HOMEWORK 5
MATRIX CALCULUS *

10-606 MATHEMATICAL FOUNDATIONS FOR MACHINE LEARNING

START HERE: Instructions

- **Collaboration Policy**: Please read the collaboration policy in the syllabus.
- **Late Submission Policy:** See the late submission policy in the syllabus.
- **Submitting your work:** You will use Gradescope to submit answers to all questions.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please use the provided template. 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. Alternatively, submissions can be written in LaTeX. Each derivation/proof should be completed in the boxes provided. To receive full credit, you are responsible for ensuring that your submission contains exactly the same number of pages and the same alignment as our PDF template.
  - **Latex Template:** https://www.overleaf.com/read/cqpdfdqbghqm#c0f172

Question	Points
Matrix Derivatives for Weighted Linear Regression	12
Gradient Descent	16
Total:	28

*Compiled on Wednesday 25th September, 2024 at 21:02

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

- ● Matt Gormley
- ○ Marie Curie
- ○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

- ● Henry Chai
- ○ Marie Curie
- ⊗ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- □ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

- ■ Stephen Hawking
- ■ Albert Einstein
- ■ Isaac Newton
- □ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

10-606	10-6067

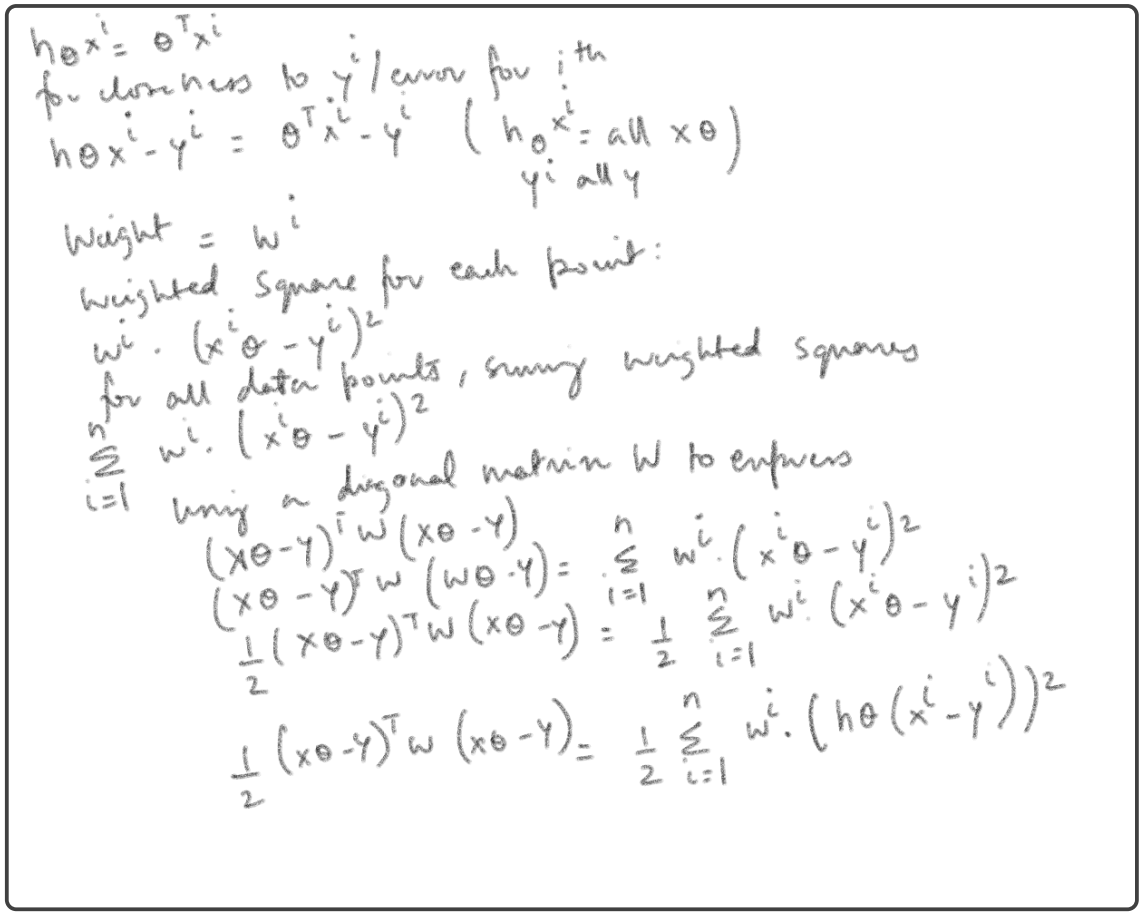
1 Matrix Derivatives for Weighted Linear Regression (12 points)

In statistics, linear regression is an approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). Relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. In this problem, we will try to fit a weighted regression model to a training dataset $(\mathbf{X}, \mathbf{Y}, \mathbf{w})$ of n datapoints where the input features $\mathbf{X} \in \mathbb{R}^{n \times m}$, the real-valued output $\mathbf{Y} \in \mathbb{R}^{n \times 1}$, and the non-negative real-valued datapoint weights $\mathbf{w} \in \mathbb{R}^{n \times 1}$.

Now, given a training set, we would like to learn a parameter $\boldsymbol{\theta}$, such that the function $h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$ is close to y for the training examples we have. To formalize this, we will define a function that measures, for each value of $\boldsymbol{\theta}$, how close the $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$'s are to the corresponding $y^{(i)}$'s. We define the objective function using provided weights on the datapoints as follows:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \mathbf{w}^{(i)} (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)})^2$$

1. (3 points) Show that $J(\boldsymbol{\theta}) = \frac{1}{2}(\mathbf{X}\boldsymbol{\theta} - \mathbf{Y})^T \mathbf{W} (\mathbf{X}\boldsymbol{\theta} - \mathbf{Y})$ where $\mathbf{W}$ is a diagonal matrix with the weights from $\mathbf{w}$ placed on the diagonal and zeros in all off-diagonal entries. Show your work in the box below. (**Hint:** The i^{th} row of $\mathbf{X}$ is $\mathbf{x}^{(i)}$ and the i^{th} element of $\mathbf{y}$ is $\mathbf{y}^{(i)}$. Consider how multiplying the rows of $\mathbf{X}$ to $\boldsymbol{\theta}$ relates to $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = \boldsymbol{\theta}^T \mathbf{x}^{(i)}$. Carefully expand and simplify the matrix-vector notation to derive the weighted sum of squared errors version.)



2. (3 points) To minimize J, we need to find its gradient with respect to $\boldsymbol{\theta}$. Derive $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. (**Hint:** You can use the following gradient formulae $\nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{A}\mathbf{x} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$ and $\nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{y} = \nabla_{\mathbf{x}} \mathbf{y}^T \mathbf{x} = \mathbf{y}$)

$$J(\theta) = \frac{1}{2}(X\theta - Y)^T W (X\theta - Y)$$

$$X\theta - Y = z$$

$$J(\theta) = \frac{1}{2} z^T W z$$

$$W^T = W$$

$$\nabla_\theta J(\theta) = \frac{1}{2}(X^T W + X^T W) z$$

$$= \frac{1}{2}(2 \cdot X^T W (X\theta - Y))$$

$$= X^T W (X\theta - Y)$$

3. (3 points) To minimize J, set its gradient to zero, and solve for $\boldsymbol{\theta}$.

$$\nabla_\theta J(\theta) = 0$$

$$X^T W (X\theta - Y) = 0$$

$$X^T W (X\theta - Y) = 0$$

$$X^T W X\theta - X^T W Y = 0$$

if $X^T W X$ is invertible then multiplying both sides by inverse

$$X^T W X\theta = X^T W Y$$

$$\theta = (X^T W X)^{-1} \cdot X^T W Y$$

Now suppose we used an l_2 regularizer. Now, our objective function is,

$$J'(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n} \mathbf{w}^{(i)}(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{1}{2}\lambda\|\boldsymbol{\theta}\|^2$$

1. (3 points) To minimize J', we need to find its gradient with respect to $\boldsymbol{\theta}$. Derive $\nabla_{\boldsymbol{\theta}} J'(\boldsymbol{\theta})$. (**Hint:** You can use the following gradient formulae $\nabla_{\mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{x} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$ and $\nabla_{\mathbf{x}}\mathbf{x}^T\mathbf{y} = \nabla_{\mathbf{x}}\mathbf{y}^T\mathbf{x} = \mathbf{y}$. Also, $\|\boldsymbol{\theta}\|^2 = \boldsymbol{\theta}^T\boldsymbol{\theta}$)

$J'\theta = \frac{1}{2}(X\theta - Y)^T W (X\theta - Y) + \frac{1}{2}\lambda\theta^T\theta$

$\nabla_\theta = \left(\frac{1}{2}(X\theta - Y)^T W (X\theta - Y)\right) = X^T W (X\theta - Y)$

$\nabla_\theta = \left(\frac{1}{2}\lambda\theta^T\theta\right) = \lambda\theta$

$\nabla_\theta J'(\theta) = X^T W (X\theta - Y) + \lambda\theta$

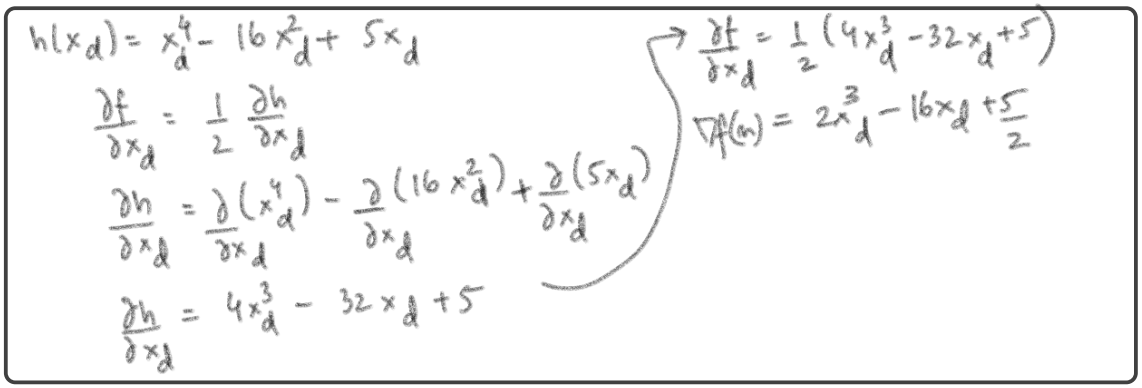
2 Gradient Descent (16 points)

In this problem, you will seek minima of a nonconvex function, the Styblinski-Tang function. The form of the function is:

$$f(\mathbf{x}) = f(x_1, x_2, ..., x_D) = \frac{1}{2}\sum_{d=1}^{D}(x_d^4 - 16x_d^2 + 5x_d)$$

where $\mathbf{x} \in \mathbb{R}^D$ are the parameters of the function to be optimized. You will optimize this function using gradient descent.

1. (3 points) **Derivation:** What is the gradient of $f(\mathbf{x})$ with respect to $\mathbf{x}$?



2. (7 points) The gradient descent algorithm computes the gradient at each step and steps opposite the gradient multiplied by a step size γ. Implement gradient descent for the function $f(\mathbf{x})$. Your implementation should be a Python function that accepts three parameters:

   `gd(x_initial, step_size, max_iterations)`

   where `x_initial` is a numpy vector of length D representing the initial point for gradient descent, `step_size` is the step size γ, and `max_iterations` is the total number of iterations that gradient descent should run before stopping. Your implementation should work for arbitrary D. At the end, it should print out *both* the value of $\mathbf{x}$ after running for the specified number of iterations and the value of $f(\mathbf{x})$ at that point. You should only use the `numpy` library.

```python
import numpy as np

def gd(x_initial, step_size, max_iterations):
    xNew = x_initial.copy()
    for i in range(max_iterations):
        gradient = 2*(xNew)**3 - 16*(xNew) + (5/2)
        xNew -= step_size * gradient

    print("x after running specified number of iterations:", xNew)
    print("f(x) at point:", 0.5 * np.sum(xNew**4 - 16*xNew**2 + 5*xNew))
```

3. (2 points) **Short answer:** What are the values of $f(\mathbf{x})$ and $\mathbf{x}$ found by your implementation when it is called as follows?

```
gd(np.array([5,5]), 0.01, 100)
```

```
x after running specified number of iterations: [2.74680277 2.74680277]
f(x) at point: -50.05889331056788
```

4. (2 points) **Short answer:** What are the values of $f(\mathbf{x})$ and $\mathbf{x}$ found by your implementation when it is called as follows?

```
gd(np.array([-5,-5]), 0.01, 100)
```

```
x after running specified number of iterations: [-2.90353403 -2.90353403]
f(x) at point: -78.33233140754282
```

5. (2 points) **Short answer:** Suppose you only knew that the Python function `gd(...)` was running gradient descent on a non-convex function (i.e. assume you don't know the exact form of the function $f(\mathbf{x})$ that it's optimizing). Provide **two** reasons why `gd(np.array([5,5]), 0.01, 100)` and `gd(np.array([-5,-5]), 0.01, 100)` could return different values of $\mathbf{x}$.

Reason 1

Different local minima changes gradient descent.

Reason 2

The starting points would change the direction of it.

3 Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found in the syllabus.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details.
3. Did you find or come across code that implements any part of this assignment? If so, include full details.

Your Answer

1. No, I haven't.
2. No, I haven't.
3. No, I did not.