# COSE474 Deep Learning ASSIGNMENT1 REPORT

MuYeong Jeong
Korea University
2019320112
jmy3033@naver.com

## Abstract

*In this assignment, the goal is to implement a two-layer fully connected neural network for MNIST classification. The neural network architecture is defined as follows:*

*Input → Fully Connected Layer → ReLU → Fully Connected Layer → Softmax*

*The required implementation includes the following components: Forward pass, Loss computation, Backward pass, Sampling for stochastic gradient descent (SGD), Parameter updating using gradients, Prediction output.*

## 1. Implement

I will explain the implementation process step by step.

### 1.1. Forward pass

$$z_1 = XW_2 + b_1 \quad (1)$$
$$p_1 = ReLU(z_1) \quad (2)$$
$$z_2 = p_1 W_2 + b_2 \quad (3)$$

The forward pass sequentially applies equations (1), (2), and (3) to compute the scores. Here, the scores are equivalent to $z_2$.

### 1.2. Loss computation

The loss was computed using softmax combined with negative log likelihood, along with L2 regularization.

$$Probabilities = softmax(z_2) \quad (1)$$
$$ansProbabilities = Probabilities(y = y_n|x_n) \quad (2)$$
$$regularizer = \lambda(\|w_1\|^2 + \|w_2\|^2) \quad (3)$$
$$L = -\frac{1}{N}\sum_{n=1}^{N} \log(ansProbabilities_i) + regularizer \quad (4)$$

### 1.3. Backward pass

The backward pass was computed following the methods described on pages 31–36 of the lecture note 04_neural network.

$$\frac{\partial L}{\partial z_2} = softmax(z_2) - y \quad (1)$$
$$\frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial z_2} W_2^T \odot 1_{(z_1>0)} \quad (2)$$
$$\frac{\partial L}{\partial W_2} = p_1^T \frac{\partial L}{\partial z_2} + 2\lambda W_2 \quad (3)$$
$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2} \quad (4)$$
$$\frac{\partial L}{\partial W_1} = X^T \frac{\partial L}{\partial z_1} + 2\lambda W_1 \quad (5)$$
$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \quad (6)$$

### 1.4. Sampling for SGD

Using np.random.choice, indices equal to the batch size were selected and stored in X_batch and y_batch.

### 1.5. Parameter updating

Parameters were updated by subtracting the product of the learning_rate and the computed gradients.

### 1.6. Prediction output

An array containing the class with the highest probability for each data point was stored in y_pred.

## 2. Result.

When applying the default model to the MNIST dataset, the validation accuracy was very low at 0.105. Therefore, an additional hyperparameter tuning process was conducted.

In parameter tuning, the learning rate was randomly chosen between $10^{-4}$ and $10^{-1}$, the learning rate decay between 0.8 and 1, and the reg between $10^{-5}$ and $10^{-2}$. The 10 parameter combinations were tested to find the best hyperparameters. The best combination was (learning rate = 0.09489, learning rate decay = 0.97922, reg = 0.00363). The validation accuracy increased to about 0.92.