1st Assignment: MLP Implementation

Daniel Jader Pellattiero
Korea University
danieljaderpellattiero@gmail.com

Abstract

The goal of the first assignment is to implement a two-layer neural network that performs image classification on the MNIST data set.

1. Introduction

The MNIST data set consists of 28×28 grayscale images of 70K handwritten digits ranging from (0–9). The goal is to train a classifier that correctly recognizes which digit is shown in each image. I have implemented a basic two-layer Multi-Layer Perceptron (MLP) with one hidden layer of size H:

$$\mathbf{z}_1 = X\mathbf{W}_1 + \mathbf{b}_1, \quad \mathbf{a}_1 = \mathrm{ReLU}(\mathbf{z}_1), \quad \mathbf{z}_2 = \mathbf{a}_1\mathbf{W}_2 + \mathbf{b}_2, \tag{1}$$

where X is the input matrix (flattened images), $\mathbf{W}_1, \mathbf{W}_2$ are the Iight matrices, $\mathbf{b}_1, \mathbf{b}_2$ are the biases, and ReLU is the activation function of the hidden layer.

Loss Function I compute the various classes' probabilities via softmax:

$$softmax(z_i)_c = \frac{\exp(z_{i,c})}{\sum_{k=1}^{C} \exp(z_{i,k})}, \tag{2}$$

and then I use the cross-entropy loss:

$$L_{data} = -\frac{1}{N} \sum_{i=1}^{N} \log(p_{i,y_i}) \tag{3}$$

where $p_{i,y_i} = softmax(z_i)_{yi}$. I also add an L2 penalty to our Iight parameters $\mathbf{W}_1, \mathbf{W}_2$ like so:

$$L_{reg} = -\frac{\lambda}{2}(||W_1||_2^2 + ||W_2||_2^2) \tag{4}$$

Train Function During training, I perform mini-batch Stochastic Gradient Descent (SGD). I firstly sample a random batch of images and labels and then I compute the forward pass to obtain the loss. Next, I perform backpropagation to calculate gradients w.r.t. all parameters—$\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, $\mathbf{b}_2$. Finally, I update these parameters via $\theta \leftarrow \theta - \eta \nabla_\theta L$, where η is the learning rate.

2. Methodology

First configuration Follows the initial configuration of the model's hyperparameters:

- Input size: $(28; 28)$
- Hidden size: 100
- Iterations: 1000
- Batch size: 200
- Learning rate: $1e^{-3}$
- Learning rate decay: 0.95
- Regularization: $1e^{-3}$

The performance of this first version of the neural network was not satisfactory as it reached a validation accuracy of 0.105 on the validation set. To address this, I automatically tested various values for the hidden layer size (H), learning rate (η), number of epochs, and regularization strength (λ). At the end I achieved a classification accuracy of over 39% on the validation set and over 50% on the test set, which is better than the baseline.

3. Performance and Hyperparameter Tuning

The most promising configurations tested, through an automated script, are reported below.

Model	H	η	λ	**Val. Acc.**
(Best)	**300**	**1e-1**	**1e-4**	**0.940**
Model B	100	1e-1	1e-4	0.939
Model C	300	1e-1	1e-3	0.939
(etc.)				

The batch size and learning rate decay remained the same, while the number of iterations was doubled.