COMS 573
Lab 2
SeokHwan Song

- There are 64 units in input layer and 10 in the output layer, and the output used softmax for the activation function. There are 765 rows of the validation set, and they are randomly picked. I used 'stochastic gradient descent' since I tried to change momentum to experiment and SGD shows better results with flatter minima than Adam[1]. The number of hidden layers did not affect that much to the accuracy. Moreover, too many hidden layers could cause overfitting. Therefore, only one hidden layer is used.
- Early stopping mode is 'auto'. Output layer used 'softmax' with 10 units. The number of epochs is 180 and the batch size is 15.

1. **Experiment with fully-connected feed-forward neural networks.**

    a. Sum-of-squares error vs. cross-entropy error function.
    i) Sum-of-squares error function

- Different learning rate

With 30 units for hidden layers and 0.2 Momentum rate, 0.3 learning rate performs well. Time decreases when the learning rate goes up and the accuracy was high at 0.3, so I selected 0.3

Learning rate	Time (Training)	Validation Accuracy
0.1	24.037	0.9556
0.15	15.566	0.9556
0.2	12.172	0.9595
0.3	9.7615	0.9608
0.4	6.9185	0.9556

- Different momentum rate

With 30 units for hidden layers and 0.3 learning rate, 0.3 Momentum rate performs well.

Momentum rate	Time (Training)	Validation Accuracy
0.1	9.9055	0.9608
0.15	9.299	0.9490

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

0.2	10.273	0.9556
0.3	8.252	0.9595
0.4	5.084	0.9503

- Different number of hidden units
  With 0.3 learning rate and 0.3 momentum rate, 30 units of hidden layer work well.

Number of hidden units	Time (Training)	Validation Accuracy
10	11.947	0.9608
15	9.094	0.9529
20	7.619	0.9451
30	8.338	0.9621
40	11.277	0.9529

Therefore, I used one hidden layer, 30 hidden units, 0.3 learning rate, 0.3 momentum rate, and these are the results with these hyper-parameters.

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

```
training set overall accuracy

0.9707036358880461


training set class accuracy

[array([0.99202128, 0.95372751, 0.98421053, 0.96915167, 0.9379845 ,
       0.98138298, 0.9867374 , 0.98966408, 0.96842105, 0.94502618])]


training set confusion matrix

[[373   0   0   0   1   0   1   0   1   0]
 [  0 371   3   0   1   1   0   3   6   4]
 [  0   0 374   0   0   0   1   2   2   1]
 [  0   0   1 377   0   6   0   0   1   4]
 [  1   2   0   0 363   0   4   0   6  11]
 [  0   1   2   1   0 369   0   0   0   3]
 [  1   3   0   0   1   0 372   0   0   0]
 [  0   0   1   1   1   0   0 383   0   1]
 [  0   4   0   2   3   2   1   0 368   0]
 [  1   4   1   3   4   1   0   3   4 361]]

testing set overall accuracy

0.9465776293823038


testing set class accuracy

[array([0.97191011, 0.93956044, 0.97175141, 0.91256831, 0.97790055,
       0.98351648, 0.96132597, 0.92178771, 0.8908046 , 0.93333333])]


testing set confusion matrix

[[173   0   0   0   1   4   0   0   0   0]
 [  0 171   2   0   0   1   0   0   1   7]
 [  0   3 172   0   0   0   0   2   0   0]
 [  0   0   6 167   0   4   0   2   2   2]
 [  0   0   0   0 177   0   0   0   3   1]
 [  0   1   0   0   0 179   0   0   0   2]
 [  1   4   0   0   1   0 174   0   1   0]
 [  0   0   0   0   1   6   0 165   2   5]
 [  0   9   0   0   0   5   0   0 155   5]
 [  0   1   1   1   5   2   0   0   2 168]]
```

In conclusion, the training set's accuracy is higher, but for class '4' the testing set's accuracy is higher than the training set's accuracy.

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

ii) Cross-entropy error function

I used the same parameters and the different error functions for this. Also, the experiment was similar to the previous one.

- Different learning rate
  With 30 units for hidden layers and 0.3 Momentum rate, 0.15 learning rate performs well.

Learning rate	Time (Training)	Validation Accuracy
0.1	2.945	0.9464
0.15	2.896	0.9595
0.2	2.284	0.9438
0.3	2.0167	0.9399

- Different momentum rate

With 30 units for hidden layers and 0.15 learning rate, 0.15 Momentum rate performs well.

Momentum rate	Time (Training)	Validation Accuracy
0.1	3.312	0.9608
0.15	3.275	0.9712
0.2	4.086	0.9634
0.3	3.981	0.9477

- Different number of hidden units
  With 0.15 learning rate and 0.15 momentum rate, 30 of hidden units work well.

Number of hidden units	Time (Training)	Validation Accuracy
10	3.594	0.9451
15	5.737	0.9556
20	1.841	0.9451
30	3.72	0.9608
40	2.848	0.9569

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

```
training set overall accuracy

0.970180486528904


training set class accuracy

[array([0.99468085, 0.98971722, 0.98684211, 0.96143959, 0.91731266,
       0.98138298, 0.95225464, 0.99224806, 0.96842105, 0.95811518])]


training set confusion matrix

[[374   0   0   0   1   0   0   0   1   0]
 [  0 385   0   0   0   0   0   1   1   2]
 [  1   1 375   0   0   0   0   0   2   1]
 [  0   0   2 374   0   4   0   1   4   4]
 [  1  12   0   0 355   0   1   0   3  15]
 [  0   1   2   1   0 369   0   0   0   3]
 [  2   9   0   0   2   1 359   0   4   0]
 [  0   0   1   1   0   0   0 384   0   1]
 [  1  10   0   0   0   1   0   0 368   0]
 [  0   6   0   0   2   1   0   2   5 366]]

testing set overall accuracy

0.9326655537006121


testing set class accuracy

[array([0.98314607, 0.98351648, 0.95480226, 0.89071038, 0.87845304,
       0.97802198, 0.92265193, 0.88826816, 0.8908046 , 0.95555556])]


testing set confusion matrix

[[175   0   0   0   0   3   0   0   0   0]
 [  0 179   0   0   0   0   0   0   0   3]
 [  0   7 169   0   0   0   0   0   1   0]
 [  0   0   5 163   0   3   0   2   6   4]
 [  0  18   0   0 159   0   0   0   2   2]
 [  0   1   0   0   0 178   0   0   0   3]
 [  2   8   0   0   1   0 167   0   3   0]
 [  0   1   0   0   2   6   0 159   2   9]
 [  0  10   0   0   0   3   0   0 155   6]
 [  0   1   0   1   0   2   0   0   4 172]]
```

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

In conclusion, the accuracies are very similar to MSE's accuracies. Some classes have bigger accuracy with cross-entropy function and some classes have bigger accuracy with MSE. However, the training time with the cross-entropy function is much shorter than the training time with MSE. Therefore, we can say cross-entropy is more appropriate than sum of squares with this dataset.

b. Tanh vs. Relu hidden units.
i) Tanh

- One layer and input scaling are used, and I changed the learning rate, momentum rate, size of the batch, and hidden units to find out which numbers would make it perform better. (Other parameters are the same as a))

- Different learning rate
  With 30 units for hidden layers and 0.3 Momentum rate, 0.2 learning rate performs well.

Learning rate	Time (Training)	Validation Accuracy
0.1	3.3610	0.9582
0.15	2.9161	0.9569
0.2	3.185	0.9608
0.3	3.403	0.9438

- Different momentum rate
  With 30 units for hidden layers and 0.2 learning rate, 0.2 Momentum rate performs well.

Momentum rate	Time (Training)	Validation Accuracy
0.1	2.780	0.9621
0.15	1.988	0.9333
0.2	3.393	0.9621
0.3	3.475	0.9516

- Different number of hidden units
  With 0.2 learning rate and 0.2 momentum rate, 20 of hidden units work well.

Number of hidden units	Time (Training)	Validation Accuracy

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

10	3.484	0.9621
15	2.861	0.9608
20	5.598	0.9699
30	2.887	0.9608
40	3.972	0.9595

```
training set overall accuracy

0.9814281977504578


training set class accuracy

[array([0.99468085, 0.98971722, 0.99473684, 0.97686375, 0.95607235,
       0.9787234 , 0.99204244, 0.98449612, 0.98157895, 0.96596859])]


training set confusion matrix

[[374   0   0   0   0   0   1   0   1   0]
 [  0 385   0   0   0   0   0   1   1   2]
 [  1   0 378   0   0   0   0   0   1   0]
 [  0   1   0 380   0   3   0   0   1   4]
 [  1   2   0   0 370   0   3   0   3   8]
 [  0   1   2   1   0 368   1   0   0   3]
 [  0   2   0   0   1   0 374   0   0   0]
 [  0   0   2   2   0   0   0 381   0   2]
 [  0   5   0   0   1   0   1   0 373   0]
 [  1   4   0   2   2   1   0   0   3 369]]
```

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

```
testing set overall accuracy

0.9532554257095158


testing set class accuracy

[array([0.98876404, 0.98901099, 0.96045198, 0.92896175, 0.96685083,
       0.96703297, 0.98342541, 0.87150838, 0.91954023, 0.95555556])]


testing set confusion matrix

[[176   0   1   0   0   1   0   0   0   0]
 [  0 180   0   0   0   0   0   0   0   2]
 [  0   6 170   0   0   0   1   0   0   0]
 [  0   0   5 170   0   3   0   0   3   2]
 [  0   2   0   0 175   0   1   0   3   0]
 [  0   1   1   0   0 176   1   0   0   3]
 [  2   1   0   0   0   0 178   0   0   0]
 [  0   0   0   0   1   7   0 156   4  11]
 [  0   8   0   0   0   2   0   0 160   4]
 [  0   3   0   1   0   2   0   0   2 172]]
```

ii) ReLU

- Used the result from part a). As a result, we can say that ‘tanh’ works well with the dataset, and the training time is also similar with the training time with cross-entropy function. Therefore, this model (‘tanh’) showed the best performance in these experiments with good scores of overall accuracy and class accuracies.

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

2. **Experiment with convolutional networks (CNNs).**
   I used only one hidden layer (ReLU), and the output layer used softmax. The input data reshaped to (8, 8, 1). I also followed the requirements of these questions like loss function and used the same optimizer. Other hyper-parameters are the same as previous experiments, and learning and momentum rates are both 0.2.

   I used convolution networks and changed some parameters in convolution layers.

- Different number of the dimensionality of the output space
  15 output space works well with small training time and high accuracy.

Dimensionality of output space	Time (Training)	Validation Accuracy
10	2.573	0.9569
15	2.936	0.9621
20	3.760	0.9595
30	1.687	0.9582

- Different kernel size
  The Validation accuracy goes up with bigger size of Kernel size and the difference of training time is not that big, so I selected 5,5 for the Kernel size.

Kernel size	Time (Training)	Validation Accuracy
2,2	2.5853	0.9582
3,3	3.625	0.9634
4,4	2.4510	0.9660
5,5	3.873	0.9699

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

```
training set overall accuracy

0.9816897724300288


training set class accuracy

[array([0.99468085, 0.98457584, 0.98947368, 0.96143959, 0.97674419,
       0.98138298, 0.99204244, 0.99741602, 0.98421053, 0.95549738])]


training set confusion matrix

[[374   0   0   0   1   0   0   0   1   0]
 [  0 383   0   0   0   0   0   1   3   2]
 [  0   1 376   0   0   0   0   1   1   1]
 [  0   0   1 374   0   4   0   1   5   4]
 [  0   0   0   0 378   0   2   0   0   7]
 [  0   1   2   1   0 369   0   0   0   3]
 [  0   2   0   0   1   0 374   0   0   0]
 [  0   0   0   1   0   0   0 386   0   0]
 [  0   3   0   0   1   0   1   0 374   1]
 [  0   5   2   2   2   1   0   1   4 365]]

testing set overall accuracy

0.9571508069003896


testing set class accuracy

[array([0.99438202, 0.96703297, 0.97740113, 0.92349727, 0.95027624,
       0.97802198, 0.97790055, 0.92178771, 0.93103448, 0.95      ])]


testing set confusion matrix

[[177   0   0   0   1   0   0   0   0   0]
 [  0 176   0   0   0   0   1   0   2   3]
 [  0   2 173   0   0   0   0   1   1   0]
 [  0   0   0 169   0   3   0   3   7   1]
 [  0   5   0   0 172   0   0   1   3   0]
 [  0   0   0   0   0 178   2   0   0   2]
 [  0   1   0   0   2   0 177   0   1   0]
 [  0   0   0   0   0   4   0 165   2   8]
 [  0   7   0   0   0   2   0   1 162   2]
 [  0   0   0   0   4   2   0   0   3 171]]
```

As a result, the model with convolution networks performs the best with shorter training time and high accuracy. Especially, all the accuracies for all classes are bigger than 0.9 (90%). Therefore, we can say that CNN is the most efficient and appropriate with this dataset shorter time and better prediction comparing to any other models above.

1. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/