Report of Training a Neural Network for MNIST dataset

Lab 4
of
Artificial Intelligence

Xiangyi Yan

11510706

1. Preliminaries

In project 4, a 4-layer fully connected neural network is developed for classifying hand written digits in MNIST dataset.

2. Methodology

Algorithms:

Initialization — Xavier Initialization:

The weight parameters are initialized as:

$$W \sim U\Big[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\Big]$$

n_j and n_{j+1} are the number of rows and columns of the weight matrixes.

Optimization — Back propagation:

Back propagation is a method used in artificial neural networks to calculate the error contribution of each neuron after a batch of data (in image recognition, multiple images) is processed. It is a special case of an older and more general technique called automatic differentiation. In the context of learning, **back propagation** is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function. This technique is also sometimes called **backward propagation** of errors, because the error is calculated at the output and distributed back through the network layers.

Activation Functions:

ReLU:

Rectifier Linear Unit is an activation function defined as the positive part of its argument:

$$f(x) = x^+ = \max(0, x)$$

where *x* is the input to a neuron.

Softmax:

Softmax function, or normalized exponential function is a generalization of the logistic function that "squashes" a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values in the range [0, 1] that add up to 1. The function is given by

$$\sigma : \mathbb{R}^K \to [0,1]^K$$
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Cross Entropy Loss:

A loss function or cost function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event.

Cross entropy loss is defined as:

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

L2 Regularization:

Regularization is a very important technique in machine learning to prevent overfitting. Mathematically speaking, it adds a **regularization** term in order to prevent the coefficients to fit so perfectly to overfit.

$$L = \frac{1}{N}\sum_{n=1}^{N} E_n(\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^2,$$

Mini-Batch Gradient Descent:

Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients.
Implementations may choose to sum the gradient over the mini-batch or take the average of the gradient which further reduces the variance of the gradient.
Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.

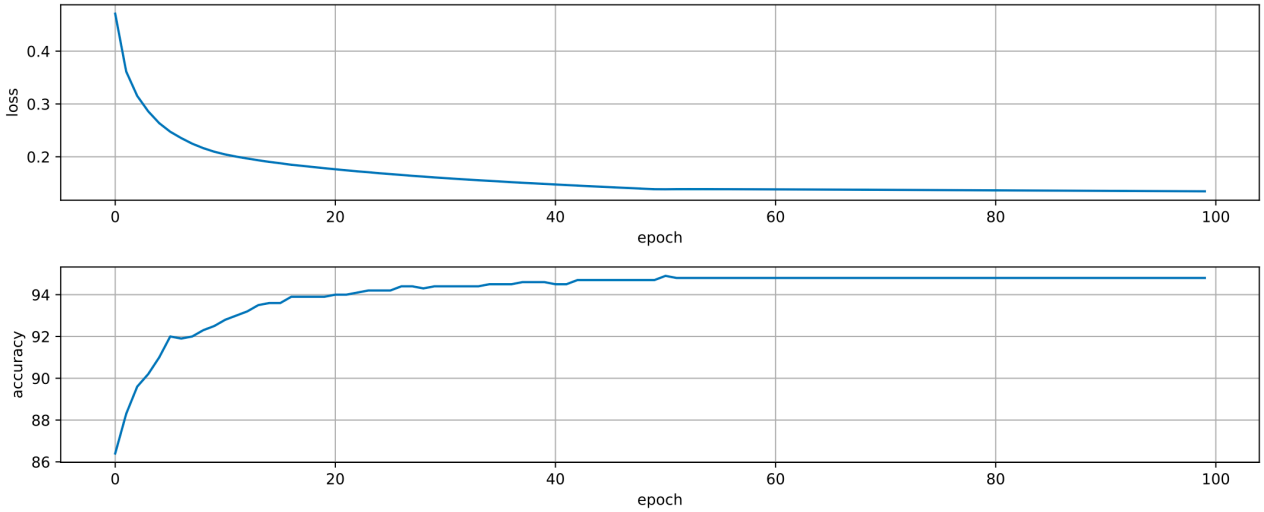
Table 1. Hyperparameters

Regulariztion parameter	**0.0005**
Learning Rate of First 50 Epoch	**0.1**
Learning Rate of Last 50 Epoch	**0.01**
Batch Size	**100**
Iterations	**100**
Epoch	**100**

3. Empirical Verification

The data used in this lab is from a famous dataset called MNIST, which is composed of many hand written digit images.

The following figure describes the loss and the test accuracy of the whole training procedure.



After 100 epochs, the accuracy of the test set (divided formerly) is 94.8% and the loss is around 0.134, which is a pretty great result.

The loss list and accuracy list have been stored into the corresponding txt files.

References

No references.