Homework 1

Instructions

- **Conceptual Problems:** Show all steps in your derivations. You may use standard matrix calculus identities derived in class.
- **Programming Problems:** You must use Python and the `JAX` library as demonstrated in the lecture notes. Submit your code and the resulting output (plots or printed values).

Part A: Conceptual Problems

Problem 1: Taylor Approximations and Curvature (Chapter 1)

Consider the function of two variables $f : \mathbb{R}^2 \to \mathbb{R}$ defined by:

$$f(x_1, x_2) = x_1^3 + x_2^3 - 3x_1x_2$$

1. Compute the gradient vector $\nabla f(\mathbf{x})$.
2. Find the critical points of the function (where $\nabla f(\mathbf{x}) = \mathbf{0}$).
3. Compute the Hessian matrix $\nabla^2 f(\mathbf{x})$.
4. Evaluate the Hessian at each critical point found in part (2). Based on the eigenvalues of the Hessian (or the determinant/trace test), classify each critical point as a local minimum, local maximum, or saddle point.

Problem 2: Matrix Calculus and Ridge Regression (Chapter 2)

In Lecture 2, we derived the Normal Equations for Ordinary Least Squares (OLS) by minimizing $||X\beta - y||_2^2$. A common modification to OLS to prevent overfitting is called *Ridge Regression*, which adds a penalty term proportional to the square of the magnitude of the coefficient vector.

The objective function for Ridge Regression is:

$$f(\beta) = ||X\beta - y||_2^2 + \lambda||\beta||_2^2$$

where $\lambda > 0$ is a scalar constant, $X \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^m$, and $\beta \in \mathbb{R}^n$.

1. Expand the term $||X\beta - y||_2^2$ into matrix-vector products (as done in the lecture notes).
2. Rewrite the penalty term $\lambda||\beta||_2^2$ using vector dot product notation (involving β^T).

3. Compute the gradient $\nabla_\beta f(\beta)$ with respect to β.

4. Set the gradient to zero and solve for the optimal $\hat{\beta}$. This result is known as the Ridge Estimator.

5. Compute the Hessian $\nabla^2_\beta f(\beta)$. Is the Hessian positive definite? (Assume $\lambda > 0$).

Problem 3: MLE for the Exponential Distribution (Chapter 3)

The Exponential distribution is often used to model the time until an event occurs. The probability density function (PDF) for a single observation x is:

$$p(x|\lambda) = \lambda e^{-\lambda x}$$

where $\lambda > 0$ is the rate parameter and $x \geq 0$.

Suppose we observe an independent and identically distributed (i.i.d.) dataset $D = \{x_1, x_2, \ldots, x_N\}$.

1. Write down the Likelihood function $L(\lambda)$ for the dataset D.

2. Write down the Log-Likelihood function $\ell(\lambda) = \log L(\lambda)$.

3. To find the Maximum Likelihood Estimate (MLE), computing the derivative $\frac{d\ell}{d\lambda}$ and set it to zero.

4. Solve for the optimal parameter $\hat{\lambda}$ in terms of the data points x_i.

5. Compute the second derivative $\frac{d^2\ell}{d\lambda^2}$ and verify that your solution corresponds to a maximum.

Part B: Programming Problems (JAX)

Problem 4: Gradient Verification for Ridge Regression

You derived the analytic gradient for Ridge Regression in Problem 2. Now you will verify it numerically using JAX.

Task:

1. Generate synthetic data:
   - Create a random matrix $X \in \mathbb{R}^{50\times5}$.
   - Create a random target vector $y \in \mathbb{R}^{50}$.
   - Create a random initial weight vector $\beta \in \mathbb{R}^{5}$.
   - Set the regularization parameter $\lambda = 10.0$.
2. Define the Ridge Regression loss function in JAX:
$$\text{loss}(\beta) = (X\beta - y)^T(X\beta - y) + \lambda\beta^T\beta$$
3. Use `jax.grad` to compute the gradient of this loss at your random β.
4. Implement the analytic gradient formula you derived in Problem 2 (Part 3) as a Python function.
5. Compute the analytic gradient using the same data.
6. Use `jax.numpy.allclose` to assert that the JAX-computed gradient and your analytic gradient are identical (within numerical precision). Print the result.

Problem 5: MLE Verification via Gradient Checking

In Problem 3, you derived the MLE for the Exponential distribution. Here, we will use JAX to confirm that the gradient of the Negative Log-Likelihood (NLL) is indeed zero at the analytic solution.

Task:

1. Generate synthetic data: Create an array of 100 samples drawn from an exponential distribution with a true rate $\lambda_{true} = 4.0$. (You can use `jax.random.exponential`, note that JAX usually parameterizes by scale $1/\lambda$, so adjust accordingly or generate simple uniform numbers and transform them).
2. Compute the analytic MLE $\hat{\lambda}$ using the formula you derived in Problem 3 (Part 4).
3. Define the Negative Log-Likelihood (NLL) function for the exponential distribution in JAX. (Note: Optimization usually minimizes, so we minimize negative log-likelihood).
4. Create a gradient function using `jax.grad` for the NLL.
5. Evaluate the gradient of the NLL at your analytic solution $\hat{\lambda}$.
6. Print the value of the gradient. It should be very close to 0. Explain why this confirms your derivation.