SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Project for Mnist

Yuxing Hu
11510225

December 31, 2017

CONTENTS

1 PRELIMINARIES

1.1 INTRODUCTION

MNIST, the abbreviate of Modified National Institute of Standards and Technology, usually followed with the word 'database'.[1] Together, Mnist database is a large database of handwritten digits that is commonly used for training various image processing systems. [2] This project, however, is to find a algorithm to solve the recognition of handwriting, to make a maximum accuracy and minimize the train loss while stabilize the parameters. [3][4]

To achieve such goal, we are highly recommended to use the neural networking. Neural networking, also known as artificial neural networking, are computing systems inspired by the biological neural networks that constitute animal brains.ANN(artificial neural network) is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). The artificial neuron that receives the signal can process it and then signal artificial neurons connected to it.

1.2 SOFTWARE

The project is written in Python 3.6, as the guide document requires. I use Terminal to run and test the program and Sublime Text to write in. And I also use PyCharm to debug the program and see the structure of the project. I do not use any outer library except MatplotLib and NumPy which is given by template coding instruction. As for inner library built in Python I used including pickle and os(although not using).

2 METHODOLOGY

There are various function, method and algorithm for us to implement in this vary project, with exact one input layer with 784 neurons, 3 hidden layers and 1 output layers. The neurons number decrease from 784 to 300, 100 and finally we got 10 neurons to output. The vital strategies including relu algorithm, softmax algorithm, feed forward, back propagation etc.

2.1 GLOBAL VARIABLE

There are several global variables I used in this project to make things more clear and increasing the readability. These variables are listed as follows:

- EPOCH: total number of training round
- loss_list: a list to store the training loss data after each epoch
- accuracy_list: a list to store the accuracy data after each epoch
- lambda_value: the parameter to prevent overfitting
- learning_rate: learning step length for gradient descent
- after_learning_rate: learning step length for gradient descent in the last 50 epochs

2.2 Data Preprocess

This part is how we reformatting the data before further calculated or algorithm implementing. The data preprocessing part including two steps. First we need to normalize the dataset. The origin data is made of several data set and each of them has a list which length is 784. Means there are 784 neurons to represent a number.

Normalization process is transform them to a number value 0 to 1. To easier the accuracy step, I also transfer the label section from a decimal number to a 10-digit list to show the value of each number.

2.3 Weight Initialization

We need to from the weight number which is a $m * n$ dimension matrix. We use random method to generate them, and also made them into a uniform distribution. The method of creating these number is called Xavier initialization. In this way, the range of number will be in:

$$[-\frac{\sqrt{6}}{\sqrt{m+n}}, \frac{\sqrt{6}}{\sqrt{m+n}}] \tag{2.1}$$

2.4 Forward Propagation

In the calculation section, the major algorithm is forward and backward propagation. Between each two layers we need to implement the forward propagation to exercising, and after this processing, the output of first layer become the input of the second layer.

We also need activation function in the feed forward function, because we need to implement non linear transformation within this process. In the first two hidden layer we use relu function, and in the output layer we use softmax as the activation function. The forward propagation can be represent as Figure 2.2 and $Algorithm1$.

$$y(x, W, b) = W^T x + b \tag{2.2}$$

2.5 Back Propagation

For back propagation, we need to calculate the gradient of every layer, also update weights and bias for every layer that using the gradient. To calculate the gradient, we need to find out the projection loss of each layer and implement the entropy loss function, which is showed as following function. The weight and bias is the derivative value of entropy loss function. And the back propagation process can be represent as $Algorithm2$.

$$E(t, h) = -\sum(t \ln h^T + (1-t)\ln(1-h)) \tag{2.3}$$

Algorithm 1 Forward Propagation

Input: Training data, x

Output: Predicted value, h

a← x
for every hidden layer **do**
    z←feed forward a by weights and bias
    a←activate z by specified activation functions
end for
h←final activation values a
return h

Algorithm 2 Back Propagation

Input: Predict value, h; Training data labels, t

Output: Training loss data, l

calculate gradient of every layer gradient
update weight w and bias b backward from output layer to input layer using calculated gradient

2.6 Gradient update

After the forward and back propagation of the data value, we now have the new value of layer dataset. And we need to update the weight value and the bias value. The function of update is highly likely. Learning rate is represent by l, and$i$ represent different epoch iteration. The function can be described as follows:

$$W(B)^{(t+1)} = W(B)^t - l\Delta W(\Delta B)^t \tag{2.4}$$

2.7 Accuracy Test

The accuracy test is simpler than other parts of the project. The accuracy value is calculated by divided the right result after check by number of all the test. And the program will print the value of loss and accuracy and add them to the data list.

2.8 Plot

The primary code given to us has already wrote some sample code for us. All we need to do is add the title, labels, and set the data source properly. The output is a pdf format and is well reserved.

3 Empirical Verification

The empirical verification is conducted on a dataset that given to us. With the training set of 10000 and test data set of 1000. Some basic parameters are already be set for us. So the mainly verification part is about the time, training lost and test accuracy.

3.1 Design

The design of the test is pretty simple because there is no input format in this project we are suppose to use the given dataset. Even the path of that dataset is set for us. The out put is a pdf graph which represent the difference via every epochs.

3.2 Related Data

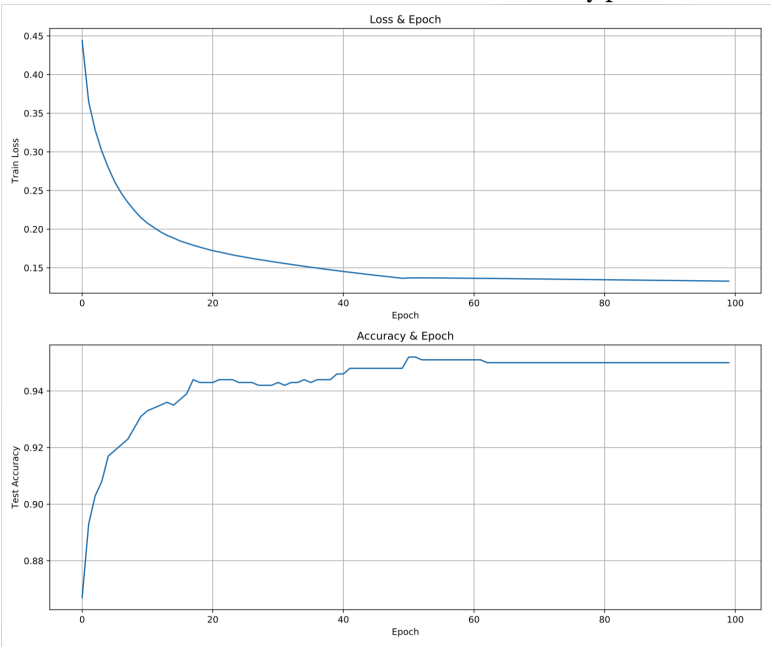
The network we used is full connection hidden neural network. The origin data is 10000 pictures and each of them is conducted with 28 * 28 numbers values in [0,255] to describe the dark value. The test data are as similar settings.

3.3 Performance Measurement

The running time is around 68s and all the component of the code has been gone through. So each round of the calculating is under a considerable time spend.

3.4 Results

The output graph showed the result very accurately, as we can see in round 1 the training loss is around 0.444035458924, and the accuracy is only about 0.867. In the final round the training loss comes to 0.132778255471 with stable 0.95 accuracy percise.



3.5 Further thinking

From the result we can observe that the final result is getting 95 percent accuracy while the train loss is still not comes to zero, which give us the direction of next step of working. There

are still lots of work to do, for instance, how to figure out the image that wrote completely out of order, which is much harder for program to read, on the other hand, people can easily read that out without any help? The neural network's program is still not the best implement to present, more straightforward function are there for us to discover,

REFERENCES

[1] Thomas R Gentile, Jeanne M Houston, and CL Cromer. Realization of a scale of absolute spectral response using the national institute of standards and technology high-accuracy cryogenic radiometer. *Applied Optics*, 35(22):4392–4403, 1996.

[2] Data Encryption Standard et al. Federal information processing standards publication 46. *National Bureau of Standards, US Department of Commerce*, 1977.

[3] Andrew Regenscheid and Karen Scarfone. Recommendations of the national institute of standards and technology. *NIST Special Publication*, 800:155, 2011.

[4] Robert R Zarr. A history of testing heat insulators at the national institute of standards and technology. *ASHRAE Transactions*, 107:661, 2001.