COMP4107 - Assignment 2

Student Name: Yunkai Wang
Student Number: 100968473

Student Name: Jules Kuehn
Student Number: 100661464

Fall 2018

1. a) The graphs and tables below can be reproduced by running the relevant q1*.py file.

The 10x10 grid described in the assignment was created and the order randomized. The actual values z = f(x,y) were computed and stored in a matrix representing [x, y, z] for all 100 training items. After training on the 10x10 grid for 50,000 epochs, error predicting z values from the test data (the 9x9 grid) was very low for both 8 neurons and 50 neurons. Note that using 2 neurons shows to be insufficient to approximate the function in both dimensions.
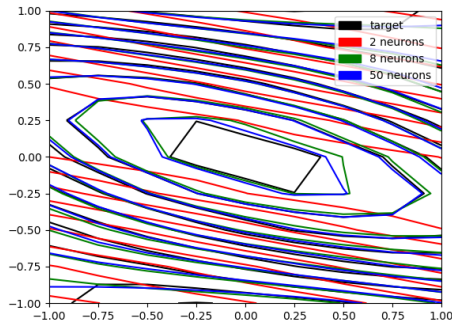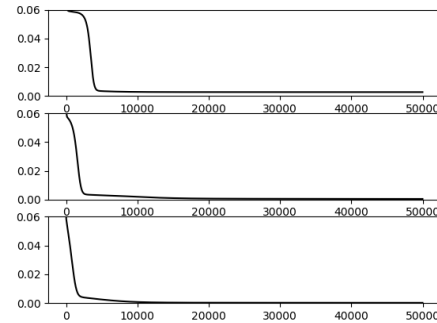

Figure 1: Function Contours

Figure 2: Cost (MSE) at epoch

Using 8 neurons for the hidden layer is appropriate for this function. The MSE for the test data with 8 neurons is better that is required, and the training converges after fewer epochs than the network using 50 neurons for the hidden layer.

The table below corresponds to the experiment shown in the graph above.

```
MSE with LR = 0.1 and 50000 epochs:
  2 neurons:  0.0028752042643646267
  8 neurons:  0.0004576152783257549
```

```
50 neurons:  0.0002225187166484513
```

b) With a tolerance of 0.02 for MSE of the training data, the appropriate number of epochs for convergence varies by training method. For traingrms (RMSPropOptimizer), having a maximum of 100 training epochs is appropriate. The other methods converge much more slowly, as seen in Figure 3.
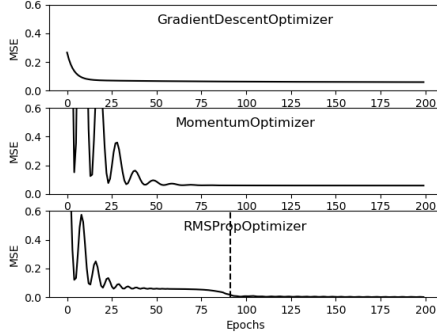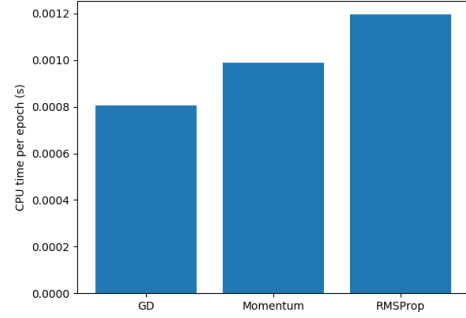

Figure 3: Convergence vs optimizer

Figure 4: CPU time vs optimizer

The reduced number of required training epochs for RMSPropOptimizer (seen in the table below) more than compensates for the increased CPU time for each epoch, seen in Figure 4.

```
Test MSE with LR = 0.02 and 100 epochs:
0.061681683118449925 (GradientDescentOptimizer)
0.058683357733909404 (MomentumOptimizer)
0.005489751198996285 (RMSPropOptimizer) <- BEST

Test MSE with LR = 0.02 and 200 epochs (as shown in Figure 3):
0.060366321060214334 (GradientDescentOptimizer)
0.059178147674373405 (MomentumOptimizer)
0.000918757108116926 (RMSPropOptimizer) <- BEST

Test MSE with LR = 0.02 (Stopping when the training error < 0.02):
0.020440891382299493 (GradientDescentOptimizer – Goal reached after 9623 epochs)
0.020200175134976295 (MomentumOptimizer – Goal reached after 918 epochs)
0.011620827855204354 (RMSPropOptimizer – Goal reached after 109 epochs) <- BEST
```

c) We can further show that 8 neurons is a good choice by noting that early stopping has no effect. Our experiments have not been able to demonstrate an overtraining phenomenon with even 50 hidden neurons, but a premature early stop can be seen in Figure 6, and the table below.

```
Test MSE with LR = 0.02, 50 neurons:
0.001858 at 10000 epochs
0.019969 at goal reached (2031 epochs)
0.075565 at early stop (399 epochs)
```
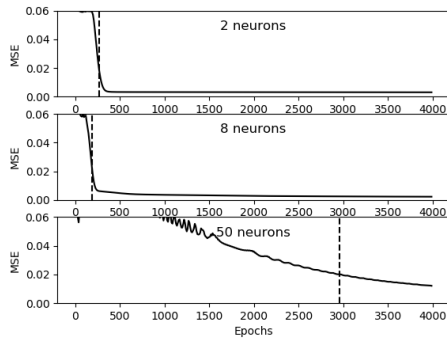
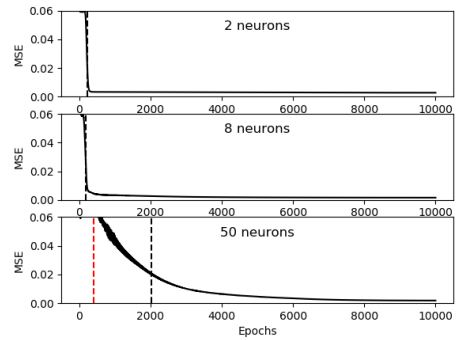Figure 5: Slow convergence

Figure 6: Premature early stop

The network with 50 hidden neurons also is very slow to converge to the target MSE - even using the RMSPropOptimizer - compared to when using 8 hidden neurons (Figure 5).
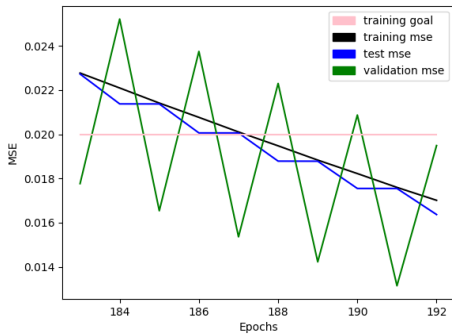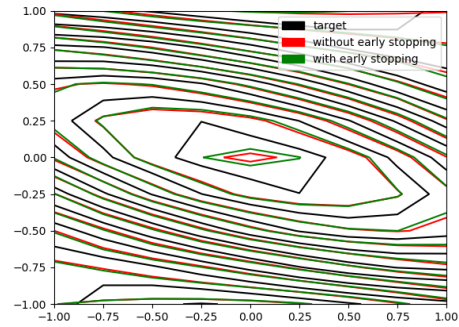

Figure 7: Reaching training goal

Figure 8: Early stopping not triggered

Figure 7 shows the MSE at each epoch near where the training goal is reached. This captures the expected trend - that the training error will decrease consistently, while the test error (9x9 grid) has more variation. The validation data is most different from the training data (as it is randomly sampled from the function), so it is expected that its error would also have the highest variance.

Figure 8 shows that having an early stopping mechanism for this function makes no difference, as the early stopping mechanism is never triggered (8 hidden neurons, 4000 epochs, 0.02 LR). Thus, the two contours are virtually identical, differing only by the stochastic elements of the experiments (initial weights, order of training data).

2. Instructions on how to run q2.py:

Running q2.py without changing anything will run all three parts. This may take a while as it runs 10 experiments for each number of hidden neurons, and after one is finished, a new

3

NN is created to run the second one, etc. Therefore, it will take about 1 min for it to finish and plot the graphs. If you want to run parts only, then comment out the corresponding line in the experiment function.

We started by using the default learning rate for AdamOptimizer (0.001). This took 400-700 epochs to converge to the result. However, if we use 0.01 as the learning rate, it converges faster – within 100 epochs every time. Therefore, we chose 0.01 as the learning rate for this question.

(a) We took the averages of 10 experiments for each of 5, 10, 15, 20, and 25 hidden neurons. These are plotted with red (5), blue (10), green (15), black (20), and purple (25) lines.
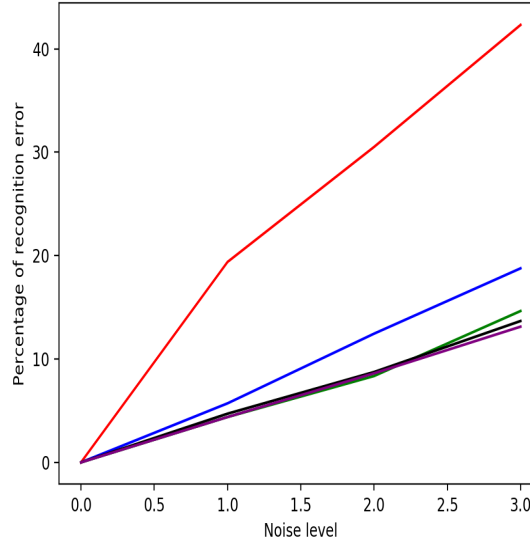

Figure 9: Percentage of Recognition Errors

```
Number of hidden neuron:  5 , error rate:  [0.0, 20.967741935483865, 32.54838709677418, 42.32258064516129]
Number of hidden neuron:  10 , error rate:  [0.0, 6.225806451612903, 11.419354838709705, 18.06451612903226]
Number of hidden neuron:  15 , error rate:  [0.0, 4.129032258064513, 7.41935483870968, 13.774193548387114]
Number of hidden neuron:  20 , error rate:  [0.0, 5.903225806451607, 10.41935483870967, 16.548387096774185]
Number of hidden neuron:  25 , error rate:  [0.0, 4.935483870967749, 9.129032258064507, 14.225806451612899]
```

Figure 10: Error rate for different number of hidden neurons

(b) 15 is chosen as the number of hidden neurons, since 15 is more accurate than 10, but using more than 15 neurons is less accurate on the noisy data. When we used 15 neurons in the hidden layer, the number of epochs to reach zero errors is much less than is shown in the assignment question, which we think is fine. Perhaps this is due to a different learning rate.
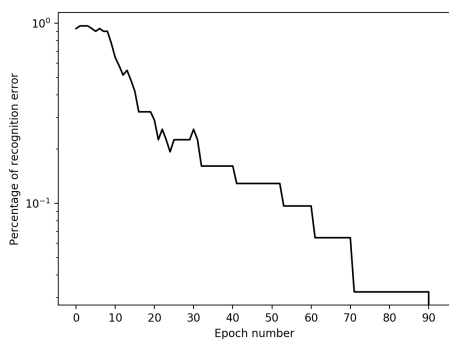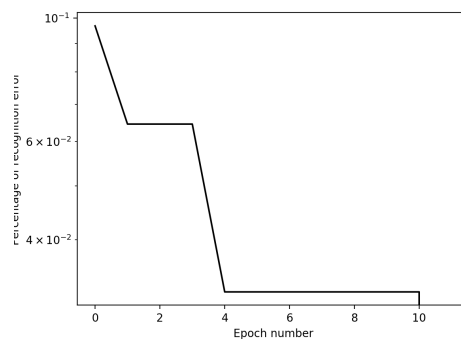
4

Figure 11: Training on perfect data

Figure 12: Re-training after noisy data

(c) The network trained with both perfect and noisy data performed just as well on perfect data, and better on noisy data. The chart below is for the network with 15 neurons. The blue dashed line is trained with only perfect data, and the red line is trained with both perfect and noisy data.
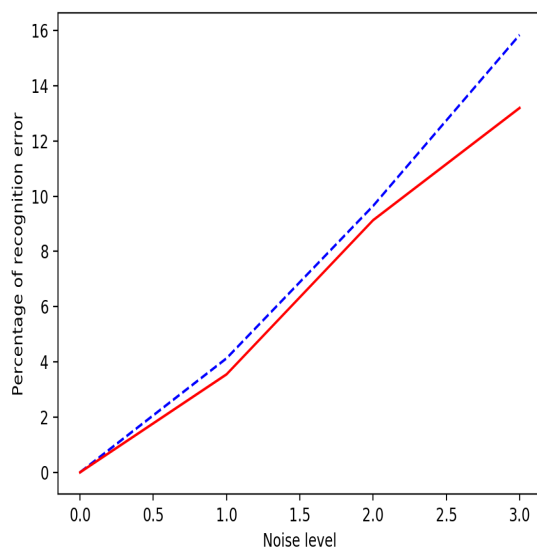

Figure 13: Percentage of Recognition Errors