ASSIGNMENT 1

Deadline: 11:59 pm, March 3, 2025

Submit via Blackboard with VeriGuide receipt.

Please follow the course policy and the school's academic honesty policy.

Submission Instructions:

Students must submit a zip file containing:

- A PDF file with their written solutions.
- The implemented `Assignment1_Q2.py` file.

For Problem 2, students must also either paste the output or include a screenshot of the output in the PDF file after executing their code.

1. Consider a feedforward neural network for text classification with 2-dimensional input vectors $\mathbf{x} = [x_1, x_2]$, a hidden layer of 3 ReLU neurons, and a 2-dimensional softmax output layer. The network uses cross-entropy loss and has no biases ($\mathbf{b}_1 = \mathbf{0}, \mathbf{b}_2 = \mathbf{0}$). Weight matrices are $\mathbf{W} \in \mathbb{R}^{3\times 2}$ (input-to-hidden) and $\mathbf{V} \in \mathbb{R}^{2\times 3}$ (hidden-to-output). Given a training example $(\mathbf{x}, y)$ with $\mathbf{x} = [1, -1]$ and true label $y = 0$ (0-indexed). The ReLU activation function for the hidden layer is defined as:

$$\mathrm{ReLU}(x) = max(0, x).$$

The softmax function for the output layer is defined as:

$$\hat{y}_k = \frac{\exp(z_k)}{\sum_{i=0}^{1} \exp(z_i)} \quad \text{for } k \in \{0, 1\} \quad \text{where } z = \mathbf{V}\mathbf{h}.$$

(1) **Loss Computation** (20 points)

Compute the hidden layer activations $\mathbf{h}$, output probabilities $\hat{\mathbf{y}}$, and cross-entropy loss L, given the following weights

$$\mathbf{W} = \begin{bmatrix} 0.4 & -0.1 \\ -1 & 0.6 \\ 0.5 & 1 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} 0.8 & 1.2 & -0.5 \\ 0.7 & -0.9 & 0.3 \end{bmatrix}.$$

(2) **Gradient Backpropagation** (20 points)

Derive the gradients of L with respect to all weights in $\mathbf{W}$ and $\mathbf{V}$. Present results as:

$$\frac{\partial L}{\partial \mathbf{V}} = \begin{bmatrix} \frac{\partial L}{\partial V_{11}} & \frac{\partial L}{\partial V_{12}} & \frac{\partial L}{\partial V_{13}} \\ \frac{\partial L}{\partial V_{21}} & \frac{\partial L}{\partial V_{22}} & \frac{\partial L}{\partial V_{23}} \end{bmatrix}, \quad \frac{\partial L}{\partial \mathbf{W}} = \begin{bmatrix} \frac{\partial L}{\partial W_{11}} & \frac{\partial L}{\partial W_{12}} \\ \frac{\partial L}{\partial W_{21}} & \frac{\partial L}{\partial W_{22}} \\ \frac{\partial L}{\partial W_{31}} & \frac{\partial L}{\partial W_{32}} \end{bmatrix}.$$

Explicitly show the chain rule steps for backpropagation.

2. In this assignment, you will implement core components of neural networks: a simple feed-forward Multi-Layer Perceptron (MLP) and the Cross Entropy Loss function. This will help you understand the fundamental building blocks of deep learning models.

   (1) **Implementing MLP Forward Pass** (30 points)

   Complete the `forward()` method in the MLP class. The MLP has two layers with the following specifications:

   - **First layer**: Linear transformation followed by ReLU activation
   - **Second layer**: Linear transformation (output layer)

   Your implementation should:

   - Apply the first linear transformation using `fc1_weight` and `fc1_bias`
   - Apply ReLU activation using `self.relu`
   - Apply the second linear transformation using `fc2_weight` and `fc2_bias`

   Note: Use `torch.mm()` for matrix multiplication.

   (2) **Implementing Cross Entropy Loss** (30 points) Complete the `my_cross_entropy_loss()` function that takes:

   - `outputs`: Raw logits from the model `[batch_size, num_classes]`
   - `labels`: Ground truth class indices `[batch_size]`

   Implementation steps:

   - Apply softmax to convert logits to probabilities
   - Extract the predicted probability for each correct class
   - Calculate negative log likelihood
   - Average over the batch

   Important:

   - Do not modify the random seed.
   - Do not import additional libraries.
   - Follow the TODOs in the comments of the attached Python file for implementation guidance.

*** END ***