Assignment #01
Neural Network and Deep Learning

Meng-Jiun Chiou
Department of Computer Science, National University of Singapore

1. Question #1

Two Neural Network are said to be identical if all of their input and output are the same. The aim of this question is to demonstrate how two networks can be essentially the same by choosing correct parameters, i.e. weights and biases.

Network two is a simple one layer neural network that multiply input by the weight followed by adding bias. Network two can be described in following equation:

$$z' = z * \widetilde{W} + \hat{b} \tag{1}$$

On the other hand, Network one is a three-layer neural network. By using the same method, we can obtain following equations:

$$\begin{aligned} z' &= \Big(\big(z * W_{0,1} + b_1\big) * W_{1,2} + b_2\Big) * W_{2,3} + b_3 \\ &= \big(z * W_{0,1} * W_{1,2} + b_1 * W_{1,2} + b_2\big) * W_{2,3} + b_3 \\ &= z * (W_{0,1} * W_{1,2} * W_{2,3}) + (b1 * W_{1,2} * W_{2,3} + b_2 * W_{2,3} + b_3) \end{aligned} \tag{2}$$

By comparing equations (1) and (2), we can observe $\widetilde{W}$ and $\hat{b}$:

$$\widetilde{W} = W_{0,1} * W_{1,2} * W_{2,3}$$
$$\hat{b} = b1 * W_{1,2} * W_{2,3} + b_2 * W_{2,3} + b_3$$

2. Question #2

a. Problem Definition

There are three networks need to be implemented in this question. The first one is a three-layer network containing two hidden layers, which we denote as **14-100-40-4 net**. The second one is 7-layer, containing six hidden layers and is denoted as **14-28*6-4 net**. The final one is the longest network, 29-layer and contains 28 hidden layers which denoted as **14-14*28-4 net**.

The layers are all fully connected layer with **ReLU** activation, except for output layer, a softmax classifier (multinomial logistic regression). ReLU is a nonlinear function:

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

On the other hand, softmax function has the form:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Combining the softmax function with cross-enptopy loss, we can squeeze the output vector into probability distribution between 0 and 1. The prediction can be made by picking the class corresponding to the highest probability.

The input of all networks is set to be a 14-digit number array, and the output is one of four numbers 1, 2, 3, 4.

$$f: \{-1, 1\}^{14} \rightarrow \{0, 1, 2, 3\}$$

To facilitate the training, I transform the output into a 4-dimension hot vector:

$$0 \rightarrow [1, 0, 0, 0]$$
$$1 \rightarrow [0, 1, 0, 0]$$
$$2 \rightarrow [0, 0, 1, 0]$$
$$3 \rightarrow [0, 0, 0, 1]$$

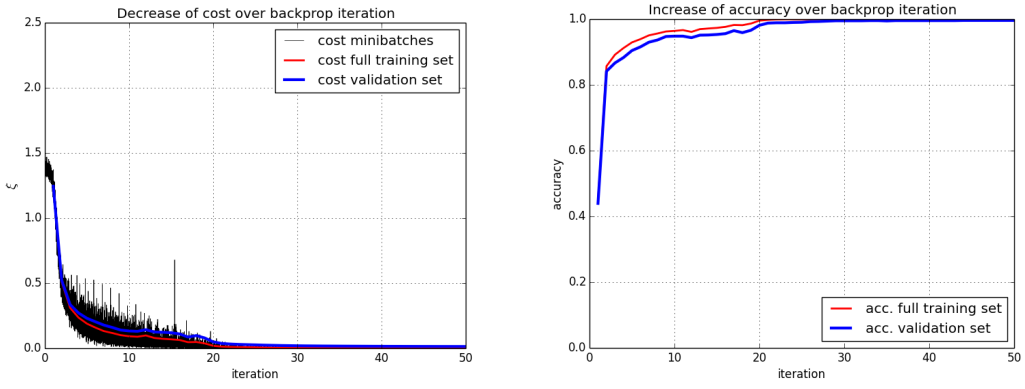
The programs are implemented in *Python* with plotting package *matplotlib*, machine learning package *scikit-learn* for easily dividing training set into training/validation set and calculating accuracy and some built-in packages.

The original training set includes 13,107 data points. I experiment with training set/validation set ratio of 0.4.

b. Training Result

Each iteration goes through the whole training set. To avoid overfitting, the early stop was applied when validation error does not decrease for three consecutive iterations. The update the parameter, I use mini-batch stochastic gradient descent. The batch size is set to 25 by default. Learning rate is set to a number within 0.01 and 0.001.

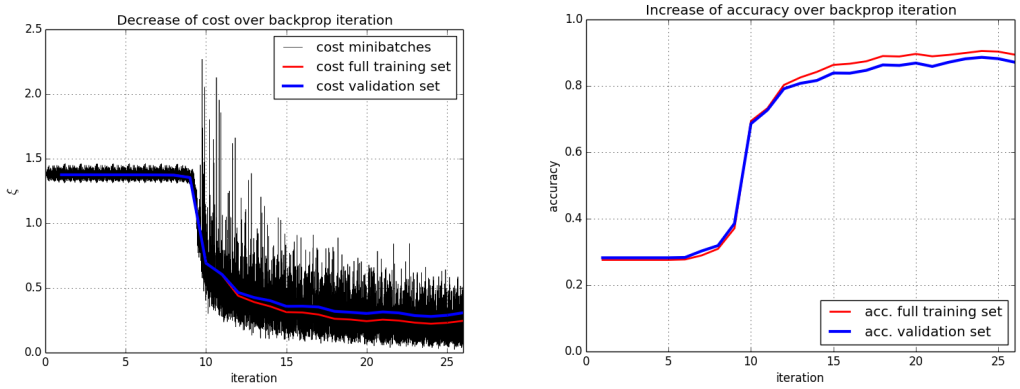
The experiment shows that shallow network works better. In the first network (figure 1), accuracy increased to 0.84 and the cost decreased to 0.4 after the second iteration, showing that the training procedure is fast. When testing, the model achieved the accuracy of 0.9954, the highest in three networks.



(a) (b)

Fig.1 (a) Cost decrease and (b) accuracy increase through iteration in the first net **14-100-40-4 net**. Final test accuracy is 0.9954 after 50 iterations.

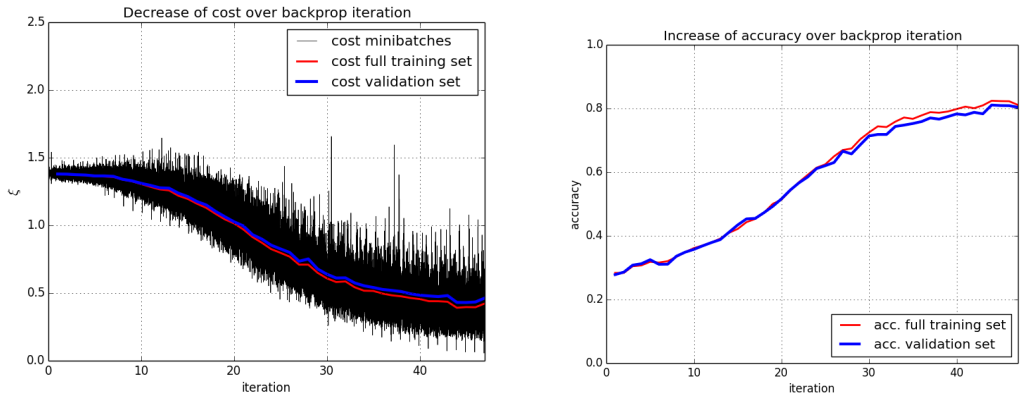
Training of the second network (Figure 2) was early stopped as validation error did not decrease for three consecutive iterations. The cost started to decrease late than the first network and saturate at a higher value. In addition, lower accuracy was observed both in training and test.



(a) (b)

Fig.2 (a) Cost decrease and (b) accuracy increase through iteration in the second net **14-28*6-4 net**. Final test accuracy is 0.8673 after 26 iterations.

However, even worse testing accuracy 0.8125 was observed in the third network (Figure 3). As it is the deepest in these three networks, it is more difficult to train since the backproped gradients are vanishing along with increasing number of layers. Also, early stop happened in the third training procedure.



(a) (b)

Fig.3 (a) Cost decrease and (b) accuracy increase through iteration in the second net **14-14*28-4 net**. Final test accuracy is 0.8125 after 47 iterations.

As the input and output is simple compared to image recognition or natural language processing problem, the experiment result indicated it is easier to obtain higher performance by training on simple neural network rather than deeper one.