Tianyang Zhang
404-743-024

Question 1:
Modify the script to print the accuracy of the learned model on the training set. What is the accuracy you got? Why is it different from the test set accuracy?

By modify the script as:

```
accuracy_on_train = sess.run(accuracy, feed_dict={x: mnist.train.images,
                                                  y_: mnist.train.labels})
print(f'Accuracy on the test set: {accuracy_on_train}\n')
```

And ran the script 5 times, I obtained that the accuracy of the learned model on the training set is about same, but most times lower than the accuracy on testing set:

```
Accuracy on the testing set: 0.9186000227928162

Accuracy on the training set: 0.916527271270752
```

```
Accuracy on the testing set: 0.9193999767303467

Accuracy on the training set: 0.9183090925216675
```

```
Accuracy on the testing set: 0.9190999865531921

Accuracy on the training set: 0.9177272915840149
```

```
Accuracy on the testing set: 0.9172000288963318

Accuracy on the training set: 0.9159818291664124
```

The reason behind is that the train set constains more complex module (complex train cases) than test set, because we want the learned model has the ability to deal with complex cases. Thus, it would be harder for the learned model to process the train set than the test set, and would has little bit lower accuracy on the train set.

Question 2:
In Line 62 in mnist softmax.py, we loop 1000 times, taking small steps towards our final optimized model. Try changing 1000 to 10, and reprint the accuracy on the test set. What is the accuracy you got? Now try increasing the number of steps to 10000, and report the accuracy in this case. Comment briefly on the results.

When looping 10 times, the accuracy of the learned model on both testing and training set are dropped to around 75%:

```
Accuracy on the testing set by 10 times looping: 0.7444999814033508

Accuracy on the training set by 10 times looping: 0.7400909066200256
```

```
Accuracy on the testing set by 10 times looping: 0.7551000118255615

Accuracy on the training set by 10 times looping: 0.7475818395614624
```

When looping 10000 times, the accuracy of the learned model on both testing and training set are increased around 1%, which is very small. But now the accuracy of the learned model on training set is higher than accuracy on testing set:

```
Accuracy on the testing set by 10000 times looping: 0.9208999872207642

Accuracy on the training set by 10000 times looping: 0.9291090965270996
```

```
Accuracy on the testing set by 10000 times looping: 0.9207000136375427

Accuracy on the training set by 10000 times looping: 0.9273454546928406
```

Tianyang Zhang
404-743-024

Question 3:
Lines 40 and 41 in mnist_softmax.py are initializing the model: W and b with zeros. These are the values that the optimization algorithm (e.g. gradient descent) starts with and then takes small steps towards new values for W and b that can perform better in recognizing digits. Try initializing W and b with ones rather than zeros (i.e. replace tf.zeros by tf.ones). What is the test set accuracy in this case? Is it significantly different? Explain the reason behind your observation.

By replace tf.zeros to tf.ones, the test set accuracy for 1000 times looping and 10000 times looping both increased a little:

```
Accuracy on the testing set by 1000 times looping: 0.9185000061988831
Accuracy on the testing set by 1000 times looping: 0.919700026512146

Accuracy on the testing set by 10000 times looping(tf.ones): 0.9226999878883362
Accuracy on the testing set by 10000 times looping(tf.ones): 0.9226999878883362
```

However, when I tried to run with only 10 times looping, the accuracy increased a lot more:

```
Accuracy on the testing set by 10 times looping(tf.ones): 0.8062999844551086
Accuracy on the testing set by 10 times looping(tf.ones): 0.8140000104904175
```

The reason is that as the values in matrix W and array b is approaching to a ‘true value’ for W and b, which means the accuracy would be 100% if values in W and b are all ‘true values’, and most of these ‘true values’ are greater than 1. Each iteration would increase the values in certain amount, therefore, in the 10 times looping case, there is only 10 changing on values of W and b, therefore it’s harder to get to the ‘true values’. When we set the initial values to 1, the values in W and b would be closer to ‘true values’ initially, which makes up the short on 10 times looping. But the changing on initial value doesn’t work so well on 1000 times and 10000 times looping cases because they already have enough chances to modify the W and b values. Thus, increased the set accuracy of 10 times looping case much more obvious than 1000 and 10000 times looping cases.