ECE 685D HW1

Submission Instructions:

1. Upload your Jupyter Notebook (.ipynb file).
2. Export all outputs and necessary derivations as a PDF or HTML (preferred) and upload them.

Mathematical Derivations:

1. You can submit handwritten derivations, but LaTeX is strongly encouraged.
2. Illegible handwriting will result in a 10% point deduction, and you will need to resubmit the work in LaTeX or Unicode within 2 days.

LLM policy:
The use of large language models (LLMs) is not allowed for this assignment.

1 Problem 1: Linear regression on a simple dataset(40 pts)

In this problem, you will implement a multi-linear regression model from scratch. Please download the **Concrete Strength regression** dataset from `https://www.kaggle.com/datasets/maajdl/yeh-concret-data`

A multi-linear regression model can be expressed as:

$$Y = X\hat{\beta} + \hat{\epsilon} \quad \text{where} \quad \hat{\epsilon} \sim N(0, \hat{\sigma} I)$$

In this formulation, $X \in \mathbb{R}^{N\times m}$ represents the matrix of independent variables, with each column corresponding to a different variable, and $Y \in \mathbb{R}^{N\times 1}$ is the column vector of the dependent variable to be predicted. The model is assumed to minimize the Mean Squared Error (MSE) loss, defined as:

$$\hat{\beta}_{\text{opt}} = \arg\min_{\hat{\beta}} ||Y - X\hat{\beta}||_2^2$$

Given the dataset, "concrete compressive strength" is the variable to be predicted. You will answer the following two questions:

1. Given the dataset with independent variables and a bias term, find $\hat{\beta}_0 \in \mathbb{R}^{9\times 1}$ that minimizes $||Y - X\hat{\beta}_0||_2^2$. You may only use matrix multiplication and inversion operation from the **Numpy** or **PyTorch** library to determine the value of $\hat{\beta}_0$.
2. Randomly divide the dataset into a training set (75%) and a validation set (25%). Train models with $\hat{\beta} \in \mathbb{R}^{i\times 1}$ for $i \in \{7, 8, 9\}$ using the training set (different values of i indicate different number of features to use). Then, compute the Mean

Squared Error (MSE) loss of your predictions on the validation set for each model. What do you observe from the MSE results of these models? Do models with more independent variables always perform better? You may select any combination of independent variables for your models.

2 Problem 2: Multinomial Logistic regression from pre-trained feature extractor (40pts)

In this problem, you will implement a logistic regression model from scratch to classify MNIST images using a pre-trained feature extractor. Follow these steps:

1. Load the pre-trained weights for the feature extractor. The architecture definition for the feature extractor is provided in the attached code.
2. Extract the latent representations $h \in \mathbb{R}^k$ from each MNIST sample $x \in \mathbb{R}^{784}$ using the pre-trained feature extractor.
3. Derive the gradients of the weight matrix $\mathbf{W}$ and bias b with respect to the cross-entropy loss between the true labels and the predictions of the model. The predictions are given by:
$$\hat{y} = \arg\max \sigma(\mathbf{W}^T h + b)$$
where $\sigma(\cdot)$ denotes the softmax function.
4. Use stochastic gradient descent (SGD), implemented from scratch using matrix operations only, along with the gradients obtained in Step 3, to optimize the weights $\mathbf{W}$ and bias b such that the cross-entropy loss is minimized.

You should **plot** both your training and validation loss throughout the training process.

3 Problem 3: Transformation from a uniform distribution (20pts)

The Box–Muller transform is a popular sampling method for generating pairs of independent, standard, normally distributed random variables from a pair of independent, standard uniformly random numbers. Let $U_1 \sim U(0,1)$ and $U_2 \sim U(0,1)$ and U_1, U_2 are independent, we define the random variable:

$$\Theta = 2\pi U_1, R = \sqrt{-2\ln(U_2)}$$

Show that the random variables

$$Z_1 = R\cos(\Theta) \ and \ Z_2 = R\sin(\Theta)$$

1. both follow standard normal distribution
2. are independent to each other

Lastly, make a plot to show the transformed distribution is indeed Gaussian. You should start with np.random.rand() to sample from a uniform distribution.