Lahore University of Management Sciences
School of Science and Engineering, Department of Electrical Engineering
AI600 – Deep Learning
Spring 2026

Assignment 1

Date Assigned: Saturday, Feb. 7th, 2026 **Date Due:** Tuesday, Feb. 17th, 2026 11:55pm

Instructions

- While discussions with your peers is strongly encouraged, please keep in mind that the assignments is to be attempted on an individual level.
- In order to gain the maximum benefit, try to perform solo brainstorming before any consultations. The thought process behind searching for the correct approach all on your own will be invaluable.
- **Late Submission Policy.** Late homework solutions will be accepted with a 25% penalty till Wednesday, February 18, 11:55 pm. No solution will be accepted after this.
- Students must submit their answers, results, and findings in the form of PDF report through the LMS Assignments Tab.
- For coding tasks, students are required to create a public GitHub repository and provide its link in the first page of report as part of the submission.
- Students are welcome to use generative AI tools as partners in their learning journey. However, they are required to explicitly mention the prompt, its output, and edits made in the end of your report. For details, please refer to the AI policy listed on the course outline.

This assignment broadly focuses on the basics of Exploratory Data Analysis (EDA), Feedforward Neural Network and Multi-layer Perceptron (MLP).

Question 1: Feedforward Neural Networks using Tabular Data

[70 marks]

In this question, you will work with a real-world tabular dataset derived from Airbnb listings in New York City. Each row corresponds to one Airbnb listing, with the following columns:

- `neighbourhood_group`: NYC borough (categorical)
- `room_type`: Type of accommodation (categorical)
- `minimum_nights`: Minimum nights required for booking (numeric)
- `number_of_reviews`: Total number of reviews (numeric)
- `availability_365`: Number of available days in the next year (numeric)
- `amenity_score`: Listing amenity/quality score (numeric)
- `price_class`: Price category label in $\{0 = Budget, 1 = Moderate, 2 = Premium, 3 = Luxury\}$ listings.

The purpose of this question is not only to train a feedforward neural network, but also to develop a principled understanding of how data characteristics influence model behavior and generalization. You are provided with two files: `train.csv` and `test.csv`. The training file should be used for model development and validation, while the test file must be treated as a held-out dataset and used only for final evaluation. The target variable is `price_class`.

Part A: Data Preprocessing and Exploratory Analysis Before training any model, you must perform a thorough exploratory data analysis (EDA) on the provided training dataset. Your submission **must explicitly include the following**:

- Describe the dataset structure, including the number of samples, number of features and data types.
- Identify missing values (if any) for each feature and clearly state the strategy used to handle missing data and justify why your choice is a better approach in current scenario.
- Analyze and report the class distribution of the target variable. Include a visualization (e.g., bar chart) and comment on any imbalance.
- Encode categorical variables using an appropriate encoding scheme. Justify your choice.
- Normalize your numerical features and clearly state the method used for normalization, and why?
- Plot and analyze relationships between individual features and the target variable using suitable visualizations.
- Visualize the correlation matrix among numerical features. Identify and discuss highly correlated feature pairs.
- Based on the above analysis, explicitly state:
  - Which features you expect to be most influential for prediction.
  - Whether any feature appears unusually predictive or suspiciously dominant.

All claims must be supported by visual or quantitative evidence.

Part B (a): Two-Layer Perceptron Implemented from Scratch Implement a feed-forward neural network with **two hidden layers** using **only NumPy (or basic tensor operations)**. You are **not allowed** to use any automatic differentiation libraries.

Your model must satisfy the following constraints:

- Explicitly implement:
  - Forward propagation
  - Cross-entropy loss computation
  - Backward propagation using the algorithms and notations discussed in class
- First use (a) Sigmoid activation function, and then (b) ReLU activation function for all hidden layers; and briefly explain how gradient properties influence optimization and gradient flow.
- Train the network using **vanilla batch gradient descent** for **at least 200 iterations for both types of activation functions**.
- Report the final training and validation accuracy values for both activation functions and present a **single plot** showing the variation of training and validation accuracy as a function of the iterations.

Part B (b): Gradient Magnitude Comparison Across Layers Using your trained networks (with both activation functions) from Part B (a), perform the following analysis:

- Compute the average magnitude of gradients for the first and second hidden layer weights and compare these gradient magnitudes quantitatively as a function of the iterations. Do this for both types of activation functions.
  - What do you observe? And what do your results reveal about gradient flow and learning dynamics in deep networks.

Part C (a): Gradient-Based Feature Attribution (Analytical) In this part, you are required to perform a **purely analytical** modification of the backpropagation algorithm discussed in class. **No implementation or coding is required in this part.** Your task is to analytically derive the equation of gradient of the loss function with respect to each **input feature** and explain how it can be used for feature attribution. Specifically, your submission must include the following:

- Using symbolic derivations, compute the equation for gradient of the loss with respect to the input vector, i.e.,

  $$\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$$

  for the network architecture studied in class. Show step by step, how these gradients are obtained by modifying the standard backpropagation procedure.

- Write **hand-written pseudocode** (algorithmic steps) that outlines the analytical computation of gradients with respect to each input feature. Also show how the average magnitude of these gradients can be used to rank feature importance.

- Briefly justify, from an optimization perspective, why the magnitude of

  $$\left|\frac{\partial \mathcal{L}}{\partial x_i}\right|$$

  is a meaningful measure of feature contribution.

Note: All derivations and pseudocode in this part must be presented analytically on paper. No numerical experiments or programming are expected.

Part C (b): Implementation and Results In this part, you may either re-train a comparable two-hidden-layer MLP using PyTorch, or reuse the architecture from Part B and implement the required modifications. The architecture must remain comparable to the network used previously.

Go through Section 2.8 of Charu C. Aggarwal's Textbook.

- Implement your proposed gradient-based feature attribution method explained in Section 2.8 of the textbook.
- Implement your ranked list of the most influential input features.
- Briefly interpret the results and relate them back to your findings from Part A.
- For your report submission, provide a pseudo-code of your version of the implemented aglrothm.

Part D: Test Evaluation and Generalization Analysis Evaluate your trained PyTorch model on the provided test dataset and analyze its generalization behavior.

Your submission must include:

- The reported test accuracy.
- A comparison between training, validation, and test performance and evidence-based analysis explaining any observed performance gap.
- Identification of the root cause of generalization failure using:
  - Exploratory data analysis from Part A
  - Gradient-based feature attribution from Part C
- A clear explanation of why the model performs well during training but fails on the test set.
- Suggest **a strategy** that could mitigate this issue in practice, with brief justification.

Question 2: Bias Gradients, Parameter Sharing, and Activation Effects **[30 marks]**

This question evaluates your understanding of backpropagation at a symbolic level, with particular emphasis on *bias parameters that are shared across multiple layers.* You are required to derive gradients analytically using the chain rule and to reason carefully about how gradients propagate through a computation graph. **All derivations must be carried out by hand on paper or written in latex.**

Consider the feedforward neural network shown in Figure 1, defined for input–output pairs $(\mathbf{x}, \mathbf{y})$.

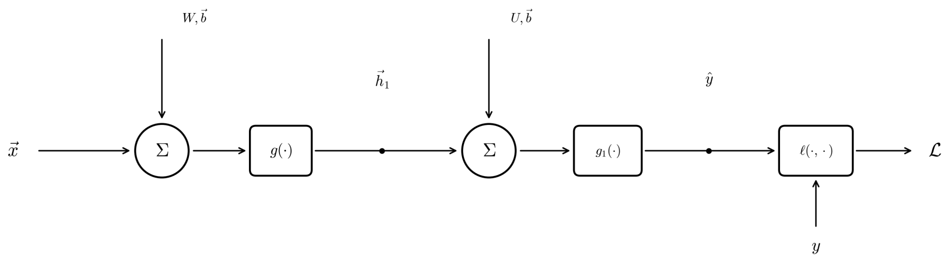

Figure 1: Computation graph for a two-layer MLP with a shared bias parameter.

The forward pass of the network is given by:

$$\mathbf{h}_1 = g(W\mathbf{x} + \mathbf{b})$$

$$\hat{\mathbf{y}} = g_1(U\mathbf{h}_1 + \mathbf{b})$$

$$L = \ell(\hat{\mathbf{y}}, \mathbf{y})$$

where:

- $\mathbf{x} \in R^d$ is the input feature vector,
- $\mathbf{y} \in R^k$ is the target vector,
- $W \in R^{m \times d}$ and $U \in R^{k \times m}$ are weight matrices,
- $\mathbf{b} \in R^m$ is a **single shared bias vector** used in both affine transformations,
- $g(\cdot)$ is an element-wise hidden-layer activation function
- $g_1(\cdot)$ is an element-wise output activation function,
- $\ell(\cdot, \cdot)$ is a differentiable loss function that maps $(\hat{\mathbf{y}}, \mathbf{y})$ to a scalar.

Shared Bias Gradient

- Modify the backpropogation algorithm discussed in class/notes to now compute the gradient vector
$$\frac{\partial L}{\partial b}.$$
for the case when biases are shared. For simplicity, the hidden and output dimensions are assumed to be equal, i.e., $m = k$.
- Your derivation must:
  - Apply the chain rule step by step, and clearly identify all computational paths through which the bias parameter b influences the loss.
  - Explicitly show how gradient contributions from multiple occurrences of the same parameter are combined.
- **Optimization and Convergence Effect:** Suppose we compare two models:
  1. **Shared-bias model:** the same bias parameter b is used in both layers (as in this question).
  2. **Independent-bias model:** each layer has its own bias parameter, i.e., b_1 and b_2.

  Do you expect bias sharing to *affect convergence speed or stability* during gradient-based training? State **yes or no** and justify your answer using reasoning grounded in optimization (e.g., parameter coupling, effective degrees of freedom, gradient magnitude scaling due to multiple paths, or conditioning). Your answer should be brief but logically consistent.

Report Submission Guidelines

You are required to submit a single PDF report containing all required components of the assignment.

- **Report Preparation:** Students are strongly encouraged to prepare their report using LaTeX. The report should be clearly structured, well-formatted, and easy to follow.
- **Programming Components:** All programming-related parts of the assignment must be published as a public **GitHub repository** with link provided on the **first page** of the report.
- **Analytical Questions:** Analytical questions can be typed directly in latex. Alternatively, students can paste the scanned copy of hand written-solution, and included as part of the PDF report.
- In the end of report, mention your usage of Generative AI tools, clearly stating
  - The exact prompt(s) used,
  - The corresponding output(s) generated by the tool, and
  - Any edits, or modifications you applied to the generated content.