CS5242 Assignment 1

Name: Pei Yingchi

Matric No: E0336111

Quenstion 1

With identity function as the activation function, a multi-layered neural network with only fully connected linear layers can be represented using one linear layer. As for each layer, we have

$$Z^{[l]} = W^{[l]} * Z^{[l-1]} + b^{[l]}$$

where $Z^{[l]}$ is the activation of layer l .

$W^{[l]}$ means the weights from layer $l-1$ to layer l .

We define layer 0 as the input layer, so $Z^{[0]} = X$.

With the given 3 layer network, we have

$$\begin{aligned} Z^{[3]} &= W^{[3]} * Z^{[2]} + b^{[3]} \\ &= W^{[3]} * (W^{[2]} * Z^{[1]} + b^{[2]}) + b^{[3]} \\ &= W^{[3]} * (W^{[2]} * (W^{[1]} * Z^{[0]} + b^{[1]}) + b^{[2]}) + b^{[3]} \\ &= W^{[3]} * (W^{[2]} * W^{[1]}) * X + W^{[3]} * W^{[2]} * b^{[1]} + W^{[3]} * b^{[2]} + b^{[3]} \end{aligned}$$

That is

$$\begin{aligned} \widetilde{W} &= W^{[3]} * W^{[2]} * W^{[1]} \\ \tilde{b} &= W^{[3]} * W^{[2]} * b^{[1]} + W^{[3]} * b^{[2]} + b^{[3]} \end{aligned}$$

Question 2

In this question, to train the network with softmax output layer, we will firstly need to one-hot encode the y labels provided. Specifically, for y values, we have the following:

0 → [1, 0, 0, 0]

1 → [0, 1, 0, 0]

2 → [0, 0, 1, 0]

3 → [0, 0, 0, 1]

The training data set provided has 13107 data points. Applying the common train-validation-test practice, I splited the training data provided into a smaller training data set and another validation data set, and use the test data provided as the test dataset. Thus the trained neural network will never see the test data before getting the test accuracy. The validation data is about 30% of the total provided training data.

All the network functions are in `q2_ref.py`.

Training Results

To make the results comparable, I used the same set of hyper-parameters for all the 3 networks.

Minibatch gradient descent is used. The hyper-parameters I used are:

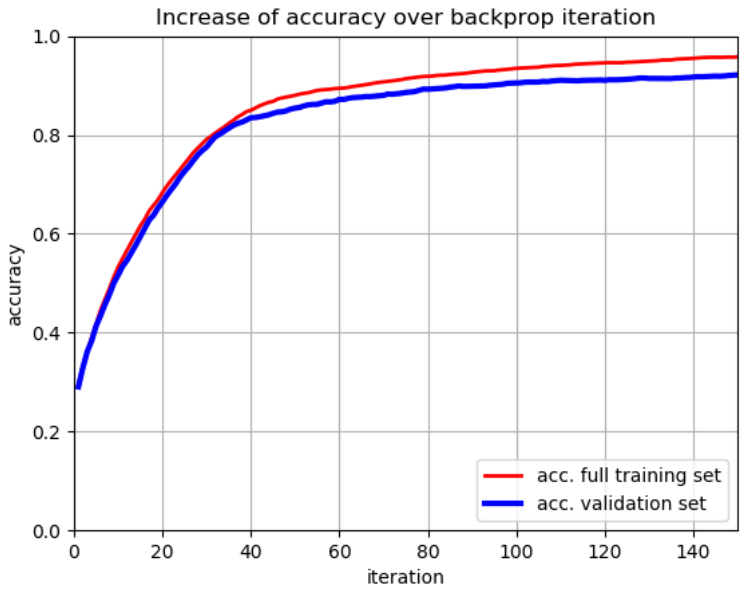
```
batch_size = 32
max_num_iterations = 150
learning_rate=0.003
```

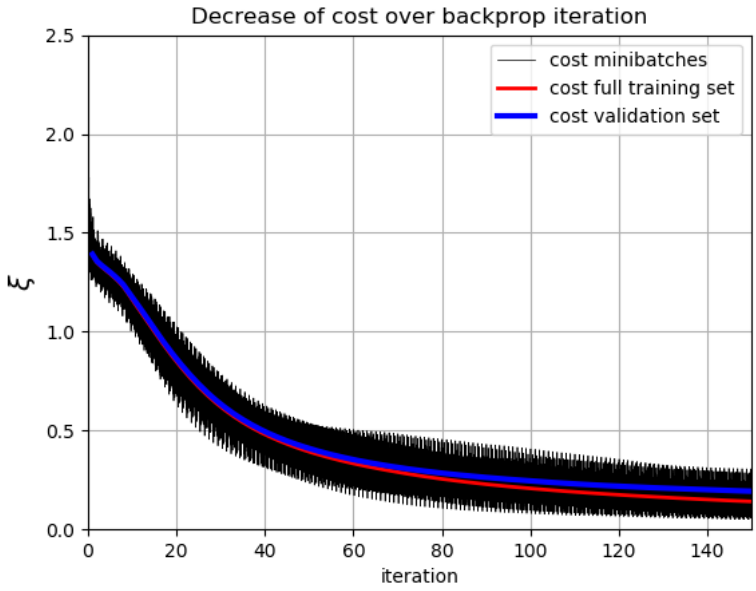
If the validation cost does not decrease for 5 consecutive iterations, early stop will be applied.

I have run the 3 networks using different random state, and overall, the performance in terms of test accuracy is: Network 1 > Network 2 > Network 3. Below is one set the performance:

Network 1

Test accuracy: 0.9130

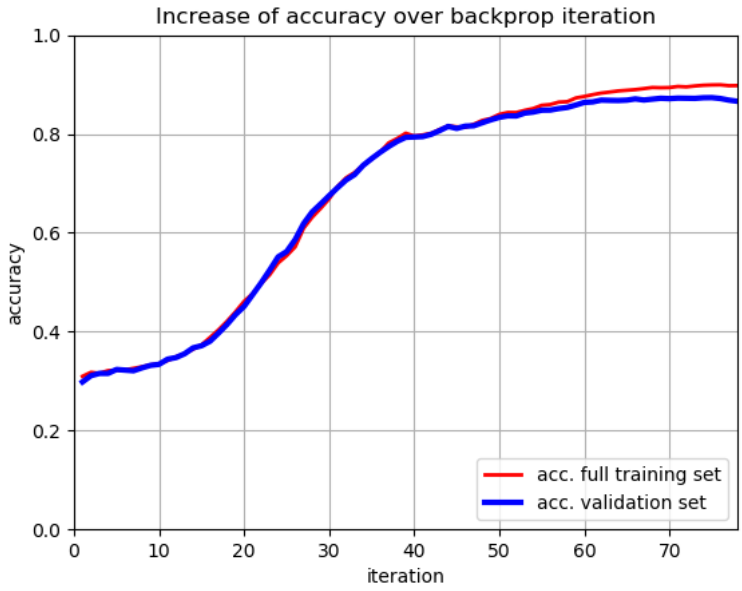


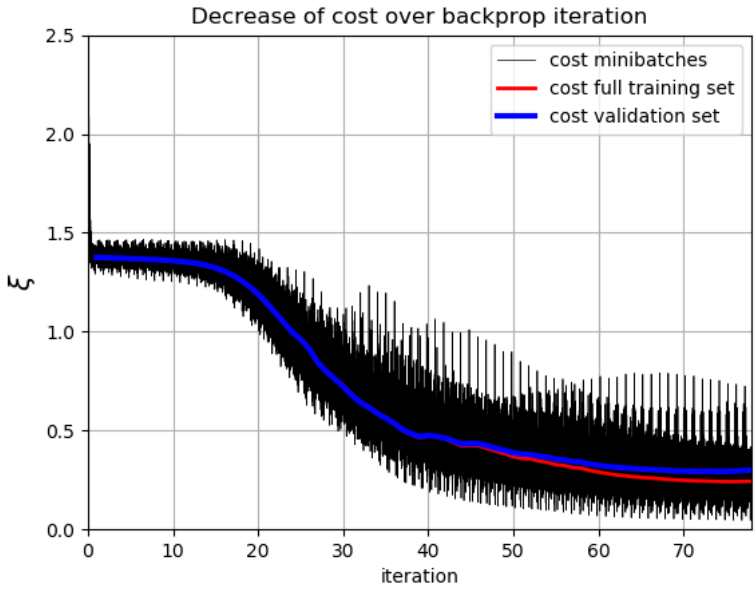


The performance improves pretty fast at the beginning of the iterations.

Network 2

Test accuracy: 0.8679

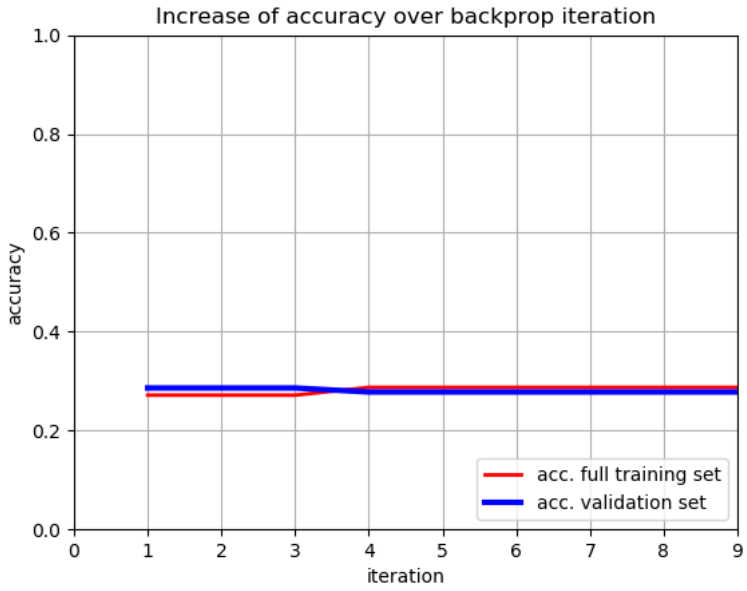


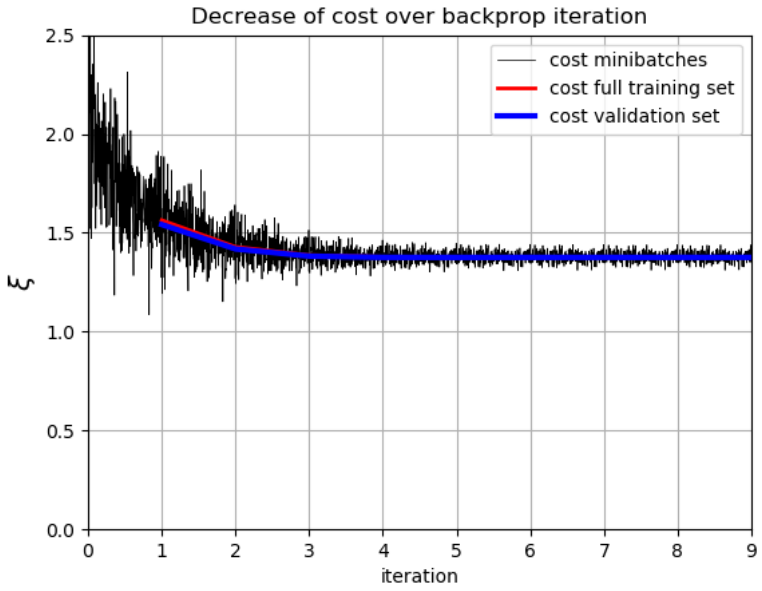


The improvement is not very obvious at the beginning of the iterations, but in the later part, we can see relatively large improvements.

Network 3

Test accuracy: 0.2887





Training for this netowkr was early stopped as the validation cost did not descrease for a while. The network training process seems stalled. When I tried to reduce the learning rate to 0.001, there is no improvement in the test accuracy and the training process seems to stop at the same level,

Possible reasons for net3:

1. Too many layers. The network is too deep. Without regularization, we will have a high change facing the vanishing gradient problem. Indeed, if we look at the intermediate gradients output, we will find that the absolute value of the gradients are way too large.
2. Another possible reason is the number of neurons in each hidden layer is not enough to capture the valuable information in the early layers.

For simple input data (a few dimensions), we should try with simple and shallow networks first. And maybe consider more number of neurons in early hidden layers to capture more information.