

# CMPT 726/410 Fall 2023 Assignment 2

Wenhe Wang (301586596 #)

November 23, 2023

©Simon Fraser University. All Rights Reserved. Sharing this publicly constitutes both academic misconduct and copyright violation.

**Due:** Thursday, November 23, 2023 at 11:59 pm Pacific Time

**Important:** Make sure to download the zip archive associated with this assignment, which contains essential data and starter code that are required to complete the programming component of the assignment.

This assignment is designed to be substantially more challenging than the quizzes and requires thorough understanding of the course material and extensive thought, so **start early!** If you are stuck, come to office hours. Make sure to check the discussion board often for updates, clarifications, corrections and/or hints.

Partial credit will be awarded to reasonable attempts at solving each question, even if they are not completely correct.

There will be office hours dedicated to assignment-related questions. Times and Zoom links will be posted on Canvas. If you have a question that you would like to ask during office hours, make sure to post a brief summary of your question in the office hours thread on the Canvas discussion board. During office hours, questions will be answered in the order they appeared on the office hours thread. We may not be able to get to all questions, so please start early and plan ahead.

**Requests for extensions will not be considered under any circumstances.** Make sure you know how to submit your assignment to Crowdmark and leave sufficient time to deal with any technical difficulties, Internet outages, server downtimes or any other unanticipated events. It is possible to update your assignment after submission, so we recommend uploading a version of your assignment at least several hours before the deadline, even if it is incomplete.

## Submission Instructions

Carefully follow the instructions below when submitting your assignment. Not following instructions will result in point deductions.

1. You should typeset your assignment in LaTeX using this document as a template. We recommend using [Overleaf](#) to compile your LaTeX. Include images in the section they go with and **do not** put them in an appendix.
2. At the beginning of the assignment (see above), please copy the following statement and sign your signature next to it. (macOS Preview and FoxIt PDF Reader, among others, have tools to let you sign a PDF file.) We want to make it *extra* clear so that no one inadvertently cheats.

*"I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted."*

3. At the start of each problem—not subproblem; you only need to do this twice—please state: (1) who you received help from on the problem, and (2) who you provided help to.
4. For all theory questions, make sure to **show your work** and include all key steps in your derivations and proofs. If you apply a non-trivial theorem or fact, you must state the theorem or fact you used, either by name (e.g.: Jensen's inequality) or by writing down the general statement of the theorem or fact. You may use theorems and facts covered in lecture without proof. If you use non-trivial theorems and facts not covered in lecture, you must prove them. All conditions should be checked before applying a theorem, and if a statement follows from a more general fact, the reason why it is a special case of the general fact should be stated. The direction of logical implication must be clearly

stated, e.g.: if statement A implies statement B, statement B implies statement A, or statement A is equivalent to statement B (i.e.: statement A is true if and only if statement B is true). You should **number** the equations that you refer to in other parts and refer to them by their numbers. You must **highlight the final answer or conclusion** in your PDF submission, either by changing the font colour to red or drawing a box around it.

5. For all programming questions, starter code is provided as a Jupyter notebook. You should work out your solution in the notebook. We recommend uploading your notebook to a hosted service like [Google Colab](#) to avoid the installation of dependencies and minimize setup time. When you are done, take a screenshot of your code and output and include it in your PDF. In addition, you should download the Jupyter notebook with your solutions (File → Download → Download .ipynb) and also download your code (File → Download → Download .py). You need to **submit both** as a zip archive named “CMPT726-410\_A2\_<Last Name>\_<Student ID>.zip”, whose directory structure should be the same as that of the zip archive for the starter code. Do **NOT** include any data files we provided. Please include a short file named **README** listing your name and student ID. The PDF should not be included in the zip archive and should be submitted **separately**. Please make sure that your code doesn’t take up inordinate amounts of time or memory. If your code cannot be executed, your solution cannot be verified.
6. Assignment submissions will be accepted through Crowdmark. The submission portal will open a week before the assignment is due. Make sure to set aside at least **one hour** to submit the assignment. Crowdmark will ask you to split the PDF by each part of each question. Make sure to upload the correct pages for each part, and double check when you are done.

## Python Configuration

We recommend using Google Colab to complete the parts of this assignment that require coding. However, if you would like to set up your own Python environment on your computer, follow the instructions below.

1. Ensure you have Python 3 installed. If you don’t, we recommend [miniconda](#) as a package manager. To ensure you’re running Python 3, open a terminal in your operating system and execute the following command: `python --version` **Do not proceed until you’re running some (recent) version of Python 3.**
2. Install `scikit-learn` and `scipy`:  
`conda install -y -c conda-forge scikit-learn scipy matplotlib`
3. To check whether your Python environment is configured properly for this homework, ensure the following Python script executes without error. Pay attention to errors raised when attempting to import any dependencies. Resolve such errors by manually installing the required dependency (e.g. execute `conda install -c conda-forge numpy` for import errors relating to the numpy package).

```
import sys
if sys.version_info[0] < 3:
    raise Exception("Python 3 not detected.")

import numpy as np
import matplotlib.pyplot as plt
from scipy import io
```

4. Please use only the packages listed above.

## Collaboration and Academic Integrity

While collaboration is encouraged, all of your submitted work must be your own. Concretely, that means you may discuss general approaches to problems with other students, but you must write your solutions on your own. For more information, see the [course syllabus](#) under “Academic Integrity.” If you got help from or gave help to other students, please note their names at start of every question. Additionally, sign the declaration at the bottom of this page.

**Warning:** Copying others’ solutions, seeking help from others not in this course, posting questions online or entering questions into automated question answering systems are considered cheating. Consequences are

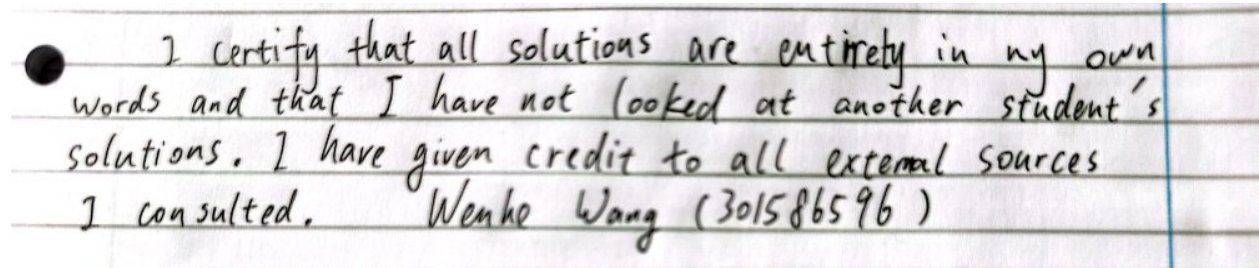
severe and could lead to suspension or expulsion. If you become aware of such instances, you must report them here: <https://forms.gle/9fw3oMLyhD1A81qy5>.

## Page Numbers

<b>Question 1 Parameter Estimation [35 points]</b>	<b>4</b>
Part 1(a) [5 points]	5
Part 1(b) [5 points]	6
Part 1(c) [5 points]	7
Part 1(d) [5 points]	8
Part 1(e) [5 points]	9
Part 1(f) [5 points]	10
Part 1(g) [5 points]	11
<b>Question 2 Linear Regression [20 points]</b>	<b>12</b>
Part 2(a) [5 points]	13
Part 2(b) [5 points]	14
Part 2(c) [5 points]	15
Part 2(d) [5 points]	16
Part 2(e) [Bonus: 3 points]	22

## Declaration

I certify that all solutions are entirely in my own words and that I have not looked at solutions other than my own. I have given credit to all external sources I consulted.



I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted. Wenhe Wang (301586596)

---

Wenhe Wang (301586596 #)

## Question 1   Parameter Estimation [35 points]

In the following parts, consider a random variable  $X$  that follows a distribution parameterized by a parameter  $\theta$ , whose probability density is:

$$p(x|\theta) = \sqrt{\frac{2}{\pi}} \frac{x^2}{\theta^3} \exp\left(\frac{-x^2}{2\theta^2}\right) \quad \theta > 0, x > 0$$

List your collaborators on this question below.

*Solution.*

Students I got help from: No

Students I gave help to: No

External sources that I consulted (websites, books, notes, etc.): Google, Wikipedia

**Part 1(a) [5 points]**

**Write down the likelihood function  $\mathcal{L}(\theta; \mathcal{D})$  of the parameter  $\theta$  given the training data  $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ , where  $x_1, x_2, \dots, x_N$  are assumed to have been drawn independently of each other from identical distributions. Show your work.**

*Solution.*

Since the data  $x_1, x_2, \dots, x_N$  are assumed to be i.i.d. from the same distribution, the likelihood function is the product of the individual probabilities of each observation:

$$\mathcal{L}(\theta; \mathcal{D}) = \prod_{i=1}^N p(x_i | \theta)$$

We also know the given pdf:

$$p(x | \theta) = \sqrt{\frac{2}{\pi}} \frac{x^2}{\theta^3} \exp\left(\frac{-x^2}{2\theta^2}\right)$$

Substituting this into the likelihood function, we get:

$$\mathcal{L}(\theta; \mathcal{D}) = \prod_{i=1}^N \left( \sqrt{\frac{2}{\pi}} \frac{x_i^2}{\theta^3} \exp\left(\frac{-x_i^2}{2\theta^2}\right) \right)$$

We can simplify this:

$$\mathcal{L}(\theta; \mathcal{D}) = \left( \sqrt{\frac{2}{\pi}} \right)^N \cdot \frac{\prod_{i=1}^N x_i^2}{\theta^{3N}} \cdot \exp\left( - \sum_{i=1}^N \frac{x_i^2}{2\theta^2} \right)$$

This is the likelihood function  $\mathcal{L}(\theta; \mathcal{D})$  for the parameter  $\theta$  given the training data  $\mathcal{D}$ .

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

$$\mathcal{L}(\theta; \mathcal{D}) = \left( \sqrt{\frac{2}{\pi}} \right)^N \cdot \frac{\prod_{i=1}^N x_i^2}{\theta^{3N}} \cdot \exp\left( - \sum_{i=1}^N \frac{x_i^2}{2\theta^2} \right)$$

**Part 1(b) [5 points]**

**Write down the log-likelihood function  $\log \mathcal{L}(\theta; \mathcal{D})$  of the parameter  $\theta$  given the training data  $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ , where  $x_1, x_2, \dots, x_N$  are assumed to have been drawn independently of each other from identical distributions. Move the logs in as much as possible and simplify. Show your work.**

***Solution.***

From the previous answer, we have the likelihood function expression:

$$\mathcal{L}(\theta; \mathcal{D}) = \left( \sqrt{\frac{2}{\pi}} \right)^N \cdot \frac{\prod_{i=1}^N x_i^2}{\theta^{3N}} \cdot \exp \left( - \sum_{i=1}^N \frac{x_i^2}{2\theta^2} \right)$$

The log-likelihood function is:

$$\begin{aligned} \log \mathcal{L}(\theta; \mathcal{D}) &= \log \left( \left( \sqrt{\frac{2}{\pi}} \right)^N \cdot \frac{\prod_{i=1}^N x_i^2}{\theta^{3N}} \cdot \exp \left( - \sum_{i=1}^N \frac{x_i^2}{2\theta^2} \right) \right) \\ \log \mathcal{L}(\theta; \mathcal{D}) &= N \log \left( \sqrt{\frac{2}{\pi}} \right) + \log \left( \prod_{i=1}^N x_i^2 \right) - 3N \log(\theta) - \sum_{i=1}^N \frac{x_i^2}{2\theta^2} \\ \log \mathcal{L}(\theta; \mathcal{D}) &= N \log \left( \sqrt{\frac{2}{\pi}} \right) + 2 \sum_{i=1}^N \log(x_i) - 3N \log(\theta) - \sum_{i=1}^N \frac{x_i^2}{2\theta^2} \end{aligned}$$

This is the log-likelihood function  $\log \mathcal{L}(\theta; \mathcal{D})$  for the parameter  $\theta$  given the training data  $\mathcal{D}$ .

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

$$\log \mathcal{L}(\theta; \mathcal{D}) = N \log \left( \sqrt{\frac{2}{\pi}} \right) + 2 \sum_{i=1}^N \log(x_i) - 3N \log(\theta) - \sum_{i=1}^N \frac{x_i^2}{2\theta^2}$$

**Part 1(c) [5 points]**

**Derive the expression for critical point(s) of the log-likelihood.** Show your work.

*Solution.*

From the previous answer, we have the log-likelihood function expression:

$$\log \mathcal{L}(\theta; \mathcal{D}) = N \log \left( \sqrt{\frac{2}{\pi}} \right) + 2 \sum_{i=1}^N \log(x_i) - 3N \log(\theta) - \sum_{i=1}^N \frac{x_i^2}{2\theta^2}$$

To find the critical point(s) of the log-likelihood function, essentially it is the same as taking the derivative of the function on  $\theta$ , and then letting the derivative equal 0, so we have:

$$\frac{\partial}{\partial \theta} \log \mathcal{L}(\theta; \mathcal{D}) = \frac{\partial}{\partial \theta} \left( N \log \left( \sqrt{\frac{2}{\pi}} \right) + 2 \sum_{i=1}^N \log(x_i) - 3N \log(\theta) - \sum_{i=1}^N \frac{x_i^2}{2\theta^2} \right)$$

There is no  $\theta$  in the first two terms, so we can eliminate them directly:

$$\frac{\partial}{\partial \theta} \log \mathcal{L}(\theta; \mathcal{D}) = -3N \frac{\partial}{\partial \theta} (\log(\theta)) - \sum_{i=1}^N \frac{\partial}{\partial \theta} \left( \frac{x_i^2}{2\theta^2} \right)$$

Make the right-hand side to 0, we have:

$$-\frac{3N}{\theta} + \sum_{i=1}^N \frac{x_i^2}{\theta^3} = 0$$

We want to find an equation with only  $\theta$  on one side, and something on the other side, so we keep simplifying this:

$$\begin{aligned} -3N\theta^2 + \sum_{i=1}^N x_i^2 &= 0 \\ \theta^2 &= \frac{1}{3N} \sum_{i=1}^N x_i^2 \end{aligned}$$

Finally, we find this critical point(s) of the log-likelihood function.

$$\theta = \sqrt{\frac{1}{3N} \sum_{i=1}^N x_i^2}$$

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

$$\theta = \sqrt{\frac{1}{3N} \sum_{i=1}^N x_i^2}$$

**Part 1(d) [5 points]**

**Prove that the critical point found in the previous part is a local maximum. Then using the fact that there is only one critical point, show that the critical point is the global maximum.** Show your work and justify each step that is not a simple algebraic manipulation.

*Solution.*

To show that the critical point found in the previous part is a local maximum and the critical point is the global maximum, we need to prove that this function is a strictly concave function and also monotonically increasing throughout the domain of  $\theta > 0$ , such that every local maximum is also a global maximum and it is not a saddle point. Because there is only one critical point being found, if it is a local maximum, then it must be the global maximum.

Using the fact that the objective function is a logarithm function that is strictly concave throughout the domain of  $\theta > 0$ . Thus, we only need to prove the second point right now.

From the previous answer, we know the first-order derivative of the log-likelihood function:

$$\frac{d}{d\theta} \log \mathcal{L}(\theta) = -\frac{3N}{\theta} + \sum_{i=1}^N \frac{x_i^2}{\theta^3}$$

Now we derive the second-order derivative of the log-likelihood function on  $\theta > 0$  again:

$$\begin{aligned} \frac{d^2}{d\theta^2} \log \mathcal{L}(\theta) &= \frac{d}{d\theta} \left( -\frac{3N}{\theta} + \sum_{i=1}^N \frac{x_i^2}{\theta^3} \right) \\ \frac{d^2}{d\theta^2} \log \mathcal{L}(\theta) &= 3\frac{N}{\theta^2} - 3 \sum_{i=1}^N \frac{x_i^2}{\theta^4} \end{aligned}$$

If  $\theta$  is less than 1, then, the magnitude of the negative term grows faster than the positive term as  $\theta > 0$  changes since it is scaled by  $\theta^{-4}$  as opposed to  $\theta^{-2}$ , so we will have an always negative second-order derivative. In this case, our objective function will be convex, and we will have 1 maximum point since  $\theta$  can not be 0, the maximum point will be 1 instead.

If  $\theta$  is greater than 1, the positive term grows faster than the positive term as  $\theta > 0$  changes, such that we will have an always positive second-order derivative and it means our objective function will be concave, and as such we will also have 1 critical maximum point which is the same as the point at 1.

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

As being proved, the critical point is the global maximum.



**Part 1(e) [5 points]**

**Given a prior over  $\theta$ , i.e.,  $\theta \sim \mathcal{N}(0, 1)$ , write down the objective function for finding the MAP estimate of the parameter  $\theta$ .** Simplify the objective function to a form that can be differentiated to find the critical point(s) in closed form easily. Justify each step that is not a simple algebraic manipulation.

***Solution.***

Given the prior  $\theta \sim \mathcal{N}(0, 1)$ , the probability density function is:

$$p(\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right)$$

The likelihood function  $\mathcal{L}(\theta; \mathcal{D})$  is also given by the product of the probability densities of all data:

$$\mathcal{L}(\theta; \mathcal{D}) = \prod_{i=1}^N p(x_i|\theta)$$

The posterior can be expressed as part of the likelihood function and the prior probability density:

$$\log p(\theta|\mathcal{D}) = \log \mathcal{L}(\theta; \mathcal{D}) + \log p(\theta)$$

Substituting the expressions 1 and 2 into this equation, we have:

$$\log p(\theta|\mathcal{D}) = \left( \sum_{i=1}^N \log p(x_i|\theta) \right) + \left( -\frac{\theta^2}{2} - \log \sqrt{2\pi} \right)$$

The constant can be taken out since it will not affect the optimization, and as such the objective function is:

$$\log p(\theta|\mathcal{D}) = \sum_{i=1}^N \log p(x_i|\theta) - \frac{\theta^2}{2}$$

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

$$\log p(\theta|\mathcal{D}) = \sum_{i=1}^N \log p(x_i|\theta) - \frac{\theta^2}{2}$$

**Part 1(f) [5 points]**

**Differentiate the objective function in the previous part to find the expression for the critical point. Then prove it is a local maximum and a global maximum.** Show your work and justify each step that is not a simple algebraic manipulation.

*Solution.*

From the previous answer, we have the objective function for the MAP estimate expression:

$$\log p(\theta|\mathcal{D}) = \sum_{i=1}^N \log p(x_i|\theta) + \log p(\theta)$$

Given  $p(x|\theta)$  and  $\theta \sim \mathcal{N}(0, 1)$  and  $\log p(\theta|\mathcal{D}) = \sum_{i=1}^N \log p(x_i|\theta) + \log p(\theta)$  and  $\log p(\theta|\mathcal{D}) = \sum_{i=1}^N \log p(x_i|\theta) - \frac{\theta^2}{2}$  from the previous answer:

$$\log p(\theta|\mathcal{D}) = N \left( \frac{1}{2} \log \left( \frac{2}{\pi} \right) - 3 \log(\theta) \right) + \sum_{i=1}^N \left( \log(x_i^2) - \frac{x_i^2}{2\theta^2} \right) - \frac{\theta^2}{2}$$

Then we take the first-order derivative of our objective function on  $\theta$ :

$$\frac{d}{d\theta} \log p(\theta|\mathcal{D}) = -3N \frac{1}{\theta} + \sum_{i=1}^N \frac{x_i^2}{\theta^3} - \theta$$

Similar to Part 1(c), setting the derivative to 0 in order to find the critical point(s), we have:

$$-3N \frac{1}{\theta} + \sum_{i=1}^N \frac{x_i^2}{\theta^3} - \theta = 0$$

There is no obvious equation for placing this critical point on the left-hand side, and everything else on the right. But we can run code iteratively to find this critical point.

After we find this point, we need to prove the concavity of our objective function by analyzing the second-order derivative of our objective function and using the nature of the log function just like Part 1(d), so we can use the same reasoning from Part 1(d) to prove that the critical point is the global maximum.

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

$$-3N \frac{1}{\theta} + \sum_{i=1}^N \frac{x_i^2}{\theta^3} - \theta = 0$$

This equation can be solved for  $\theta$  to find the critical point, and then we can use the same reasoning as Part 1(d) to show the critical point is the global maximum.

**Part 1(g) [5 points]**

Consider an arbitrary probability distribution  $p(x|\theta)$  parameterized by a scalar parameter  $\theta \in [a, b]$ . **Given that the maximum likelihood estimate (MLE) and maximum a posteriori (MAP) estimate of  $\theta$  are the same, can you find the prior distribution of  $\theta$ ?** If so, state the name of the distribution and its parameters, and derive it. If not, explain why.

*Solution.*

The objective function of MLE is the likelihood function. While the objective function of MAP is the posterior distribution. The objective function of MLE is part of the function of MAP except for the prior  $\log p(\theta)$ .

If their estimates are the same, it means the prior distribution  $\log p(\theta)$  does not affect the process of maximization of these two objective functions. So the prior distribution of  $\log p(\theta)$  is either a constant function or a uniform distribution. When taking the derivatives of our objective function, we always get 0 for this term in our process of optimization.

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

The prior distribution of  $\log p(\theta)$  is either a constant function or a uniform distribution.

## Question 2 Linear Regression [20 points]

List your collaborators on this question below.

*Solution.*

Students I got help from: No

Students I gave help to: No

External sources that I consulted (websites, books, notes, etc.): Google

Wikipedia

[https://web.stanford.edu/~mrosenfe/soc\\_meth\\_proj3/matrix\\_OLS\\_YU\\_notes.pdf](https://web.stanford.edu/~mrosenfe/soc_meth_proj3/matrix_OLS_YU_notes.pdf)

<https://www.mdpi.com/1424-8220/20/12/3494>

## Part 2(a) [5 points]

Consider a vector  $\vec{v} \in \mathbb{R}^n$ , where  $n \geq 2$ . Given the first element of  $\vec{v}$  and the differences between adjacent elements of  $\vec{v}$ , our goal is to reconstruct the vector.

Formulate an ordinary least squares (OLS) problem such that the input data  $A$  and desired outputs  $\vec{y}$  are derived from the quantities we are given and the solution (i.e.,  $\vec{w}^* := \arg \min_{\vec{w}} \|A\vec{w} - \vec{y}\|_2^2$ ) can be turned into a reconstruction of the vector. **Write down what each element of  $A$ ,  $\vec{y}$  and  $\vec{w}$  corresponds to. What are the dimensions of  $A$ ,  $y$  and  $\vec{w}$ ?** Justify your answer.

### *Solution.*

To formulate a general OLS problem, we first need to know what is  $\vec{x}$  and  $\vec{y}$ , and normally an OLS problem is like  $A\vec{x} = \vec{y}$ . In our case,  $\vec{x}$  is  $\vec{w}$  corresponds to the vector  $\vec{v}$  given by the statement, and  $\vec{y}$  is the actual differences between adjacent elements of  $\vec{v}$ . And we also know  $A$  is the adjacent relationship for elements in  $\vec{w}$ . And we will try to minimize this expression  $\|A\vec{w} - \vec{y}\|_2^2$  to achieve the best estimate  $\vec{w}^*$  of  $\vec{v}$ .

So we can define all the parameters in our OLS problem as:

- $\vec{v} = [v_1, v_2, \dots, v_n]^\top$ , where  $v_1$  is the first element of  $\vec{v}$ , dimension is  $n \times 1$ .
- The differences between adjacent elements:  $d_i = v_{i+1} - v_i$  for  $i = 1, 2, \dots, n-1$ , dimension is  $(n-1) \times 1$ .
- Matrix  $A$  (coefficients), the adjacent relationships for elements in  $\vec{w}$ :

$$A = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

Dimension is  $(n-1) \times n$ .

- $\vec{y} = [d_1, d_2, \dots, d_{n-1}]^\top$ , dimension is  $(n-1) \times 1$ .
- $\vec{w}$ , the unknown vector we need to solve, dimension is  $n \times 1$ .

And the OLS problem is:

$$\vec{w}^* := \arg \min_{\vec{w}} \|A\vec{w} - \vec{y}\|_2^2$$

The solution  $\vec{w}^*$  will give us the best OLS approximation of  $\vec{v}$ . However, since the linear system has  $n-1$  equations and  $n$  unknowns, this problem can not be solved, so we will give the first element  $v_1$  as  $w_1$  to start the system.

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

- Matrix  $A$  (coefficients), the adjacent relationships for elements in  $\vec{w}$ :

$$A = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

Dimension is  $(n-1) \times n$ .

- $\vec{y} = [d_1, d_2, \dots, d_{n-1}]^\top$ , dimension is  $(n-1) \times 1$ .
- $\vec{w}$ , the unknown vector we need to solve, dimension is  $n \times 1$ .

And the OLS problem is:

$$\vec{w}^* := \arg \min_{\vec{w}} \|A\vec{w} - \vec{y}\|_2^2$$

## Part 2(b) [5 points]

Any grayscale image can be represented as a matrix  $I$  where  $I_{i,j}$  is the intensity of the pixel on the  $i$ th row and  $j$ th column. Consider an  $n \times n$  image  $I \in \mathbb{R}^{n \times n}$ , where  $n \geq 2$ . Given the values of  $I$  in the first column and the horizontal image gradients, which are the differences between values in adjacent columns, our goal is to reconstruct the full image.

More precisely, we denote the horizontal image gradient at a pixel in the  $i$ th row and  $j$ th column as  $\Delta_{i,j}^x$ , which is defined to be  $I_{i,j+1} - I_{i,j}$ . We are given  $\Delta_{i,j}^x$  for all  $i \in \{1, \dots, n\}$  and  $j \in \{1, \dots, n-1\}$  and  $I_{i,1}$  for all  $i \in \{1, \dots, n\}$ , and would like to find  $I_{i,j}$  for all  $i \in \{1, \dots, n\}$  and  $j \in \{2, \dots, n\}$ .

Formulate an ordinary least squares (OLS) problem such that the input data  $A$  and desired outputs  $\vec{y}$  are derived from the quantities we are given and the solution (i.e.,  $\vec{w}^* := \arg \min_{\vec{w}} \|A\vec{w} - \vec{y}\|_2^2$ ) can be turned into a reconstruction of the image. **Write down what each element of  $A$ ,  $\vec{y}$  and  $\vec{w}$  corresponds to. What are the dimensions of  $A$ ,  $y$  and  $\vec{w}$ ?** Justify your answer.

### *Solution.*

Similar to the previous OLS problem, this time we need to reconstruct not only one vector but a number of vectors (pixel intensities). And we can formulate our OLS problem:

$$\vec{w}^* := \arg \min_{\vec{w}} \|A\vec{w} - \vec{y}\|_2^2$$

We put both the knowns and unknowns coefficients in one matrix for  $A$ :

- The matrix will have  $n \times (n-1) + n$  rows, where  $n \times (n-1)$  rows come from the horizontal gradient information for each pixel (excluding the last column) and  $n$  rows for the known first column values.
- Each column in  $A$  corresponds to an unknown pixel value in the image, excluding the first column which is already known. So, there are  $n \times (n-1)$  columns.

Dimension of  $A$  is  $(n \times (n-1) + n) \times (n \times (n-1))$ .

The vector  $\vec{w}$  represents the unknown pixel values we are trying to solve for (all the pixel values except for the first column).

The length of  $\vec{w}$  will be the same as the number of columns in  $A$ , i.e.,  $n \times (n-1)$ .

Dimension of  $\vec{w}$  is  $(n \times (n-1)) \times 1$ .

The vector  $\vec{y}$  contains the known values on the right-hand side of each equation. It consists of the horizontal gradient values for each pixel and the known first-column values.

The length of  $\vec{y}$  will be the same as the number of rows in  $A$ , i.e.,  $n \times (n-1) + n$ .

Dimension of  $\vec{y}$  is  $(n \times (n-1) + n) \times 1$ .

Similarly, the solution  $\vec{w}^*$  will give us the best OLS approximation of the unknown pixel intensities  $I$ .

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

Dimension of  $A$  is  $(n \times (n-1) + n) \times (n \times (n-1))$ .

Dimension of  $\vec{w}$  is  $(n \times (n-1)) \times 1$ .

Dimension of  $\vec{y}$  is  $(n \times (n-1) + n) \times 1$ .

## Part 2(c) [5 points]

Unlike in the previous part, we are no longer given the values of  $I$  in the first column. However, in addition to being given the horizontal image gradients, we are also given the vertical image gradients, which are the differences between values in adjacent rows. Our goal is still to reconstruct the full image.

More precisely, we will denote horizontal image gradients in the same way as before, and the vertical image gradient at a pixel in the  $i$ th row and  $j$ th column as  $\Delta_{i,j}^y$ , which is defined to be  $I_{i+1,j} - I_{i,j}$ . We are given  $\Delta_{i,j}^x$  for all  $i \in \{1, \dots, n\}$  and  $j \in \{1, \dots, n-1\}$ , and  $\Delta_{i,j}^y$  for all  $i \in \{1, \dots, n-1\}$  and  $j \in \{1, \dots, n\}$ .

We are given  $\Delta_{i,j}^x$  for all  $i \in \{1, \dots, n\}$  and  $j \in \{1, \dots, n-1\}$  and  $\Delta_{i,j}^y$  for all  $i \in \{1, \dots, n-1\}$  and  $j \in \{1, \dots, n\}$ , and would like to find  $I_{i,j}$  for all  $i \in \{1, \dots, n\}$  and  $j \in \{1, \dots, n\}$ .

Formulate an ordinary least squares (OLS) problem such that the input data  $A$  and desired outputs  $\vec{y}$  are derived from the quantities we are given and the solution (i.e.,  $\vec{w}^* := \arg \min_{\vec{w}} \|A\vec{w} - \vec{y}\|_2^2$ ) can be turned into a reconstruction of the image. **Write down what each element of  $A$ ,  $\vec{y}$  and  $\vec{w}$  corresponds to. What are the dimensions of  $A$ ,  $y$  and  $\vec{w}$ ?** Justify your answer.

*Solution.*

Similar to the previous problem, our OLS problem is:

$$\vec{w}^* := \arg \min_{\vec{w}} \|A\vec{w} - \vec{y}\|_2^2$$

Matrix  $A$ , contains all the horizontal and vertical gradients coefficients:

There are  $n \times (n-1)$  rows for horizontal gradients and  $(n-1) \times n$  rows for vertical gradients.

Each column in  $A$  corresponds to an unknown pixel value  $I_{i,j}$ , so there are  $n \times n$  columns.

Dimension of  $A$  is  $2n(n-1) \times n^2$ .

$\vec{w}$ , the unknown pixel values in the image

Dimension of  $\vec{w}$  is  $n^2 \times 1$ .

$\vec{y}$ , contains the known gradient values:

Dimension of  $\vec{y}$  is  $2n(n-1) \times 1$ .

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

Dimension of  $A$  is  $2n(n-1) \times n^2$ .

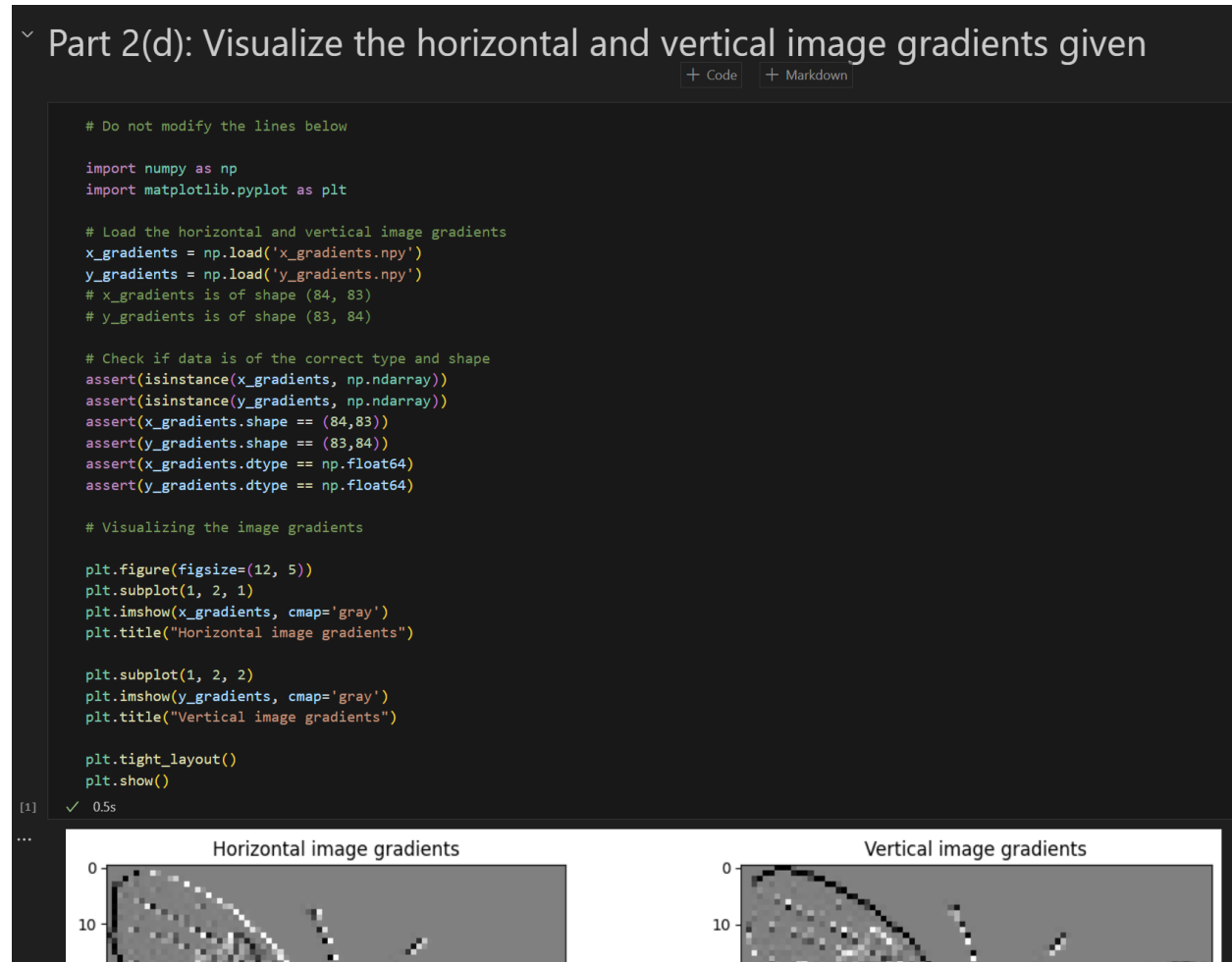
Dimension of  $\vec{w}$  is  $n^2 \times 1$ .

Dimension of  $\vec{y}$  is  $2n(n-1) \times 1$ .

## Part 2(d) [5 points]

In the `x_gradients.npy` and `y_gradients.npy` files, you are given the horizontal image gradients and vertical image gradients of an unknown image. Write an implementation that constructs the input data matrix  $A$  and the vector of desired outputs  $\vec{y}$  used in the OLS formulation in the previous part. Then write an implementation that solves the OLS problem using the closed-form formula for the solution. You can only use functions that implement basic linear algebra operations, such as matrix multiplication, inverses, transposes and solving a linear system of equations where the number of equations is equal to the number of unknowns. You have been provided with starter code for loading and visualizing data.

*Solution.*







```
#                                     YOUR CODE ABOVE                                     #
#####

# Do not modify the lines below
# Check if A and y are of the correct type and have sensible shapes

assert(isinstance(A, np.ndarray))
assert(isinstance(y, np.ndarray))
assert(len(A.shape) == 2)
assert(len(y.shape) == 1)
assert(A.shape[0] == y.shape[0])
assert(A.dtype == np.float64)
assert(y.dtype == np.float64)

print("Sample of Input Data: {}\n{}\n\nSample of Desired Outputs: {}\n{}".format(A.shape, A[:1000,:1000], y.shape, y[:100]))

✓ 0.0s

Sample of Input Data: (13944, 7056)
[[-1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.]]

Sample of Desired Outputs: (13944,)
[ 0.          0.          0.          0.          0.01960784
  0.         -0.2235294  0.          0.          0.02352941  0.19607843
 -0.00392157  0.65098035  0.         -0.09019608  0.05882353  0.
  0.          0.01960784  0.01176471  0.12549019 -0.95294118  0.18431371
  0.01568628 -0.03921569  0.02745098 -0.67843139  0.14509803 -0.00784314
 -0.04313726 -0.47843137  0.56470591 -0.21568629 -0.01960784 -0.03529412
  0.4039216   0.18039218  0.0627451   0.          0.04313725 -0.89803922
 -0.74117649  0.02352941  0.          0.03137255  0.         -0.11764705
 ...
 -0.60000002  0.         -0.32156864  0.         -0.01568627  0.02352941
  0.          0.          0.         -0.00784314 -0.40392157  0.
  0.          0.          0.          0.00784314  0.          0.]
```

```

Sample of Desired Outputs: (13944,)
[ 0.         0.         0.         0.         0.         0.01960784
  0.        -0.2235294  0.         0.         0.02352941  0.19607843
 -0.00392157  0.65098035  0.         -0.09019608  0.05882353  0.
  0.         0.01960784  0.01176471  0.12549019 -0.95294118  0.18431371
  0.01568628 -0.03921569  0.02745098 -0.67843139  0.14509803 -0.00784314
 -0.04313726 -0.47843137  0.56470591 -0.21568629 -0.01960784 -0.03529412
  0.4039216   0.18039218  0.0627451   0.         0.04313725 -0.89803922
 -0.74117649  0.02352941  0.         0.03137255  0.         -0.11764705
 ...
 -0.60000002  0.         -0.32156864  0.         -0.01568627  0.02352941
  0.         0.         0.         -0.00784314 -0.40392157  0.
  0.         0.         0.         0.00784314  0.         0.
  0.         0.         ]

```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings...](#)

## Part 2(d): Solve the OLS problem and rearrange the solution into the reconstructed image

```

# Solve the OLS problem given the input data matrix A and the desired outputs y.
# Then rearrange the solution w into the reconstructed image
#
# Required: Write code below to find the solution w to the OLS problem and
# rearrange w into the reconstructed image
#
# w should be a vector of type numpy.float64 that contains the solution to the
# OLS problem
# w_as_img should be a matrix of type numpy.float64 that contains the
# reconstructed image
#
# Include a screenshot of this part in your PDF submission
#
#####
#                               YOUR CODE BELOW                               #
#####

w = np.linalg.lstsq(A, y, rcond=None)[0] # Solve the OLS problem
w_as_img = w.reshape((n, n)) # Reshaping our solution to the size of image

```

```

# Solve the OLS problem given the input data matrix A and the desired outputs y.
# Then rearrange the solution w into the reconstructed image
#
# Required: Write code below to find the solution w to the OLS problem and
# rearrange w into the reconstructed image
#
# w should be a vector of type numpy.float64 that contains the solution to the
# OLS problem
# w_as_img should be a matrix of type numpy.float64 that contains the
# reconstructed image
#
# Include a screenshot of this part in your PDF submission
#
#####
#                                     YOUR CODE BELOW
#####

# Solving the OLS problem using SVD
U, s, Vh = np.linalg.svd(A, full_matrices=False)
w = Vh.T @ np.linalg.inv(np.diag(s)) @ U.T @ y

# Reshaping our solution to the size of image
w_as_img = w.reshape((n, n))

#####
#                                     YOUR CODE ABOVE
#####

# Do not modify the lines below
# Check if w is of the correct type and has a sensible shape
assert(isinstance(w_as_img, np.ndarray))
assert(len(w.shape) == 1)
assert(w.shape[0] == A.shape[1])
assert(w.dtype == np.float64)

# Check if w_as_img is of the correct type and has a sensible shape
assert(isinstance(w_as_img, np.ndarray))
assert(len(w_as_img.shape) == 2)
assert(np.prod(w_as_img.shape) == w.shape[0])
assert(w_as_img.shape[0] == x_gradients.shape[0])
assert(w_as_img.shape[1] == x_gradients.shape[1]+1)
assert(w_as_img.shape[0] == y_gradients.shape[0]+1)
assert(w_as_img.shape[1] == y_gradients.shape[1])
assert(w_as_img.dtype == np.float64)

```



Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE



**Part 2(e) [Bonus: 3 points]**

In one or two sentences comment on the memory requirement of the approach implemented for the previous part and if it can be reduced by considering the characteristics of input data matrix  $A$ .

***Solution.***

Since the majority of entries in matrix  $A$  are zero, owing to each row representing first-order derivative coefficients that involve only two-pixel values, we can effectively utilize sparse matrix representations and operations. This approach is particularly advantageous given the substantial size of  $A$ , enabling more memory-efficient handling of the data.

**Final Conclusion: YOUR FINAL ANSWER OR CONCLUSION HERE**

We can use the sparse matrix to represent the huge matrix  $A$  and calculate based on sparse matrix rules to save memory.