Report of program2

1. Experimental Setting

For this programming assignment, we are constructing a multi-classifier for the handwritten digit recognition, using a subset of the MNIST dataset. The data contains the 10 digits: from 0 to 9. The grayscale images are 28 $\times$ 28 and each pixel is an interger between 0 and 255. The dataset is split into 50000 training images and 10000 test images, each with the labels in a test file.

Our task is to train a 4-layer (2 hidden layers) fully connected neural network (NN) using the stochastic gradient descent method with mini-batch. This NN has one input layer (784 nodes), two hidden layers with 100 nodes respectively, and one output layer with 10 nodes. The ReLU activation function is used for the hidden nodes and softmax function, for the ouput layer.

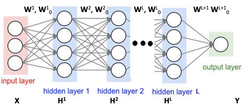
The only preprocessing of the input images is normalizing the image vectors by dividing by 255, so all the values are in [0,1]. This preprocessing can prevent the explosiion of the value of the pixels during the training. We are also instructed to convert $y[m]$ to a "1 of K" encoding, so that $y[m][k] = \mathbb{I}_{y=k}$, where $\mathbb{I}_{y=k}$ is the indicator function, i.e. equals 1 if y=k, and 0 otherwise. However, in my implement, I use an one-hot embed $\tilde{y}$ to simplify the calculation. The $\tilde{y}[m]$ is a $K \times 1$ vector that represent the probability that the input $x[m]$ belongs to class k.

We used Tensorflow to code the experiment. See the attached file: tensorcode.py.

2. Neural Network

The goal of this NN is to learn a mapping function which takes an input vector of features $x[m]$, and assigns it to one of K classes $y[m] = k \in \{1, \dots, K\}$. For the classification problem, we can use a probabilistic approach. In this assignment we used a discrimant method. The NN outputs a probability distribution over all possible value, $p(y = k|x)$, conditioned on the input x.

The model has four layers: input layer, hidden layer1, hidden layer2, output layer. See the diagram (chapter3, slide 6)



Here $X = (x_1, x_2, \dots, x_N)^\top$ represents input layer with N nodes. In this assignment N = 784. $Y = (y_1, y_2, \dots, y_k)^\top$ represents output layer with K nodes. In this assignment K = 10. $H^l = (h_1^l, h_2^l, \dots, h_{N_l}^l)$ represents lth hidden layer with N_l nodes. In this assignment, $l = 1, 2$ and the node number of the hidden layer (N_1 and N_2) are both 100. w^l is the weight matrix, with dimension as ($N_{l-1} \times N_l$), and w_0^l is the bias vector, the dimension of which is $N_l \times 1$.

General loss function:

Let the training data $D = \{x[m], y[m]\}, m = 1, 2, \dots, M$, and the parameters $\Theta = [w^1, w_0^1, w^2, w_0^2, w^3, w_0^3]$ and the whole NN as $\hat{y} = P_\Theta(y[m]|X[m])$. The loss function for this task can be defined as:

$$\begin{aligned}\mathcal{L}(D; \Theta) &= -\frac{1}{M}\sum_{m=1}^{M} l(\tilde{y}[m], \hat{y}[m]) \\ &= -\frac{1}{M}\sum_{m=1}^{M} l(\tilde{y}[m], P_\Theta(y[m]|X[m]))\end{aligned}$$

The forward propagation can be summarized as:*

1. From input layer to hidden1 layer: This is just a simple Linear regression model add an activation function. The reason we add the activation function is to add some nonlinearity to the NN, increasing the capacity of the model. The operations in this step can be described as:

$$z^1[m] = (w^1)^\top X[m] + w_0^1, \text{ where } z^1[m] \in \mathbb{R}^{N_1 \times 1}$$

$$h^1[m] = \max(0, z^1[m]), \text{ where } h^1[m] \in \mathbb{R}^{N_1 \times 1}$$

1. From hidden1 layer to hidden2 layer: This step is basically the step1, except the input is not $X[m]$ but $h[m]$.

$$z^2[m] = (w^2)^\top h^1[m] + w_0^2, \text{ where } z^2[m] \in \mathbb{R}^{N_2 \times 1}$$

$$h^2[m] = \max(0, z^2[m]), \text{ where } h^2[m] \in \mathbb{R}^{N_2 \times 1}$$

1. From hidden2 layer to output layer: In this step, we make the predicted values resemble a probability distribution. We use multi-class logistic regression, in which we apply the softmax activation function to the linear regression, so that the output values are all postive and sum to one, like a probability distribution:

$$z^3[m] = (w^3)^\top h^2[m] + w_0^3, \text{ where } z^3[m] \in \mathbb{R}^{K \times 1}$$

$$\hat{y}[m][k] = p([m] = k|x[m]) = \sigma_M(z^3[m][k]), \text{ where } \hat{y} \in \mathbb{R}^{K \times 1}$$

Note that $\sigma_M(z^3[m][k]) = \frac{\exp(z^3[m][k])}{\sum_{k'=1}^{K} \exp(z^3[m][k'])}$.

1. Loss layer: We want to have our predictive probability distribution $\hat{y}[m]$ to approximate the real target distribution $\tilde{y}[m]$ (as we define is the one-hot distribution of $y[m]$). One way is to minimize the cross entropy between these two distribution $\hat{y}[m]$ and $\tilde{y}[m]$:

$$l(\tilde{y}[m], \hat{y}[m]) = \sum_{k=1}^{K} -\tilde{y}[m][k] \log(\hat{y}[m][k])$$

Backpropagation of the NN

Here we shows the backpropagation for two layers: the output layer and the hidden layer.

From output layer to hidden 2

We can start with the output layer. As we mentioned, the NN output a probability distribution $\hat{y}[m]$ and to minimize the cross entropy between the target one-hot distribution $\tilde{y}$. So the gradient of the predictive probability $\bigtriangledown \hat{y}$ can be written as:

$$\bigtriangledown \hat{y} = \frac{\partial l}{\partial \hat{y}} \tag{1}$$

$$= -\frac{\tilde{y}}{\hat{y}} \tag{2}$$

This is a scalar by vector derivative and the $\bigtriangledown \hat{y}$ is a $K \times 1$ vectors.

For the output layer, the task is to compute the $\bigtriangledown h^2, \bigtriangledown w^3$, and $\bigtriangledown w_0^3$, given $\bigtriangledown \hat{y}$. Both w^3 and w_0^3 are the wegiths of the NN, so we need the gradient of them to update the weight of the NN. The reason to calculate the gradient of h^2 is that we need it to calculate the gradient of the weights of the early layers.

We first initiate with $\bigtriangledown w^3$.

$$\bigtriangledown w^3 = \frac{\partial z^3}{\partial w^3} \frac{\partial \hat{y}}{\partial z^3} \bigtriangledown \hat{y} \tag{3}$$

where $\frac{\partial \hat{y}}{\partial z^3}$ should be a $K \times K$ matrix. The derivation can be obtained by:

$$\frac{\partial \hat{y}}{\partial z^3} = \begin{bmatrix} \frac{\partial \hat{y}[1]}{\partial z^3} & \cdots & \frac{\partial \hat{y}[K]}{\partial z^3} \end{bmatrix}^{1 \times K} \tag{4}$$

$$= \begin{bmatrix} \frac{\partial \hat{y}[1]}{\partial z^3[1]} & \cdots & \frac{\partial \hat{y}[K]}{\partial z^3[1]} \\ \cdots & & \\ \frac{\partial \hat{y}[1]}{\partial z^3[K]} & \cdots & \frac{\partial \hat{y}[K]}{\partial z^3[K]} \end{bmatrix}^{K \times K} \tag{5}$$

$$= \frac{\partial \hat{y}[j]}{\partial z^3[i]} = \begin{cases} \hat{y}[i](1 - \hat{y}[i]) \text{ if } i = j \\ -\hat{y}[i]\hat{y}[j] \text{ otherwise} \end{cases} \tag{6}$$

And $\frac{\partial z^3}{\partial w^3}$ should be a $N_2 \times K \times K$. Using the same technique:

$$\frac{\partial \hat{y}}{\partial z^3} = \begin{bmatrix} \frac{\partial z^3[1]}{\partial w^3} & \cdots & \frac{\partial z^3[K]}{\partial w^3} \end{bmatrix}^{1 \times K} \tag{7}$$

$$\text{For each element } \frac{\partial z^3[k]}{\partial w^3} \tag{8}$$

$$= \begin{bmatrix} \frac{\partial z^3[k]}{\partial w^3[1][1]} & \cdots & \frac{\partial z^3[k]}{\partial w^3[1][K]} \\ \cdots & & \\ \frac{\partial z^3[k]}{\partial z^3[N_2][1]} & \cdots & \frac{\partial z^3[k]}{\partial z^3[N_2][1]} \end{bmatrix}^{N_2 \times K} \tag{9}$$

$$= \frac{\partial z^3[k]}{\partial z^3[][j]} = \begin{cases} h^2 \text{ if } j = k \\ \vec{0} \text{ otherwise} \end{cases} \tag{10}$$

Combine each of these elements, we can calculate $\bigtriangledown w^3$ which is a $N_2 \times K$ matirx:

$$\bigtriangledown w^3 = \begin{bmatrix} \frac{\partial l}{\partial w^3[1][1]} & \cdots & \frac{\partial l}{\partial w^3[1][K]} \\ \cdots & & \\ \frac{\partial l}{\partial w^3[N_2][1]} & \cdots & \frac{\partial l}{\partial w^3[N_2][K]} \end{bmatrix}^{1 \times K} \tag{11}$$

$$\text{For each element, } \frac{\partial l}{\partial w^3[i][k]} = h^2[i](\hat{y}[k] - \tilde{y}[k]) \tag{12}$$

Due to this equation, we can also write $\bigtriangledown w^3 = h_2(\hat{Y} - \hat{Y})^\top$. In fact, this equation can also be used in mini-batch situation. In the real implementation, I use this equation.

We then discuss $\bigtriangledown w_0^3$.

$$\bigtriangledown w_0^3 = \frac{\partial z^3}{\partial w_0^3} \frac{\partial \hat{y}}{\partial z^3} \bigtriangledown \hat{y} \tag{13}$$

The gradient is $K \times 1$ dimension. The last two terms are the same with that in $\bigtriangledown w^3$ and $\frac{z^3}{w_0^3} = 1$. So $\bigtriangledown w_0^3 = (\hat{Y} - \hat{Y})$.

Now, we turn to $\bigtriangledown h^2$.

$$\bigtriangledown h^2 = \frac{\partial z^3}{\partial h^2} \frac{\partial \hat{y}}{\partial z^3} \bigtriangledown \hat{y} \tag{14}$$

The gradient is $N_2 \times 1$ dimension. The last two terms are the same with that in $\bigtriangledown w^3$ and $\frac{z^3}{h^2} = w^3$. So $\bigtriangledown w_0^3 = w^3(\hat{Y} - \hat{Y})$.

From hidden2 layer to hidden 1

The task here is to comput $\bigtriangledown w^2, \bigtriangledown w_0^2, \bigtriangledown h^1$, given $\bigtriangledown h^2$

Still we can start with $\bigtriangledown w^2$.

$$\bigtriangledown w^2 = \frac{\partial z^2}{\partial w^2} \frac{\partial h^2}{\partial z^2} \bigtriangledown h^2 \tag{15}$$

This is a $N_1 \times N_2$ matrix. Among then $\frac{\partial h^2}{\partial z^2}$ is a $N_2 \times N_2$ matrix

$$\frac{\partial h^2}{\partial z^2} = \begin{bmatrix} \frac{\partial h^2[1]}{\partial z^2} & \cdots & \frac{\partial h^2[N_2]}{\partial z^2[1]} \\ \cdots & & \\ \frac{\partial h^2[1]}{\partial z^2[N_2]} & \cdots & \frac{\partial h^2[N_2]}{\partial z^2[N_2]} \end{bmatrix}^{N_2 \times N_2} \tag{16}$$

$$\text{For each element } \frac{\partial h^2[j]}{\partial z^2[i]} = \begin{cases} 1, \text{ if } z^2[i] > 0 \text{ and } i = j \\ 0 \text{ otherwise} \end{cases} \tag{17}$$

And $\frac{\partial z^2}{\partial w^2}$ is $N_1 \times N_2 \times N_2$. Referring to the derivation of $\frac{\partial z^3}{\partial w^3}$, we know that each element $\frac{\partial z^2[k]}{\partial w^2[i][j]} = h^1[i]$ if $j = k$, 0 otherwise. Now we can write

$$\bigtriangledown w^2[i][j] = \begin{cases} h^1[i]h^2[j](\hat{Y}[j] - \hat{Y}[j]), \text{ if } z[i] > 0 \\ 0, \text{ otherwise} \end{cases} \tag{18}$$

And the matrix form $\bigtriangledown w^2 = h^1(\frac{\partial h^2}{\partial z^2} h^2(\hat{Y} - \hat{Y})^\top$

Then $\bigtriangledown w_0^2$:

$$\bigtriangledown w_0^2[i] = \begin{cases} (\hat{Y} - \hat{Y}), \text{ if } z[i] > 0 \\ 0, \text{ otherwise} \end{cases} \tag{19}$$

Then $\bigtriangledown h^1$:

$$\bigtriangledown h^1 = \sum_i w^2[][i] \frac{\partial h^2}{\partial z^2} w^3(\hat{Y}[j] - \hat{Y}[j]) \tag{20}$$

In [ ]: