NAME: DUONG DUY CHIEN

Multi-class Classification and Neural Networks

I. Theory:

1. Multi-class Classification

Hypothesis:

$$h_\theta(x) = g(\theta^T x) = g(\theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n)$$

$$=g(z)=\frac{1}{1+e^{-z}}$$

Cost function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)})] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Gradient descent algorithm:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for j =0}$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \quad + \frac{\lambda}{m}\theta_j \text{ for j } \geq 1$$

The way to

with m= number of samples

j= 1....n : number of feartures

implement by vectorization is quite similar to the homework 1 so that I won't mention it again in here.

One-vs-all Classification

When we train the network for each class the output is the probability from 0 to 1, so if its belong to the class we are training, the output should be close to 1. if it belongs to others 9 class the output should be train to zero.

For testing: we will check each samples in 10 class and choose the class which has the highest probability.

2. Neural Networks

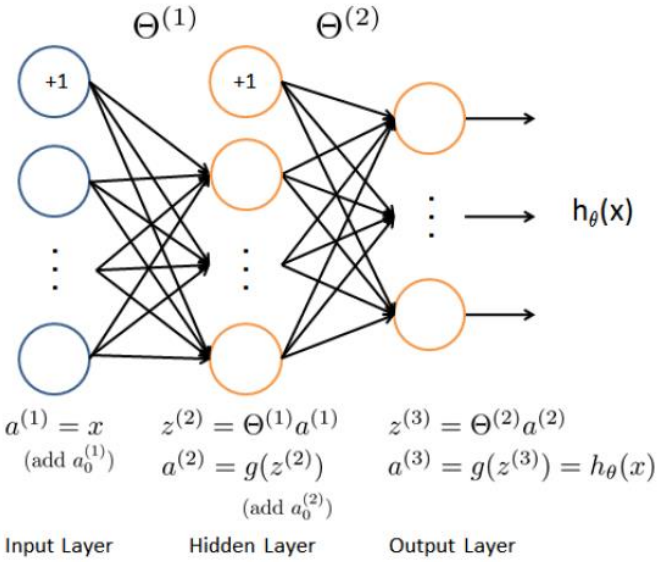

Figure 1 Neural Network model

II. Results:

1. Multi-class Classification

- Our training dataset includes 5000 sample images with size 20x20. So I need to reshape the input data become a vector 5000x400 (each row of vector is one image). Particularly, we are assuming that the image x_i has all of its pixels flattened out to a single column vector of shape (Dx1). I also visualize random 100 input images (as shown in figure 1)

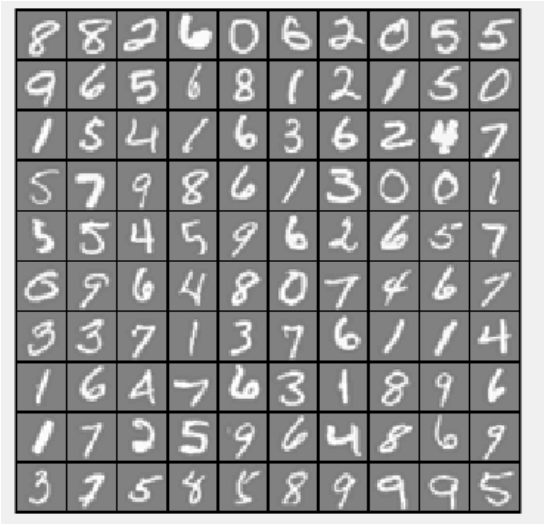

Figure 2 visualize random 100 images from training dataset

- After calculate cost function and gradient with regularization I got the result (as shown in figure 3) then apply one-vs-all Prediction to training the input image and got the accuracy is 95,04%

```
Cost: 2.534819
Expected cost: 2.534819
Gradients:
 0.146561
 -0.548558
 0.724722
 1.398003
Expected gradients:
 0.146561
 -0.548558
 0.724722
 1.398003
```

Figure 3 The cost and gradient of logistic regression

2. Neural Network

After using feedforward technique with trained parameters I got the accuracy is 97,52%. The output is also show some pictures of samples and the predict of network (as shown in figure 4)

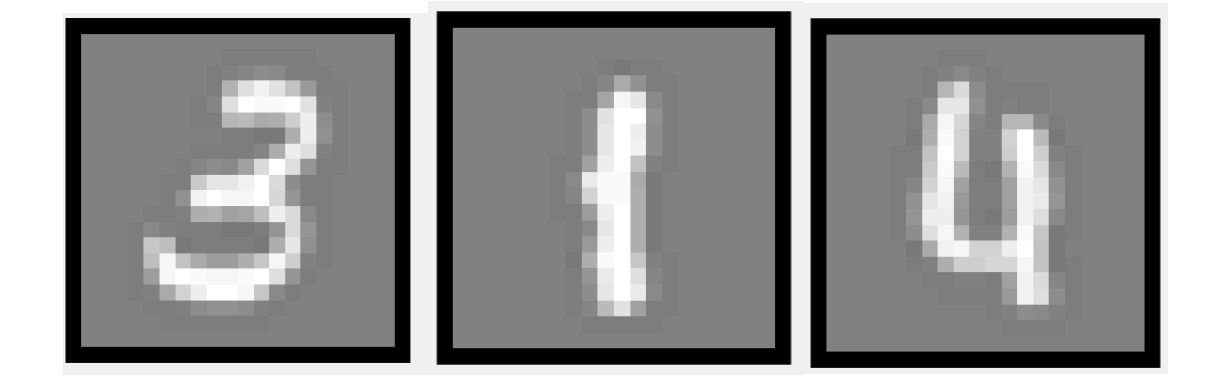

Figure 4 Display example images and output of neural network