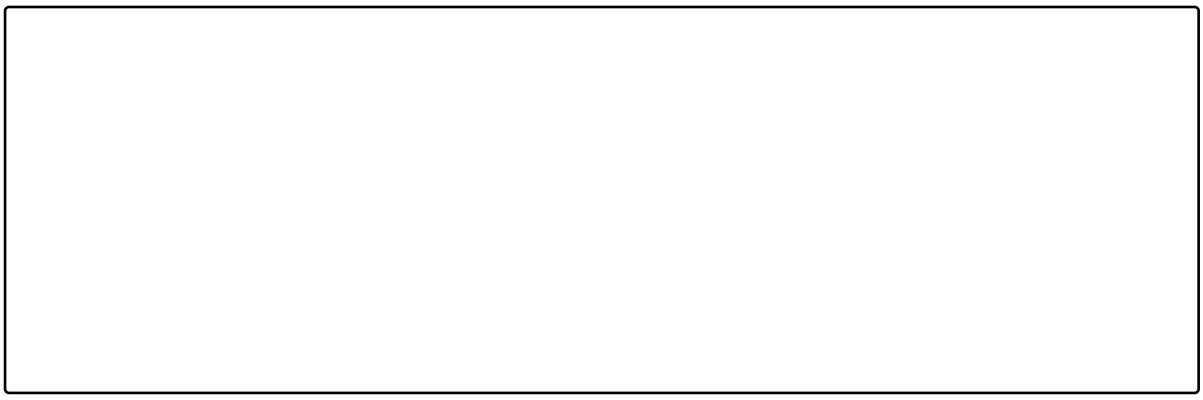
The performance of Naive Bayes is negatively impacted by repeated features, while the performance of logistic regression is unaffected.

Collaboration Questions Please answer the following:

1. Did you receive any help whatsoever from anyone in solving this assignment?
   Yes / **No**.
   - If you answered 'yes', give full details: _____
   - (e.g. "Jane Doe explained to me what is asked in Question 3.4")

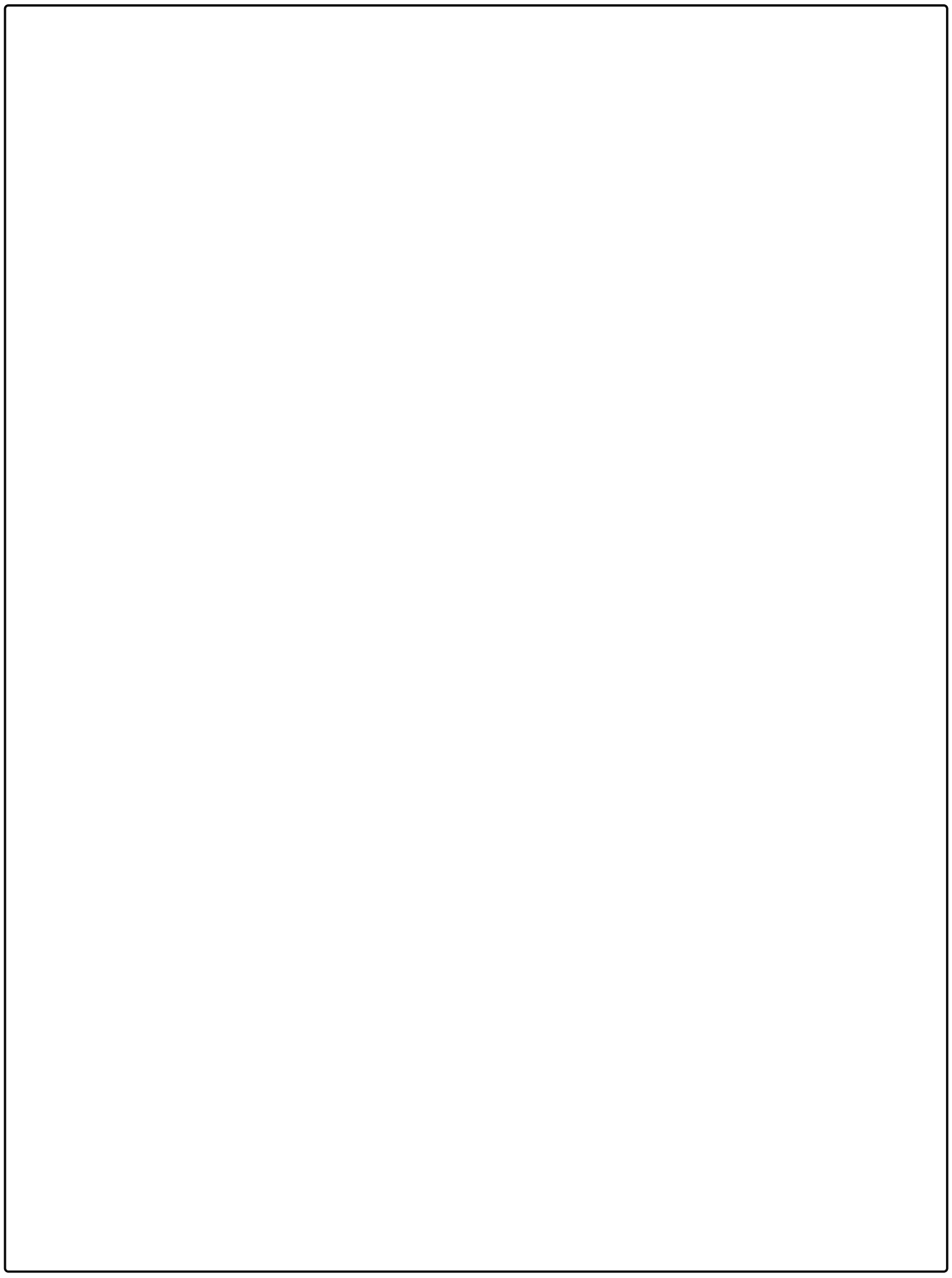
2. Did you give any help whatsoever to anyone in solving this assignment?
   Yes / **No**.
   - If you answered 'yes', give full details: _____
   - (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. Did you find or come across code that implements any part of this assignment ?
   Yes / **No**. (See below policy on "found code")
   - If you answered 'yes', give full details: _____
   - (book & page, URL & location within the page, etc.).

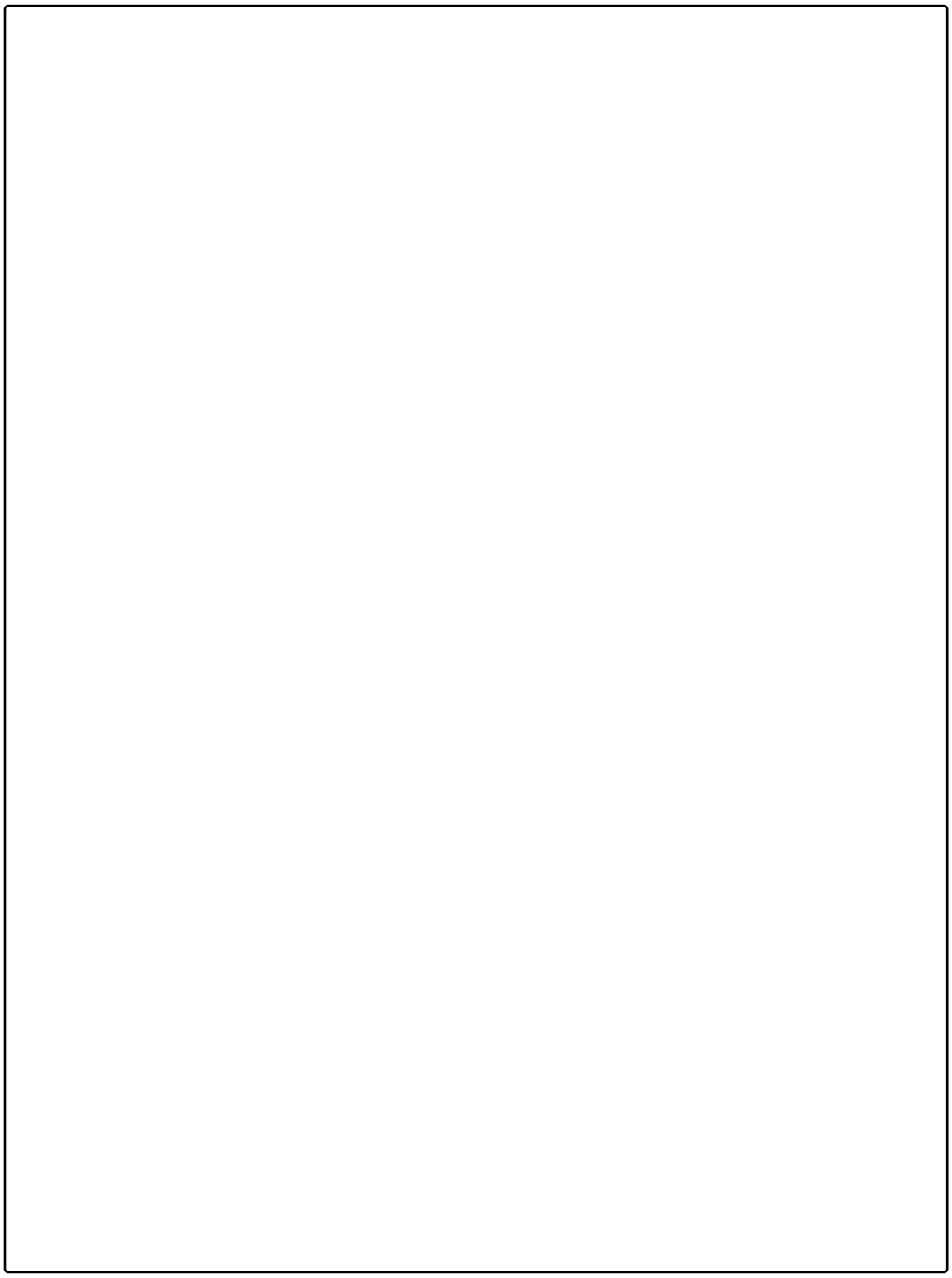


Overflow boxes

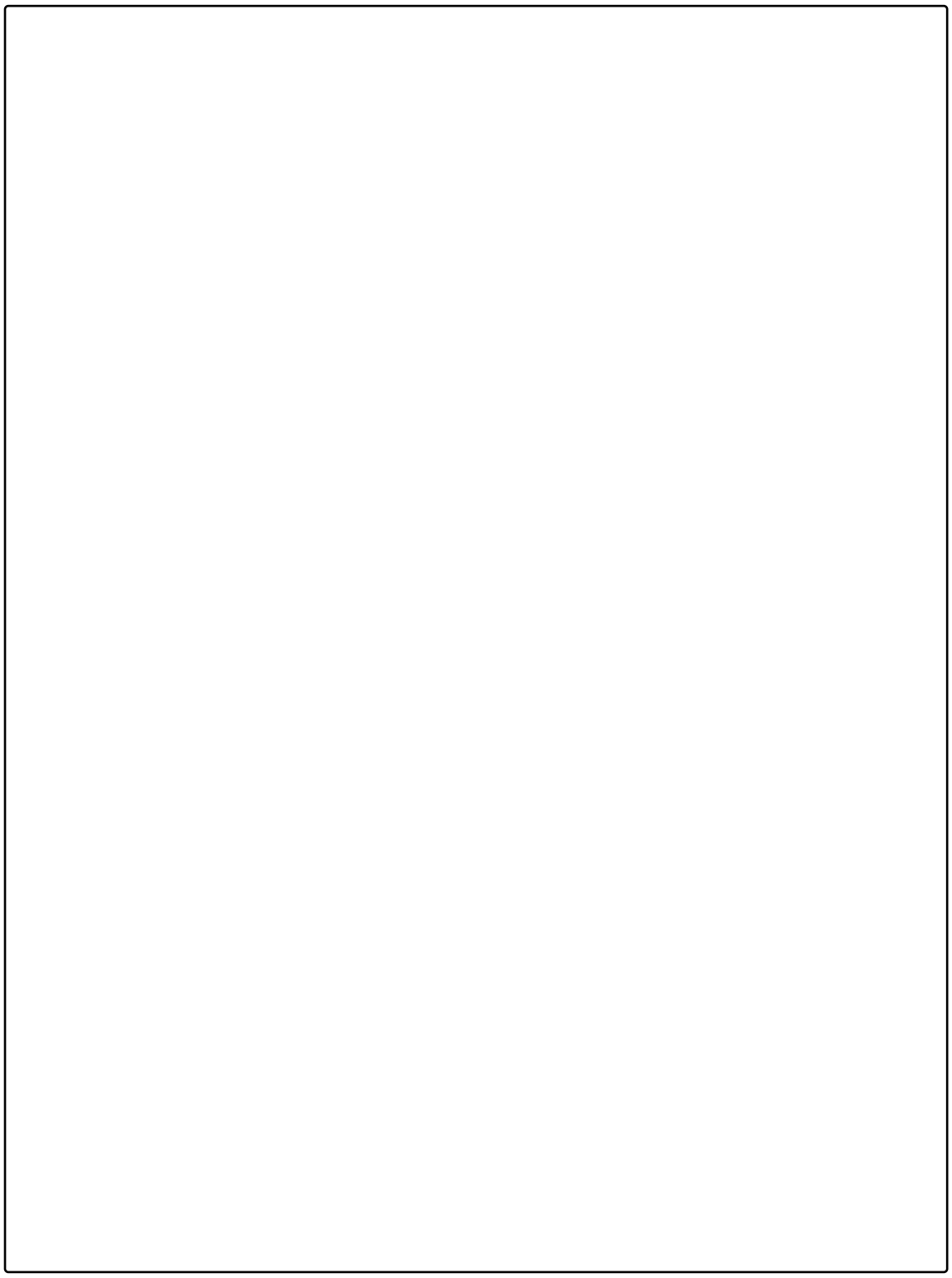
Section 1



Section 2



Section 3



Section 4