# Homework 2

## Decision Trees, Linear Regression, Logistic Regression[1]

CMU 10-701: Machine Learning (Fall 2021)

piazza.com/cmu/fall2021/10701/home
OUT: Wednesday , Sep 22nd, 2021

**DUE: Wednesday, Oct 6th, 2021, 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://piazza.com/class_profile/get_resource/ksetdrgdkob78/ksqc9bxxjt56ic
- **Late Submission Policy:** See the late submission policy here: https://piazza.com/class_profile/get_resource/ksetdrgdkob78/ksqc9bxxjt56ic
- **Submitting your work:**
  - **Gradescope:** There will be two submission slots for this homework on Gradescope: Written and 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

[1]Compiled on Wednesday 22nd September, 2021 at 18:13

rest of the solution'.

You are also required to upload your code, which you wrote to solve the final question 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 LaTeXusers, use ■ and ●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 Nearest Neighbors and Decision Trees [12 Points]

Let us try and classify data points in 2D Euclidean space. We are given $n$ instances of such points: $P_1, P_2, \ldots, P_n$ and the corresponding category for each point $C_1, C_2, \ldots, C_n$ (where $C_1, C_2, \ldots, C_n$ take values from the set of all possible class labels). Under the $k$ nearest neighbors classification scheme, each new element $Q$ is simply categorized by a majority vote among its $k$ nearest neighbors in instance space. The 1-NN is a simple variant of this which divides up the input space for classification purposes into a convex region (see Figure 1 below for the 1-NN decision boundaries under the Euclidean distance measure), each corresponding to a point in the instance set.

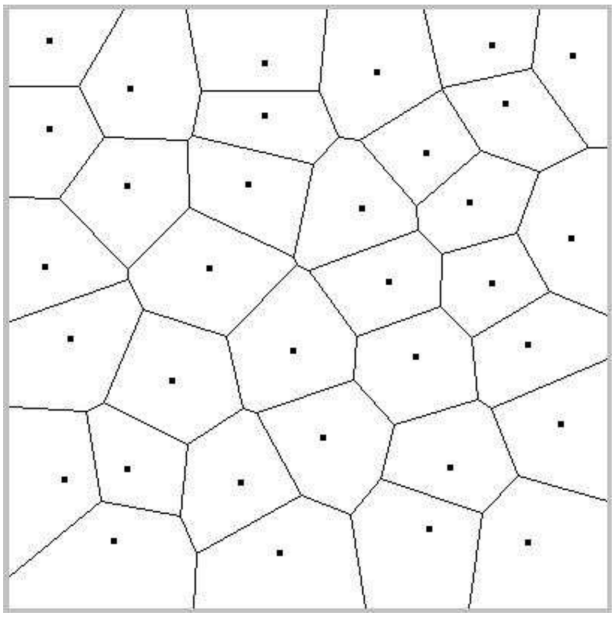
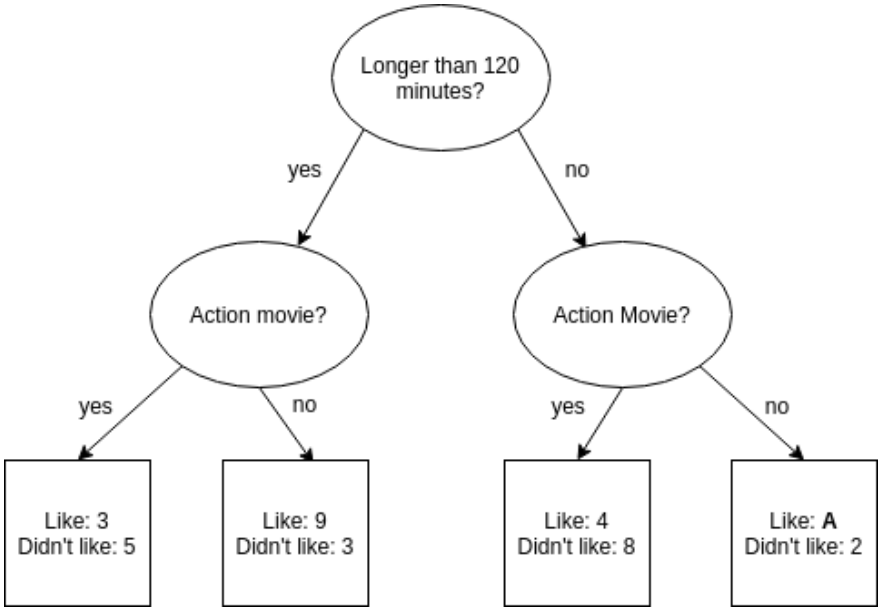


Figure 1: 1-NN decision boundaries under the Euclidean distance measure.

1. [**2 Points**] Is it possible to build a decision tree (with decisions at each node of the form "is $x > a$", "is $x < b$", "is $y > c$", or "is $y < d$" for any real constants $a, b, c, d$) which classifies exactly according to the 1-NN scheme using the Euclidean distance measure? If so, explain how. If not, explain why not.

   No, because the boundaries in figure 1 are not horizontal or vertical to x or y-axis, we cannot build a decision tree using just constants a,b,c,d for lines with slope not equal to 1 or 0.

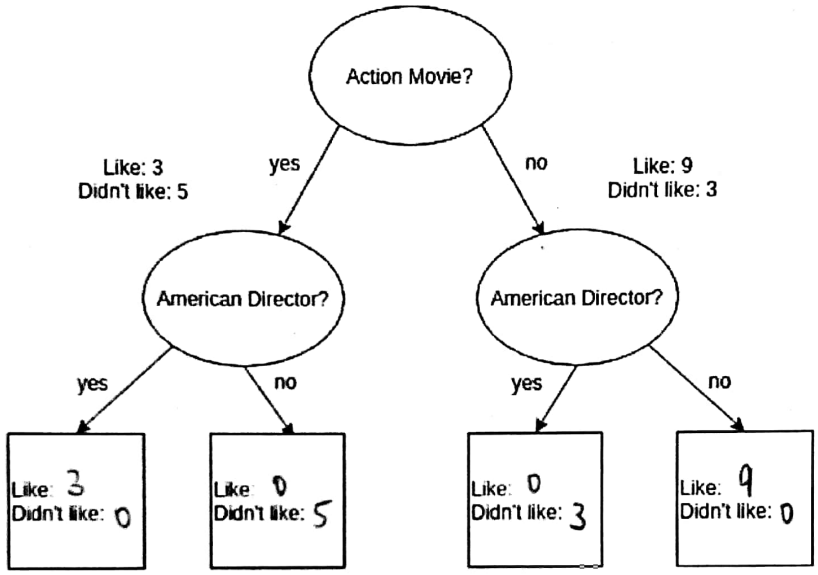
2. [**6 Points**] The following figure presents the top two levels of a decision tree learned to predict the attractiveness of a movie. What should be the value of A if the decision tree was learned using the algorithm discussed in class (you can either say 'At most X' or 'At least X' or 'Equal to X' where you should replace X with a number based on your calculation). **Explain your answer?**




Based on the algorithm, the first question (longer than 120 minutes?) should have the highest IG comparing to the second question (Action movie?).
IG(Like|Length)=H(Like)-H(Like|Length)
=(-P(Like)log2P(Like)-P(DLike)log2P(DLike)) - (P(Long)H(Like|Long)+P(Short)H(Like|Short))
IG(Like|ActMov)=H(Like)-H(Like|ActionMovie)
=(-P(Like)log2P(Like)-P(DLike)log2P(DLike))
- (P(ActMov=Yes)H(Like|ActMov=Yes)+P(ActMov=No)H(Like|ActMov=No))
Check the two IG values for A = 0, 1, ..., 50 and found that
when A=0, IG(Like|Length) > IG(Like|ActMov) (See below for the values)

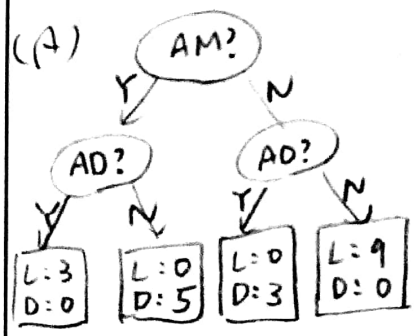
| A | IG(Like\|Length) | IG(Like\|ActMov) | |
|---|---|---|---|
| 0 | 0.0710 | 0.0609 | |
| 1 | 0.0510 | 0.0721 | |
| 2 | 0.0364 | 0.0828 | |
| 3 | 0.0256 | 0.0930 | |
| 4 | 0.0175 | 0.1026 | |
| 5 | 0.0116 | 0.1116 | |
| 6 | 0.0073 | 0.1201 | A=0 |

3. **[4 Points]** We now focus on all samples assigned to the left side of the tree (i.e. those that are longer than 120 minutes). We know that we have a binary feature, 'American director' that after the 'Action movie' split provides a perfect split for the data (i.e. all samples on one side are 'like' and all those on the other side 'didn't like'. Fill in the missing values in the picture below (there are multiple correct answers, just choose one):

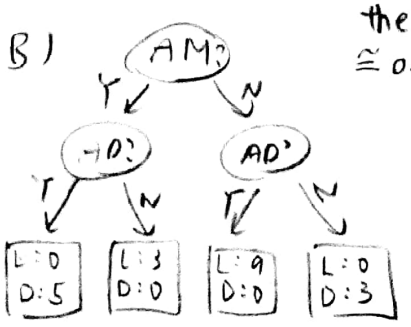



American Director perfect split, but can only has IG lower than Action Movie, so can't be all "like" or "Didn't like" for both (0.1024)
American Director = Yes.
The answer should be either (A) or (B) below, and the IG(like|AD) for both are the same! ≅ 0.0128

(A)




(B)



## 2 Relative Entropy and Mutual Information [17 Points]

We define **relative entropy** between two discrete distributions $\mathbf{p} \in \{p_1, \cdots, p_n\}$ and $\mathbf{q} = \{q_1, \cdots, q_n\}$ as:

$$D(\mathbf{p}\|\mathbf{q}) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

The above quantity is equal to 0 if and only if $\mathbf{p} = \mathbf{q}$. Otherwise, this quantity is positive. We can think of **relative entropy** as a measure of "distance" between two distributions. This quantity is widely known as the **Kullback-Leibler (KL) divergence** between two distributions (you can assume that none of the $q_i$ and $p_i$ are 0).

1. **[5 Points]** Prove that the **KL divergence** is always non-negative. Show all steps of your work
   *Hint:* You may use this inequality without proof: $x - 1 \geq \log(x)$, where $x \in \mathbb{R}$ with equality if and only if $x = 1$.

$$D(p\|q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

$$\xrightarrow{(-1)} -D(p\|q) = -\sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

$$= -\sum_{i=1}^{n} p_i (\log p_i - \log q_i)$$

$$= \sum_{i=1}^{n} p_i (\log q_i - \log p_i)$$

$$= \sum_{i=1}^{n} p_i \log \frac{q_i}{p_i} \leq \sum_{i=1}^{n} p_i \left(\frac{q_i}{p_i} - 1\right)$$

$$x$$

$$= \sum_{i=1}^{n} p_i \times \frac{q_i}{p_i} - \sum_{i=1}^{n} p_i$$

$$= \sum_{i=1}^{n} q_i - \sum_{i=1}^{n} p_i = 1 - 1 = 0$$

$$\Rightarrow -\sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \leq 0 \Rightarrow \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \geq 0 \quad \ast$$

2. **[5 Points]** True or False: **KL-divergence** is symmetric (i.e. $D(\mathbf{p}\|\mathbf{q}) = D(\mathbf{q}\|\mathbf{p})$). If you think the statement is true, please write down a sketch of proof. Otherwise, give a counterexample. Show all steps of your work

False, counter example:

$P \in \{0.2, 0.8\}$

$q \in \{0.6, 0.4\}$

$$D(P\|q) = \sum_{i=1}^{n} P_i \log \frac{P_i}{q_i} = 0.2 \log \frac{0.2}{0.6} + 0.8 \log \frac{0.8}{0.4} \cong 0.1454$$

$$D(q\|P) = \sum_{i=1}^{n} q_i \log \frac{q_i}{P_i} = 0.6 \log \frac{0.6}{0.2} + 0.4 \log \frac{0.4}{0.8} \cong 0.1659$$

$\hookrightarrow D(P\|q) \neq D(q\|P)$ ✱

3. **[7 Points]** Let $X$ and $Y$ be random variables taking the values $\{1, \cdots, n\}$. Recall the definition of mutual information between $X$ and $Y$:

$$I(X;Y) = H(Y) - H(Y|X)$$

Using the conclusion of part 1), show that the mutual information is always non-negative. Show all steps of your work.

$$H(Y) = -\sum_{j=1}^{n} P(y) \log P(y)$$

$$= -\sum_{j=1}^{n} \sum_{i=1}^{n} P(x, y) \log P(y)$$

$$H(Y|X) = \sum_{i=1}^{n} P(X=i) H(Y \mid X=i)$$

$$= \sum_{i=1}^{n} P(X=i) \times \left( -\sum_{j=1}^{n} P(Y=j \mid X=i) \log_2 P(Y=j \mid X=i) \right)$$

$$= -\sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i) P(Y=j|X=i) \log_2 P(Y=j \mid X=i)$$

$$= -\sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i, Y=j) \log_2 P(Y=j \mid X=i)$$

$$I(X;Y) = H(Y) - H(Y|X)$$

$$= -\sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i, Y=j) \log_2 P(Y=j) + \sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i, Y=j) \log_2 P(Y=j \mid X=i)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i, Y=j) \left[ \log_2 P(Y=j \mid X=i) - \log_2 P(Y=j) \right]$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i, Y=j) \log_2 \frac{P(Y=j \mid X=i)}{P(Y=j)}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i, Y=j) \log_2 \frac{P(Y=j \mid X=i) P(X=i)}{P(Y=j) P(X=i)}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} P(X=i, Y=j) \log_2 \frac{P(Y=j, X=i)}{P(Y=j) P(X=i)}$$

$$= D\left( p(X=i, Y=j) \,||\, p(X=i) P(Y=j) \right)$$

Based on the conclusion of part 1) $= D(p \,||\, q) \geq 0$

$$I(X;Y) = D\left( p(X=i, Y=j) \,||\, P(X=i) P(Y=j) \right) \geq 0 \quad \text{※}$$

# 3 Regularized Linear Regression Using Lasso [14 Points]

Lasso is a form of regularized linear regression, where the L1 norm of the parameter vector is penalized. It is used in an attempt to get a sparse parameter vector where features of little "importance" are assigned to zero weight. But why does lasso encourage sparse parameters? For this question, you are going to examine this.

Let $\mathbf{X}$ denote an $n \times d$ matrix where rows are training points, $\mathbf{y}$ denotes an $n \times 1$ vector of corresponding output value, $\mathbf{w}$ denotes a $d \times 1$ parameter vector and $\mathbf{w}^\star$ denotes the optimal parameter vector. To make the analysis easier we will consider the special case where the training data is whitened (i.e., $X^\top X = I$). For lasso regression, the optimal parameter vector is given by

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}}\, J_\lambda(\mathbf{w}), \tag{1}$$

where $J_\lambda(\mathbf{w})$ is the function we want to minimize, which is given by

$$J_\lambda(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1, \tag{2}$$

where $\lambda > 0$. Note that the $L_1$ norm for a vector $\mathbf{w} = [w_1, \ldots, w_d]^\top \in \mathbb{R}^d$ is defined as $\|\mathbf{w}\|_1 = |w_1\| + \ldots + |w_d|$.

1. **[3 Points]** In 3.2 and 3.3, we will show that whitening the training data nicely decouples the features, making $\mathbf{w}_i^\star$ determined by the $i$th feature and the output regardless of other features. To show this, begin by writing $J_\lambda(\mathbf{w})$ in the form

$$J_\lambda(\mathbf{w}) = g(\mathbf{y}) + \sum_{i=1}^{d} f\left(X_{\cdot i}, \mathbf{y}, w_i, \lambda\right), \tag{3}$$

where $X_{\cdot i}$ is the $i$th column of $X$, $g$ is a function of only $\mathbf{y}$ and $f$ is a function of $X_{\cdot i}, \mathbf{y}, w_i, \lambda$

$$\begin{aligned} J_\lambda(w) &= \tfrac{1}{2}(y - Xw)^T(y - Xw) + \lambda \sum_{i=1}^{d} |w_i| \\ &= \tfrac{1}{2}\left[y^Ty - y^T(Xw) - (Xw)^Ty + (Xw)^T(Xw)\right] + \lambda \sum_{i=1}^{d} |w_i| \\ &= \tfrac{1}{2}(y^Ty) + \tfrac{1}{2}(Xw)^T(Xw) - y^T(Xw) + \lambda \sum_{i=1}^{d} |w_i| \\ &= \underbrace{\tfrac{1}{2}(y^Ty)}_{g(y)} + \sum_{i=1}^{d} \underbrace{\left[\tfrac{1}{2}(X_{\cdot i}w_i)^T(X_{\cdot i}w_i) - y^T(X_{\cdot i}w_i) + \lambda|w_i|\right]}_{f(X_{\cdot i}, y, w_i, \lambda)} \end{aligned}$$

2. **[3 Points]** Assume that $w_i^* > 0$, what is the value of $w_i^*$ in this case?

$$\frac{d}{dw_i} J_\lambda(w_i) = \frac{1}{2} \cdot 2 \underbrace{X_i^T X_i}_{I} w_i - y^T X_i + \lambda = 0$$

$$\Rightarrow w_i^* = y^T X_i - \lambda \quad \#$$

3. **[3 Points]** Assume that $w_i^* < 0$, what is the value of $w_i^*$ in this case?

$$\frac{d}{dw_i} J_\lambda(w_i) = X_i^T X_i w_i - y^T X_i - \lambda = 0$$

$$\Rightarrow w_i^* = y^T X_i + \lambda \quad \#$$

4. **[3 Points]** From 2 and 3, what is the condition for $w_i^*$ to be 0? How can you interpret that condition?

If $w_i^* = 0$

$\Rightarrow \frac{d}{dw_i} J_\lambda(w_i)$ at $w_i = 0$ can be $+1$ or $-1$

$\Rightarrow y^T X_i + \lambda (\text{sgn } w_i) = 0$, $\text{sgn } w_i \in [1, -1]$

$\Rightarrow |y^T X_i| = \lambda$

5. **[2 Points]** Now consider ridge regression where the regularization term is replaced by $\frac{1}{2}\lambda\|\mathbf{w}\|_2^2$. What is the condition for $w_i^* = 0$? How does it differ from the condition you obtained in 4?

$$J_\lambda(w) = \frac{1}{2}\|y - Xw\|_2^2 + \frac{1}{2}\lambda\|w\|_2^2$$

Based on Q3.1

$$J_\lambda(w) = \frac{1}{2}(y^T y) + \sum_{i=1}^{d}\left[\frac{1}{2}(X_{\cdot i}\, w_i)^T(X_{\cdot i}\, w_i) - y^T(X_{\cdot i}\, w_i) + \frac{1}{2}\lambda w_i^2\right]$$

$$\frac{d}{dw_i} J_\lambda(w) = w_i - y^T X_{\cdot i} + \lambda w_i = 0$$

$$\Rightarrow w_i^*(1+\lambda) = y^T X_{\cdot i}$$

$$\Rightarrow w_i^* = \frac{y^T X_{\cdot i}}{1+\lambda}$$

for the condition $w_i^* = 0 \Rightarrow y^T X_{\cdot i} = 0$

→ the new $J_\lambda(w)$ is differentiable at $w_i^* = 0$

Not like in Q4 the regularization term $\lambda\|w\|_1$ was not differentiable and need to take subderivatives.

# 4 Logistic Regression; Improving our understanding of Convexity [25 points]

Consider a binary classification problem where the goal is to predict a class $y \in \{0, 1\}$, given an input $x \in \mathcal{R}^p$. 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. Let's investigate why this is a reasonable choice:

(Affine function definition: 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.)

## 4.1 Setup

1. **[2 points]** For a probability value $p \in (0, 1)$, what is the range of the odds, $\frac{p}{1-p}$, and the log-odds, $\log\left(\frac{p}{1-p}\right)$? Explain why this makes the log-odds a desirable transformation of our data to fit with our affine model.

   $p=0 \to \frac{p}{1-p}=0$ ; $p=1 \to \frac{p}{1-p}=\infty \Rightarrow$ range $(\frac{p}{1-p})=(0, \infty)$

   $\log \frac{p}{1-p}=-\infty$ $\log \frac{p}{1-p}=\infty \Rightarrow$ range$(\log \frac{p}{1-p})=(-\infty, \infty)$

   The log-odds transformation makes the function covers $-\infty$ to $\infty$ of the affine model.

2. **[2 points]** We can proceed to model the log-odds with an affine model:

$$\log \frac{P(y_i = 1|x_i, w)}{1 - P(y_i = 1|x_i, w)} = w^\top x_i.$$

   Conclude that the log-likelihood can be written as

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

where:

- $\mathbf{X} \in \mathcal{R}^{n \times (1+p)}$ is a data matrix, with the first column composed of all ones
- $w \in \mathcal{R}^{(p+1) \times 1}$ is the weight vector, with the first index $w_1$ acting as the bias term
- $x_i$ is a column vector of the $i^{th}$ row of $\mathbf{X}$
- $y \in \mathcal{R}^{n \times 1}$ is a column vector of labels $y_i \in \{0, 1\}$

- $p$ is the dimension of data (number of features in each observation)

$$\log \frac{P(y_i=1 \mid x_i, w)}{1 - P(y_i=1 \mid x_i, w)} = w^T x_i$$

$$\frac{P(y_i=1 \mid x_i, w)}{1 - P(y_i=1 \mid x_i, w)} = e^{w^T x_i} = \frac{\frac{e^{w^T x_i}}{1+e^{w^T x_i}}}{\frac{1}{1+e^{w^T x_i}}}$$

$$L(w) = \log P(y \mid X, w)$$

$$P(y \mid X, w) = \prod_{i=1}^{n} \left( P(y_i=1 \mid x_i, w)^{y_i} \left(1 - P(y_i=1 \mid x_i, w)\right)^{(1-y_i)} \right.$$

$$\Rightarrow \log P(y \mid X, w) = \sum_{i=1}^{n} \left[ y_i \log \left( P(y_i=1 \mid x_i, w) + (1-y_i) \log \left(1 - P(y_i=1 \mid x_i, w)\right)\right)\right]$$

$$= \sum_{i=1}^{n} \left[ y_i \log \frac{P(y_i=1 \mid x_i, w)}{1 - P(y_i=1 \mid x_i, w)} + \log \left(1 - P(y_i=1 \mid x_i, w)\right)\right]$$

$$= \sum_{i=1}^{n} \left[ y_i w^T x_i + \cancel{\log 1} - \log \left(1 + e^{w^T x_i}\right)\right]$$

$$= \sum_{i=1}^{n} \left[ y_i w^T x_i - \log \left(1 + e^{w^T x_i}\right)\right] \quad **$$

## 4.2 Convex Optimization

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.

1. **[3 points]** A real-valued function $f : S \to \mathcal{R}$ 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, $f_1, f_2, ..., f_n$, $\sum_{i=1}^{n} a_i f_i(x)$ is also a convex function $\forall a_i \in R^+$.

$$\sum_{i=1}^{n} a_i f_i(x) = \sum_{i=1}^{n} a_i f_i(t x_1 + (1-t) x_2)$$
$$= a_1 f_1(t x_1 + (1-t) x_2) + \cdots + a_n f_n(t x_1 + (1-t) x_2)$$
$$\xrightarrow{a_i \in R^+} \leq a_1 (t f_1(x_1) + (1-t) f_1(x_2)) + \cdots +$$
$$a_n (t f_n(x_1) + (1-t) f_n(x_2))$$
$$= t(a_1 f_1(x_1) + \cdots + a_n f_n(x_1))$$
$$+ (1-t)(a_1 f_1(x_2) + \cdots + a_n f_n(x_2))$$
$$= \sum_{i=1}^{n} a_i (t f_i(x_1) + (1-t) f_i(x_2))$$

$f_1, f_2, \ldots f_n$ are convex $\Rightarrow \sum_{i=1}^{n} f_i$ is convex

$a_i$ is nonnegative

$\sum_{i=1}^{n} a_i f_i(x)$ is also convex.

2. **[2 points]** Show that a linear combination of $n$ concave functions, $f_1, f_2, \ldots, f_n$, $\sum_{i=1}^{n} a_i f_i(x)$ 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))

$f(x)$ is convex when

$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$

$\xrightarrow{\times(-1)}$ $-f(tx_1 + (1-t)x_2) \geq -tf(x_1) - (1-t)f(x_2)$

Let $g(x) = -f(x)$ is concave

$\Rightarrow g(tx_1 + (1-t)x_2) \geq tg(x_1) + (1-t)g(x_2)$

$\Rightarrow g(x)$ is concave

$\Rightarrow$ the sum of n concave function with a nonnegative $a_i$ $\left(\sum_{i=1}^{n} a_i f_i(x)\right)$ is also concave.

3. **[4 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 )

$$f(x) = \log(1+e^x)$$

$$\xrightarrow{d} f'(x) = \frac{1}{1+e^x} \times e^x = e^x(1+e^x)^{-1}$$

$$\xrightarrow{d} f''(x) = e^x(1+e^x)^{-1} - (e^x)^2(1+e^x)^{-2}$$

$$= \frac{e^{2x}+e^x-e^{2x}}{(1+e^x)^2} = \frac{e^x}{(1+e^x)^2} \text{ (exists)}$$

$$\Rightarrow f''(x) = \frac{e^x}{(1+e^x)^2} \geq 0$$

$\Rightarrow f$ is convex //

4. **[4 points]** Let $f_i : \mathcal{S} \to \mathcal{R}$ for $i = 1, \ldots, n$ 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.

Yes

$f(tx_1 + (1-t)x_2) = f_i(tx_1 + (1-t)x_2)$ for $i = 1, \ldots, n$

$\because f_i$ is convex

$\Rightarrow f_i(tx_1 + (1-t)x_2) \leq t f_i(x_1) + (1-t) f_i(x_2)$

$\leq t \max_i f_i(x_1) + (1-t) \max_i f_i(x_2)$

$= t f(x_1) + (1-t) f(x_2)$

$\Rightarrow f(tx_1 + (1-t)x_2) \leq t f(x_1) + (1-t) f(x_2)$

$\Rightarrow f(x) = \max_i f_i(x)$ also convex

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.

$$LL(y|X;w) = \sum_{i=1}^{n} [\underbrace{y_i w^T X_i}_{①} \underbrace{- \ln(1+e^{w^T X_i})}_{②}]$$

① $y_i w^T X_i$ is an affine function

→ is both convex and concave

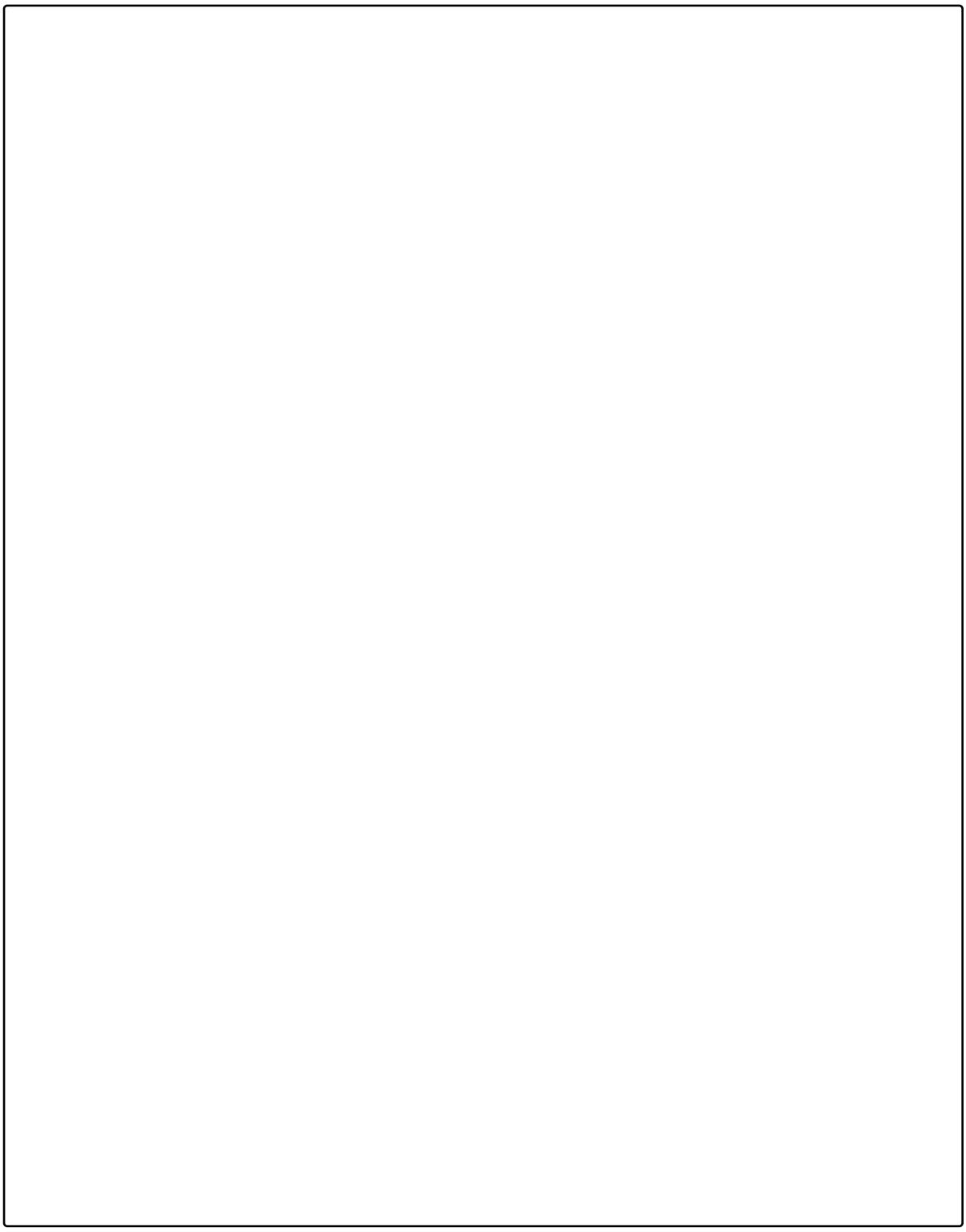
② from Q4.2.3, we proved that $f(x) = \log(1+e^x)$ is convex

→ $\ln(1+e^{w^T X_i})$ is also convex. (let $X$ be $w^T X_i$ here).

from Q4.2.2. $g = -f(x) = -\ln(1+e^{w^T X_i})$ is concave

⇒ $LL(y|X;w)$ combines the summation of two concave functions

⇒ is also concave.



# 5 Programming Linear Regression [32 Points]

**Note: Your code for all of the programming exercises including this one should be submitted to the corresponding programming submission slot on Gradescope. Feel free to use any programming language, as long as your TAs can read your code. Turn in your code in a single .tar ball that might contain multiple source code files along with a README which contains instructions on how to run your code and generate the results and plots.** Visualizations and written answers should still be submitted as a part of the rest of the homework in this PDF. In your code, **please use comments to point out primary functions that compute the answers to each question.**

In this problem you will implement linear regression with different regularization techniques and use stochastic gradient descent to learn the model parameters.

## 5.1 Preliminaries

Recall from class that Linear Regression assumes a linear relationship between the covariates $x_1, \ldots, x_m$ and the continuous response $y$. Mathematically, this can be written as:

$$y = \hat{y} + \epsilon = \sum_{j=1}^{m} w_j x_j + b + \epsilon$$

Here $\hat{y}$ represents the model prediction for the true $y$, $\{w_i\}_{i=1}^{m}$ and $b$ are the model parameters to be estimated and $\epsilon$ is a zero-mean random error term. We estimate the parameters by minimizing the following loss function:

$$\mathcal{L}(w_1, \ldots, w_m, b) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2$$

Here, the superscript $^{(i)}$ denotes a particular datapoint in our dataset and $\hat{y}^{(i)} = \sum_{j=1}^{m} w_j x_j^{(i)} + b$. Note that here we are assuming that we have $n$ points in our dataset with $m$ covariates. The optimal model parameters can then be written as the solution to this equation:

$$w_1^\star, \ldots, w_m^\star, b^\star = \underset{w_1,\ldots,w_m,b}{\operatorname{argmin}} \mathcal{L}(w_1, \ldots, w_m, b)$$

There are a number of ways to minimize the above loss function and find the optimal parameters. In this assignment, you will be implementing **stochastic gradient descent**.

## 5.2 Data preprocessing

Data preprocessing is a very important step in all ML algorithms including linear regression. You will be using the Carseats dataset, the details of which can be found at

https://rdrr.io/cran/ISLR/man/Carseats.html

You can find the train and test data in `carseats_train.csv` and `carseats_test.csv` files in the handout respectively. This dataset contains **10** covariates (`CompPrice, Income, Advertising, Population, Price, ShelveLoc, Age, Education, Urban, US`) and the response variable that is to be predicted is `Sales`. There are three important steps that you must do.

1. **Binary variable encoding**: In this dataset, `Urban, US` are both binary variables which take values `No` and `Yes`. You must convert them to 0/1 binary variables so that these numerical values can be used for linear regression.
2. **Categorical variable encoding**: `ShelveLoc` is a categorical variable that takes **three** values, namely `Bad, Good, Medium`. You must do **one-hot encoding** for this particular variable. This means that you must create three dummy variables: `ShelveLocBad, ShelveLocGood, ShelveLocMedium`. `ShelveLocBad` should be 1 when the value of `ShelveLoc` is `Bad` and 0 otherwise. `ShelveLocGood, ShelveLocMedium` should be encoded in a similar way. This means that for any datapoint, **exactly** one of `ShelveLocBad, ShelveLocGood, ShelveLocMedium` will take the value 1 and the other two will be 0.
3. **Feature standardization**: Feature standardization makes the data such that it has zero mean and unit variance. For every continuous covariate, you must subtract the mean and divide it by the standard deviation.

   $$\hat{x}_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

   where $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}$ and $\sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_j^{(i)} - \mu_j)^2}$.
   NOTE: You must do the feature standardization for the **test** set using the mean and standard deviation calculated from the **train** set.

## 5.3 Stochastic Gradient Descent

You will use stochastic gradient descent (SGD) to train your linear regression model. Stochastic gradient descent is a stochastic approximation to gradient descent in which the gradient of the loss function is replaced by an estimate from a single datapoint. Mathematically

$$\mathcal{L} \approx (\hat{y}^{(i)} - y^{(i)})^2$$

Recall the gradient descent update rule:

$$w_j \leftarrow w_j - \eta \frac{\partial \mathcal{L}}{\partial w_j}$$

After computing $\frac{\partial \mathcal{L}}{\partial w_j}$, this becomes

$$w_j \leftarrow w_j - 2\eta(\hat{y}^{(i)} - y^{(i)})x_j^{(i)}$$

Similarly for $b$:

$$b \leftarrow b - 2\eta(\hat{y}^{(i)} - y^{(i)})$$

NOTE: Make sure that the parameters ($w_1, ..., w_m$ and $b$) are updated *simultaneously*. **You might want to vectorize your code to make this easier**.

This is also a good time to learn about some jargon related to SGD. A **step** or a **training step** is one gradient update. An **epoch** is when one full cycle over the training data is completed, i.e., each datapoint has been seen once.

1. [**2 Point**] Suppose your training data consists of 1000 datapoints and you are using SGD to learn your model parameters. How many steps will the algorithm take in **2 epochs**?

   2000

**Note on reproducibility:** When you use Stochastic Gradient Descent in the wild, you must **randomly shuffle** the training data at the beginning of each epoch and then perform the gradient update rule for each datapoint (this is the reason why SGD is *stochastic*).
**In this assignment however, please make sure that you do *NOT* shuffle the data before every epoch and instead cycle through the datapoints in the order in which they're given in the .csv file so that your solution can be compared with the reference solution. This will remove the randomness from your experiments and produce a deterministic answer [2] in each run.**

## 5.4 Regularization

Regularization is an important technique to prevent overfitting and achieve better generalization. In this assignment you will implement $L_2$ and $L_1$ regularization (also known as ridge and lasso regression respectively).

### 5.4.1 Ridge regression

The loss for $L_2$ regularization can be written as

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2 + \lambda\sum_{j=1}^{m} w_j^2$$

Not that we do not regularize $b$. With the SGD approximation, this becomes

$$\mathcal{L} \approx (\hat{y}^{(i)} - y^{(i)})^2 + \lambda\sum_{j=1}^{m} w_j^2$$

[2] In practice you would want to set a (pseudo) random number generator seed to make your experiments reproducible. However, that is specific to the programming language and libraries that you use. So it won't be comparable to the reference solution in this case.

1. **[3 Points]** Write the stochastic gradient descent update rules for $w_j$ and $b$ for ridge regression.

$L \approx (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \sum_{j=1}^{n} w_j^2$

$= ((\sum w_j x_j^{(i)} + b) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} w_j^2$

$\frac{dL}{dw_j} = [2(\hat{y}^{(i)} - y^{(i)}) x_j + 2\lambda w_j]$

$\Rightarrow w_j \leftarrow w_j - \eta(2(\hat{y}^{(i)} - y^{(i)}) x_j + 2\lambda w_j)$

$\Rightarrow w_j \leftarrow (1 - 2\lambda\eta) w_j - 2\eta(\hat{y}^{(i)} - y^{(i)}) x_j$

$b \leftarrow b - \eta \times 2(\hat{y}^{(i)} - y^{(i)})$

### 5.4.2 Lasso regression

The loss for $L_1$ regularization can be written as

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \sum_{j=1}^{m} |w_j|$$

Note that we do not regularize $b$. With the SGD approximation, this becomes

$$\mathcal{L} \approx (\hat{y}^{(i)} - y^{(i)})^2 + \lambda \sum_{j=1}^{m} |w_j|$$

1. **[3 Points]** Write the stochastic gradient descent update rules for $w_j$ and $b$ for lasso regression.

if $w_j \geq 0$:

$w_j \leftarrow w_j - 2\eta(\hat{y}^{(i)} - y^{(i)}) x_j - \eta\lambda$

else:

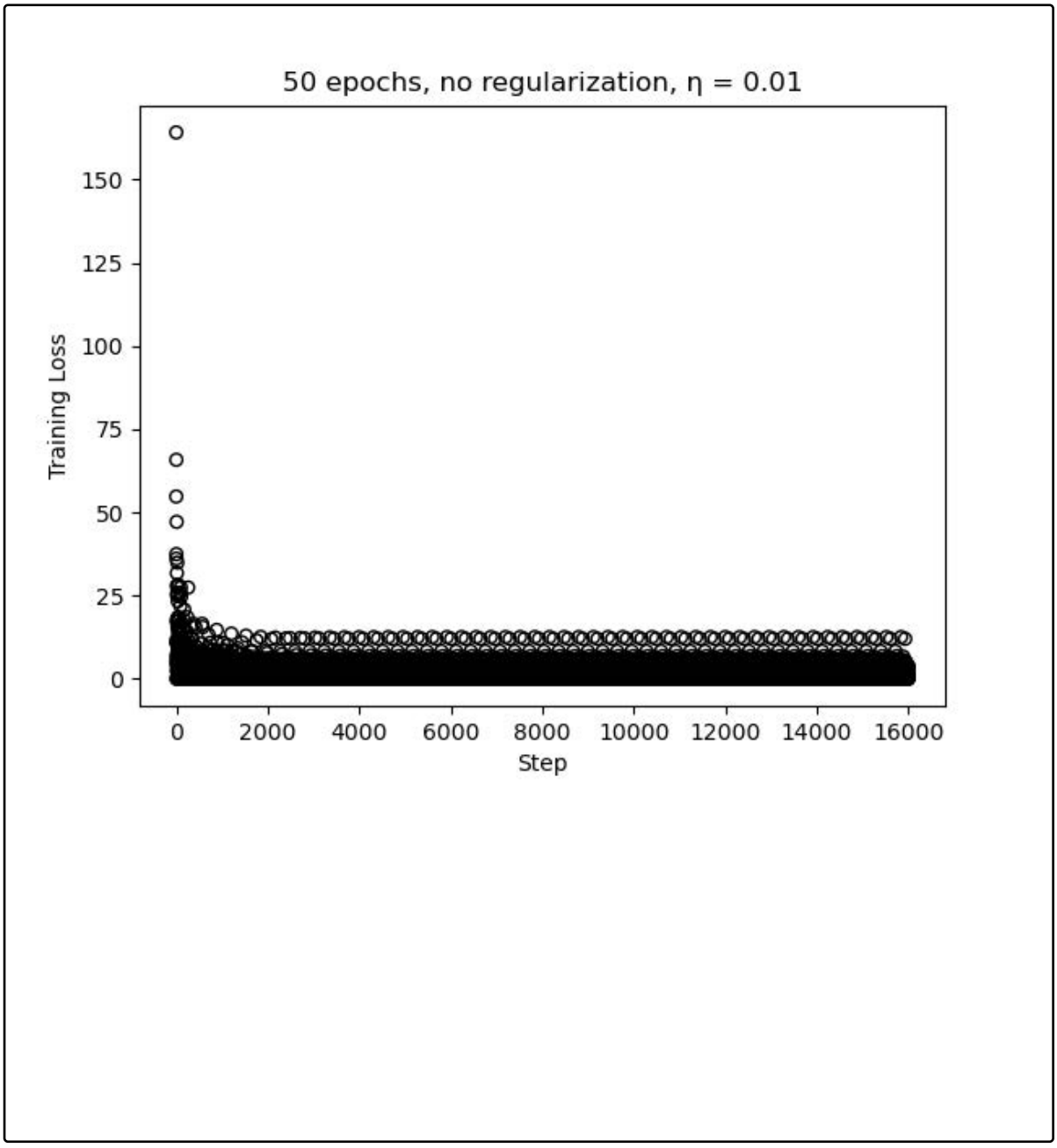
$w_j \leftarrow w_j - 2\eta(\hat{y}^{(i)} - y^{(i)}) x_j + \eta\lambda$

$b \leftarrow b - 2\eta(\hat{y}^{(i)} - y^{(i)})$

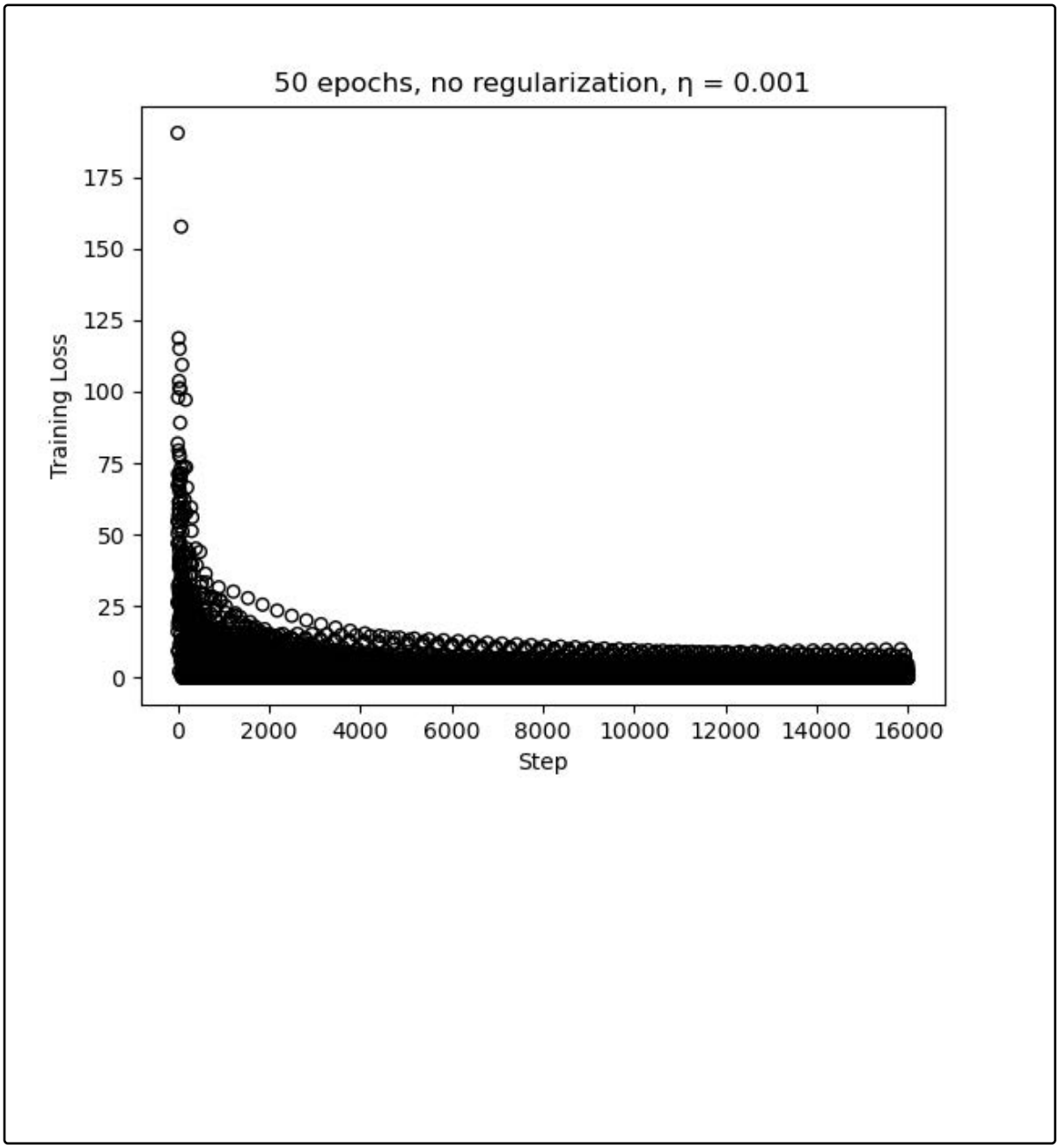
## 5.5 Training

Now is the time to actually train your linear model. Plot the loss curves for the following training configurations. Please note that you should plot the **training loss** computed at each step. Also note that the loss that you plot should be the SGD approximation to the real training loss computed using the particular datapoint for that step. **You must initialize the model parameters to zeros**. Make sure to clearly label the axes.

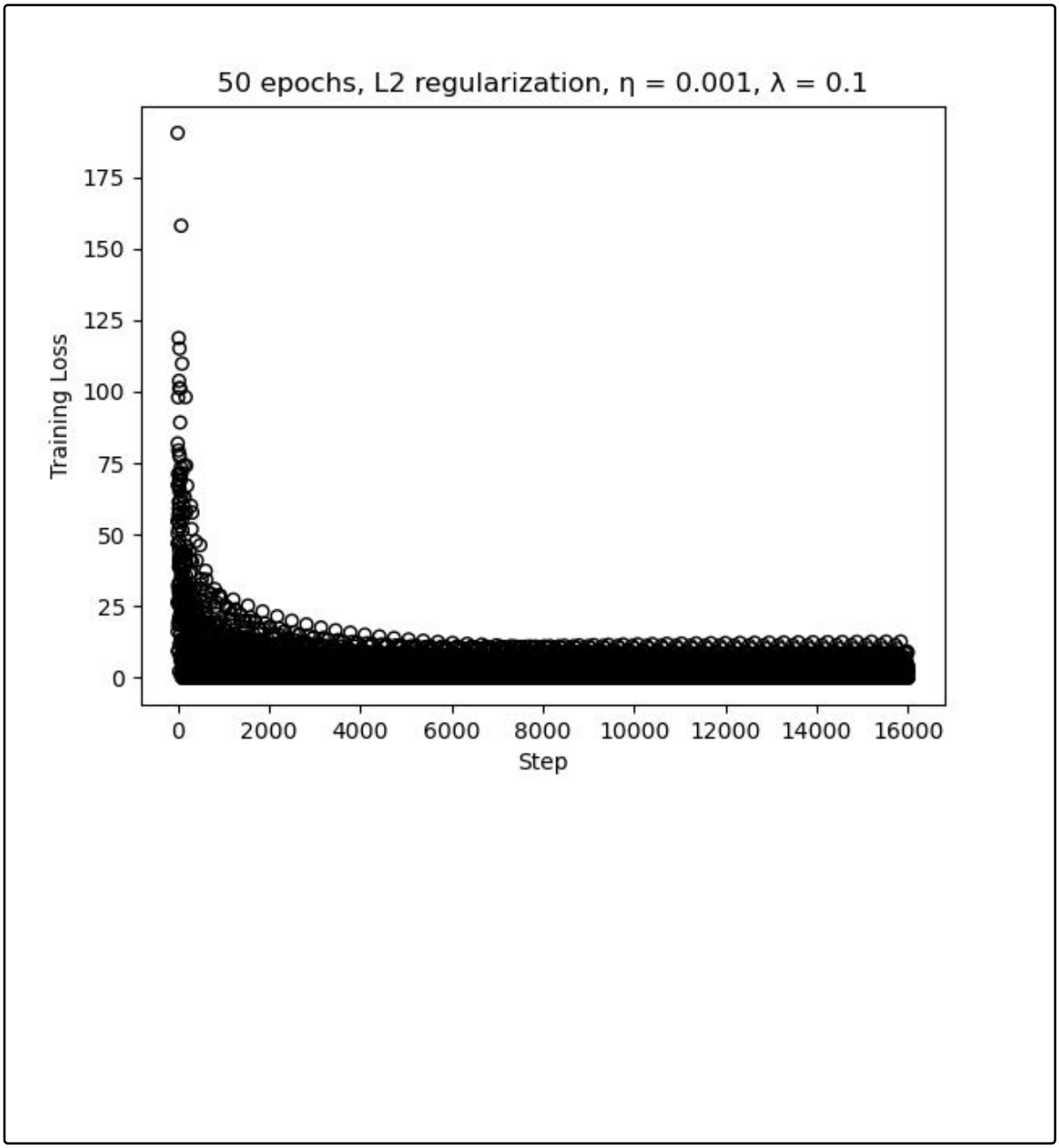
1. **[5 Points] 50 epochs, no regularization,** $\eta = 0.01$



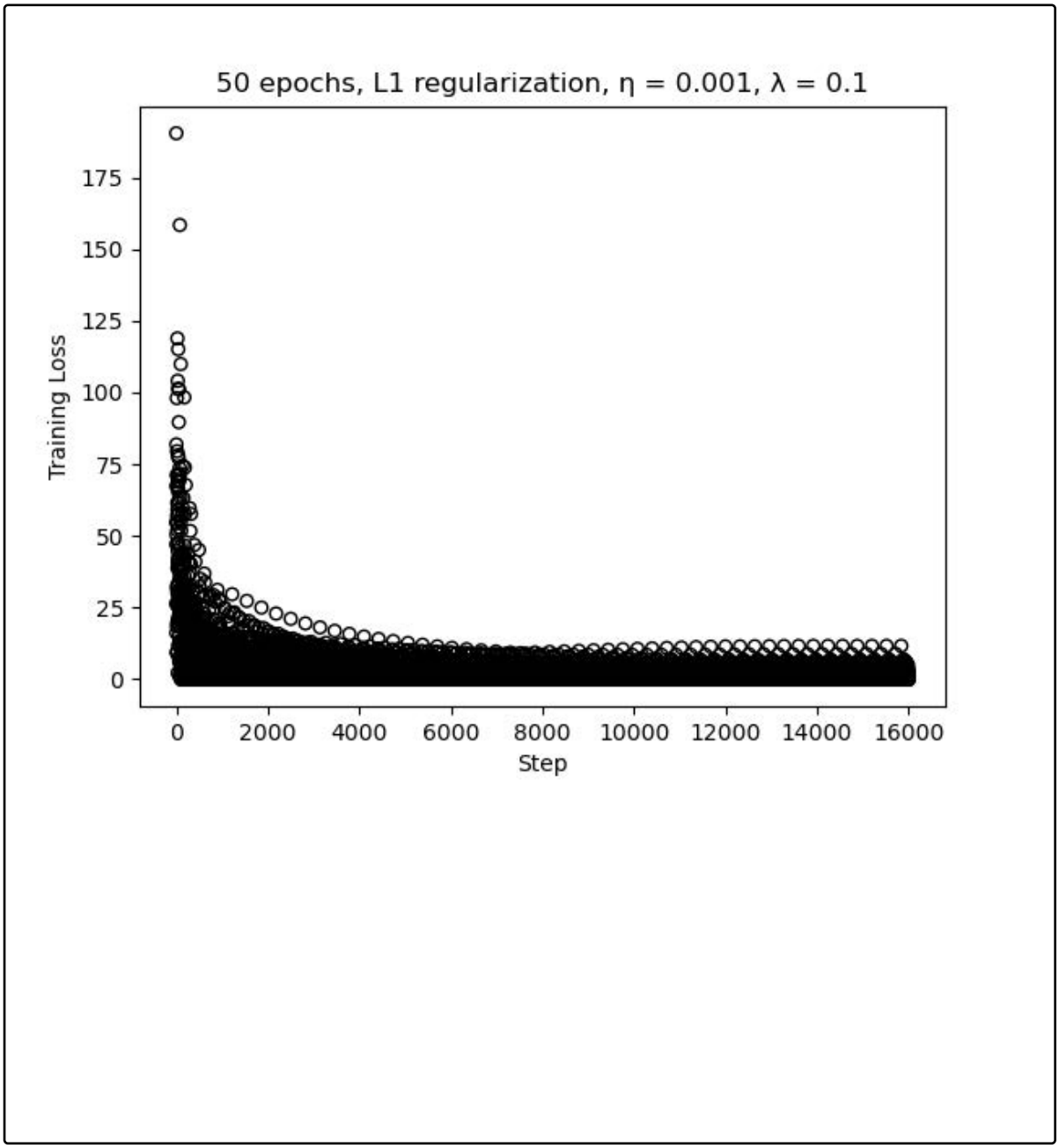
2. **[5 Points] 50 epochs, no regularization,** $\eta = 0.001$



3. **[5 Points] 50 epochs, $L_2$ regularization, $\eta = 0.001$, $\lambda = 0.1$**



4. **[5 Points] 50 epochs, $L_1$ regularization, $\eta = 0.001$, $\lambda = 0.1$**



## 5.6 Evaluation

You will now use your trained linear models to make predictions for datapoints not seen during training and evaluate the performance of your model. In the following questions, you have to compute the test loss which can be written as:

$$\mathcal{L}_{\text{test}} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (\hat{y}^{(i)} - y^{(i)})^2$$

Note that you are not supposed to add the $L_1$ or $L_2$ penalties when you are evaluating your model. For each of the following training configurations, report the test loss after training. Please report the loss values to **five** decimal places.

1. **[1 Point] 50 epochs, no regularization,** $\eta = 0.01$

   1.09765

2. **[1 Point] 50 epochs, no regularization,** $\eta = 0.001$

   1.06948

3. **[1 Point] 50 epochs,** $L_2$ **regularization,** $\eta = 0.001$**,** $\lambda = 0.1$

   1.74264

4. **[1 Point] 50 epochs,** $L_1$ **regularization,** $\eta = 0.001$**,** $\lambda = 0.1$

   1.12679

## 5.7 Code Submission Checklist

1. Did you add all your source code files in a single .tar ball?

   ● Yes

   ○ No

2. Did you include a README in the .tar ball which contains instructions on how to run your code and generate the different plots and evaluate the trained models?

   ● Yes

   ○ No

3. Did you add comments to point out primary functions that compute the answers to each question?

   ● Yes

   ○ No

# 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**

Yajushi Khurana and Swapnil Keshari explained to me
the question 3.1, 3.4, 3.5, 4.1, 4.2.1, 4.2.2

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**

I pointed Yajushi Khurana and Swapnil Keshari to question 1, 2, 4.1.2, 5

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**