ECE 685D HW2

Submission Instructions

1. Upload your Jupyter Notebook (`.ipynb` file).
2. Export all outputs and necessary derivations as one or multiple PDF files 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 permitted for this assignment; if you use them, you **MUST** disclose how.

1 Problem 1: Backprop on a Residual MLP (30 pts)

We define f as a k-layer fully connected neural network (MLP) with *identity skip connections* at every hidden layer. Let $x \in \mathbb{R}^{m\times 1}$ and set $h_0 := x$. For $\ell = 1, \ldots, k-1$:

$$a_\ell = W_\ell^\top h_{\ell-1} + b_\ell, \tag{1}$$

$$h_\ell = g(a_\ell) \; + \; h_{\ell-1}, \tag{2}$$

where g is an element-wise 1-Lipschitz activation with derivative $g'(\cdot)$. The output layer remains linear:

$$\hat{y} \; = \; W_k^\top h_{k-1} + b_k. \tag{3}$$

Let $k = 3$ and use the mean-squared error (MSE) loss

$$L(Y, \hat{Y}) \; = \; \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

Tasks.

(a) Derive $\dfrac{\partial L}{\partial W_3}$ and $\dfrac{\partial L}{\partial b_3}$.

(b) Define $e_\ell := \frac{\partial L}{\partial h_\ell}$ and $\delta_\ell := \frac{\partial L}{\partial a_\ell} = e_\ell \odot g'(a_\ell)$ for $\ell = 1, 2$. Show the *residual recursion*

$$e_{\ell-1} \;=\; e_\ell \;+\; W_\ell\, \delta_\ell \quad \text{for } \ell = 1, 2.$$

(Hint: the skip adds an identity path $h_\ell \to h_{\ell-1}$ with derivative I.)

(c) Using (b), give expressions for $\dfrac{\partial L}{\partial W_2}$, $\dfrac{\partial L}{\partial b_2}$, $\dfrac{\partial L}{\partial W_1}$, and $\dfrac{\partial L}{\partial b_1}$. (The final result should not involve e_ℓ and δ_ℓ)

2 Problem 2: Customized *Multi-View* Dataset (30 pts)

In this problem, you will build a custom dataset `MNISTTwoView` by inheriting `torch.utils.data.Dataset`. For each sample, return *two independently augmented views* of the same image:

$$(x^{(1)},\, x^{(2)},\, y).$$

Both $x^{(1)}$ and $x^{(2)}$ are produced by applying the *same transform pipeline* with independent randomness (e.g., small random crop/resize, random affine, random erasing). Use any standard library (e.g., `torchvision` or `albumentations`). You can either download the MNIST dataset from Kaggle `https://www.kaggle.com/datasets/oddrationale/mnist-in-csv` or directly through torchvision. (Remark: such two view setting is commonly used in self-supervised learning)

Requirements.

(a) Your dataset must be compatible with `torch.utils.data.DataLoader`.

(b) Visualize one minibatch of size 8 as a grid with three columns per example: *original*, $x^{(1)}$, $x^{(2)}$. Show labels under each original image.

3 Problem 3: Logistic Regression using Gradient and Newton's Methods (30 pts)

In this problem, we are going to fit a Logistic Regression model on the Breast Cancer dataset: `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html` using gradient descent and Newton's methods. Since it's a binary classification problem, we will use the `BCELoss` from PyTorch. Specifically,

$$L(D; w) = \sum_{n=1}^{N} \Big[-y_n \log\big(\sigma(w^\top x_n)\big) - (1 - y_n) \log\big(1 - \sigma(w^\top x_n)\big) \Big], \tag{1}$$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}, \qquad D = \{x_n, y_n\}. \tag{2}$$

For this problem, you will compute the gradient $\frac{\partial L(D;w)}{\partial w}$ and the Hessian $\frac{\partial^2 L(D;w)}{\partial w_i \partial w_j}$ for the model weights. You can use the autograd tool in this problem.

Then, using the gradients and Hessian computed above, write a Python program to fit the Logistic Regression model on the Breast Cancer dataset using:

1. Gradient descent
2. Newton's method with exact expression for the Hessian

3. Newton's method with diagonal approximation for the Hessian

Plot and compare the loss, accuracy, and runtime graphs on the test set (learning rate = 0.1, test size = 30%) for all three methods. Briefly comment on the performance of the three methods.