Handwritten Digital Recognition

Yue Leng 11510213

1. Preliminaries

Neural networks approach the problem, handwritten digital recognition in a good way. The idea is to take a large number of handwritten digits, known as training examples, and then develop a system which can learn from those training examples. In other words, the neural network uses the examples to automatically infer rules for recognizing handwritten digits. Furthermore, by increasing the number of training examples, the network can learn more about handwriting, and so improve its accuracy.

In this project, I wrote a computer program implementing a neural network that learns to recognize handwritten digits. This short program can recognize digits with an accuracy over 95 percent.

2. Methodology

2.1. Overview

The procedure of this algorithm is:

(i) Data processing

(ii) Weight initialization

(iii) Train of network

a. Forward propagation

b. Back propagation

c. Gradient update

d. Test for accuracy

(iv) Plot the loss, accuracy

2.2. Initialization — Xavier Initialization

The weight parameters are initialized as:

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

nj and nj+1 are the number of rows and columns of the weight matrixes.

2.3. 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.

2.4. Activation Functions

2.4.1. 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.

2.4.2. Softmax

Softmax function, or normalized exponential function is a generalization of the logistic function that "squashes" a Kdimensional 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

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

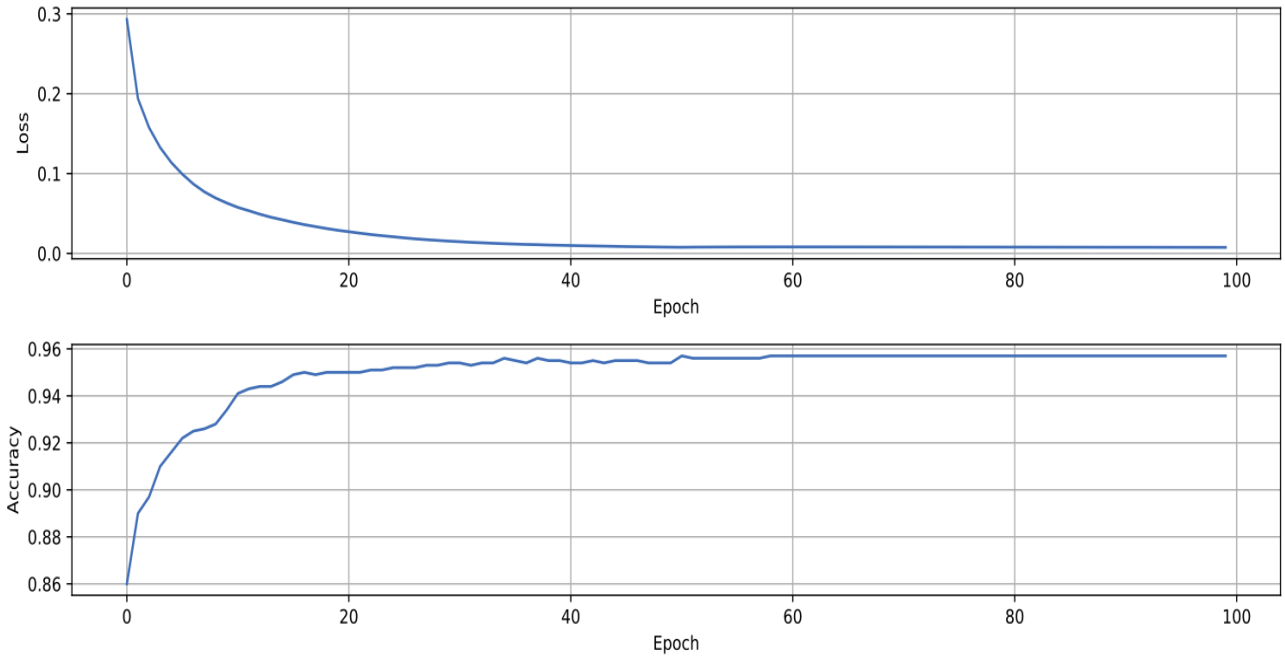
2.5. Mini-Batch Gradient Descent

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 minibatch 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.

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. The accuracy gets 94% when 15 epochs. After 100 epochs, the accuracy of the test set is more than 95%.



References

[1] Neural Networks and Deep Learning. Michael Nielsen, Dec 2017. http://neuralnetworksanddeeplearning.com/