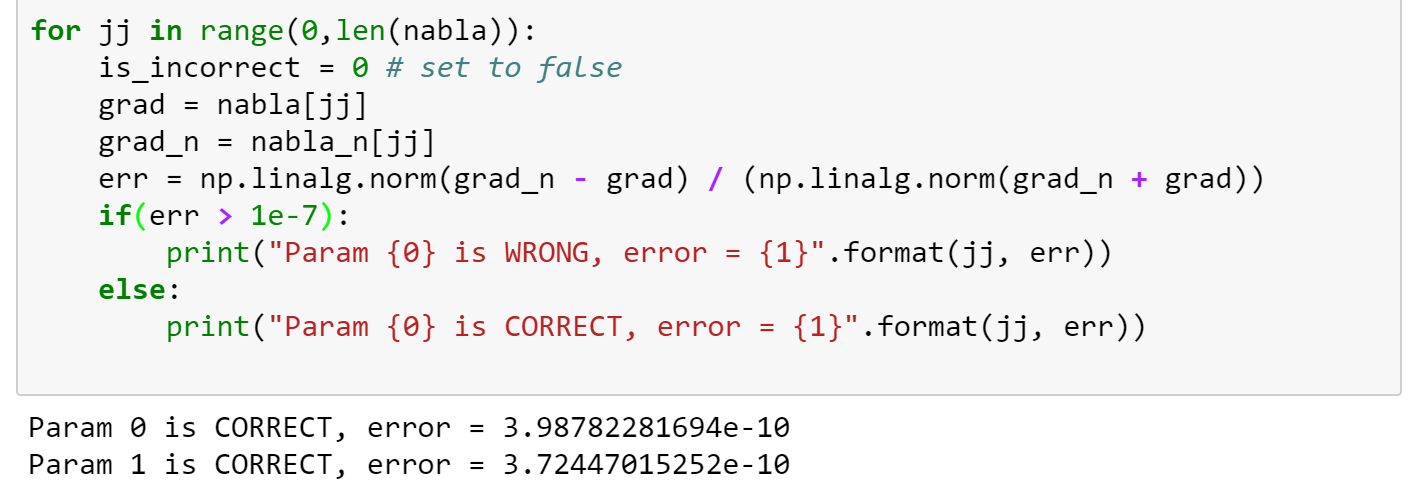
**HW2 of IST 597 – Deep Learning**

**Min-Chun Wu**

**Problem 1**

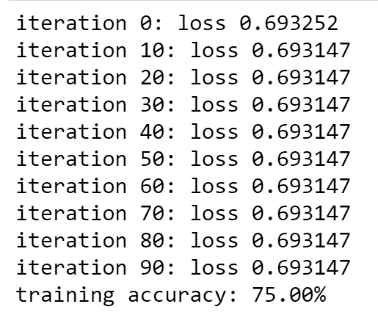
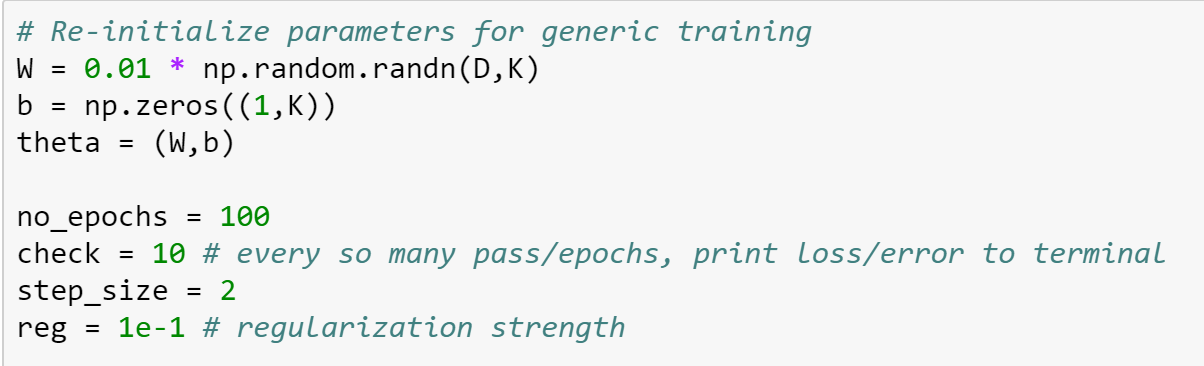
**(a)**

**1. Gradient-checking the code you have written for the partial derivatives is absolutely essential.**



Look at the figure above. The threshold 1e-7 is set to check the validity of the computed gradient. The errors are as shown, which are smaller than this threshold and pass the test.

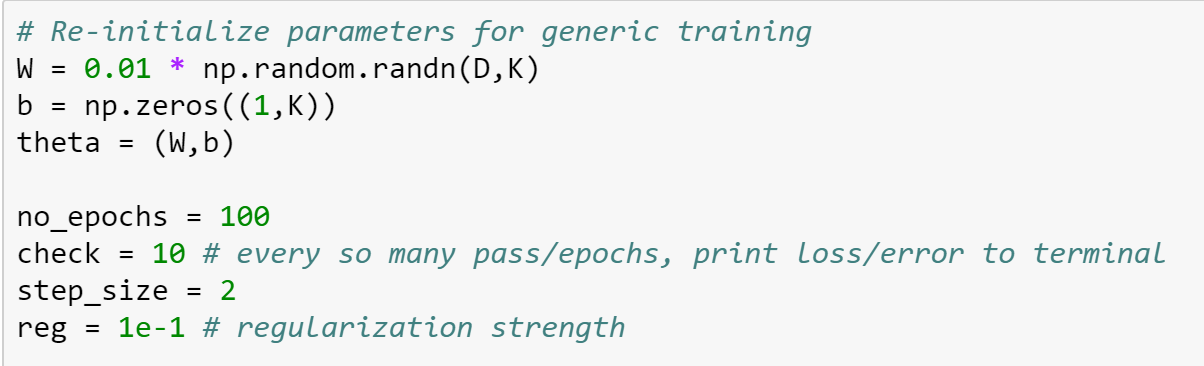
**2. Fit your multinoulli regressor to the XOR dataset. Record what tried for your training process and any observations (such as any of the model’s ability to fit the data) as well as your accuracy.**

Look at these two figures. The parameters are set as in the first figure and the training accuracies are as shown in the iteration process. The final training accuracy is 75 %.

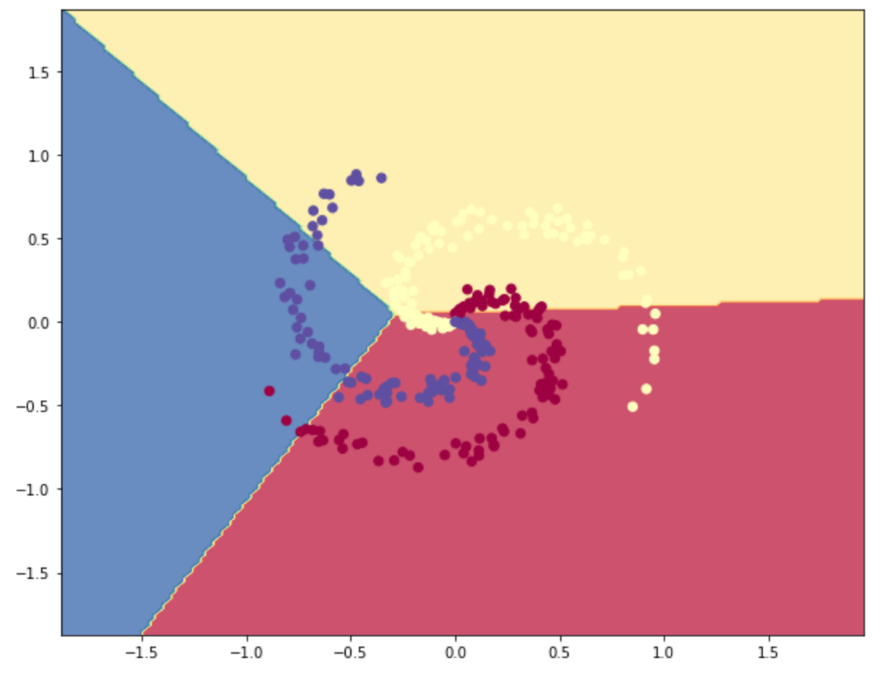
**(b)**

**1. Save and update your answer document with the generated plot. Comment on your observations and describe any insights/steps taken in tuning the model.**

Due to the restriction of the model, the data cannot be fitted perfectly. When tuning the parameters, the “center” of the model shifts according to our choice of step-size. It seems we are looking for the best center possible to fit the data. The following is the best possible parameters:

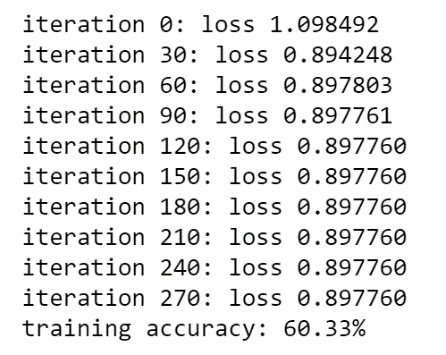


This is the plot.



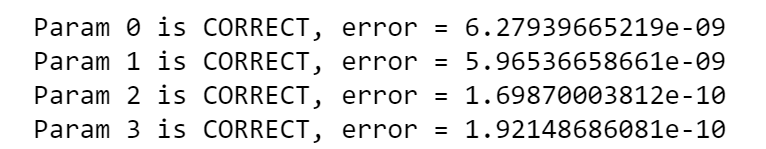
**2. Report your accuracy**

See the figure below.

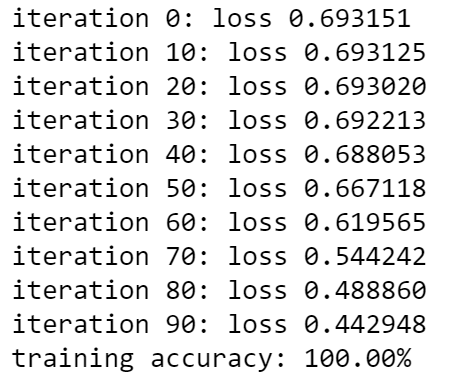


**(c)**

**1. Make sure your gradients pass the finite difference check.**



**2. Fit your MLP to the XOR problem data. Report your accuracy as well as record your loss as a function of epochs.**



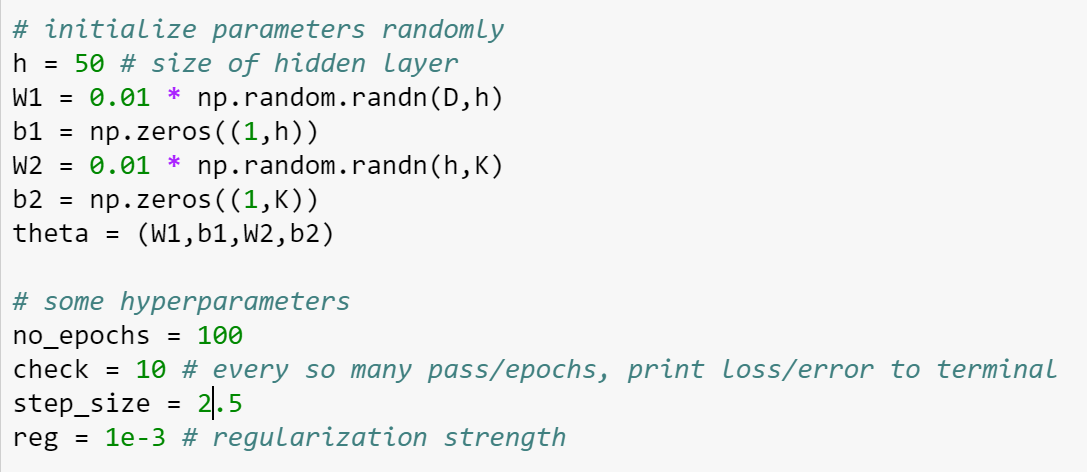
**3. Was your MLP model better able to fit the XOR data? Why would this be the case, when compared to your softmax regressor?**

Yes, it was much better, as shown in the figure above (100% accuracy). This is mainly because the more layers the model has, the more capacity it has. Therefore, this is not surprising.

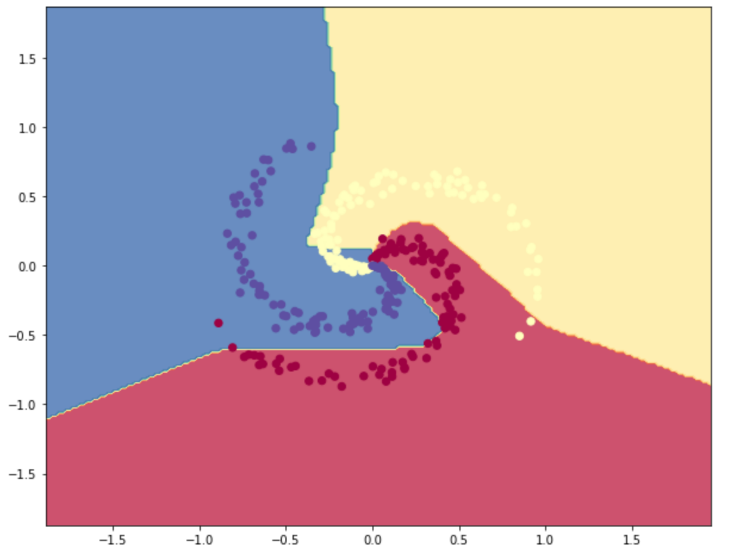
**(d)**

**1. Document what you did to tune the MLP, including what settings of the hyper-parameters you explored. Write your observations and thoughts/insights**

It’s too bothering to write down all hyper-parameters I explored, so I only write down the final best hyper-parameters I found:



**2. Generate the decision boundary plot of your tuned MLP and paste it into your answer document.**



**3. What is different between the MLP’s decision boundary and the multinoulli regressor’s decision boundary?**

Those for MLP can be curved while those for multinoulli regressors’ cannot.

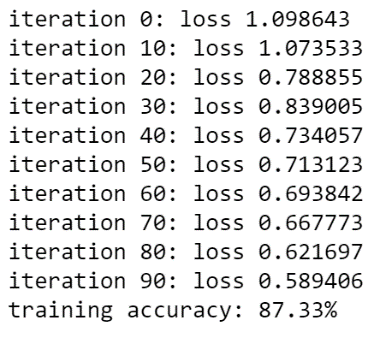
**4. Does the MLP decision boundary accurately capture the data distribution?**

Yes, as one can see from the figure shown above. It fits quite well.

**5. How did the regularization coefficient affect your model’s decision boundary?**

When your regularization coefficient becomes larger, the decision boundary has less freedom and becomes more like a straight line.

**6. Report your accuracy.**

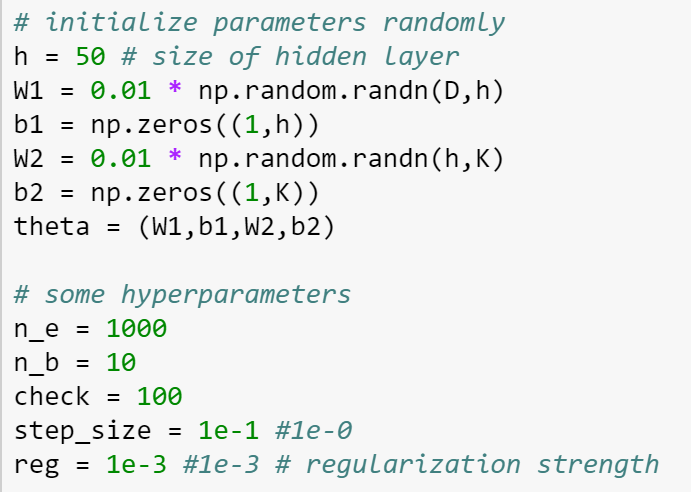
****

**Problem 2**

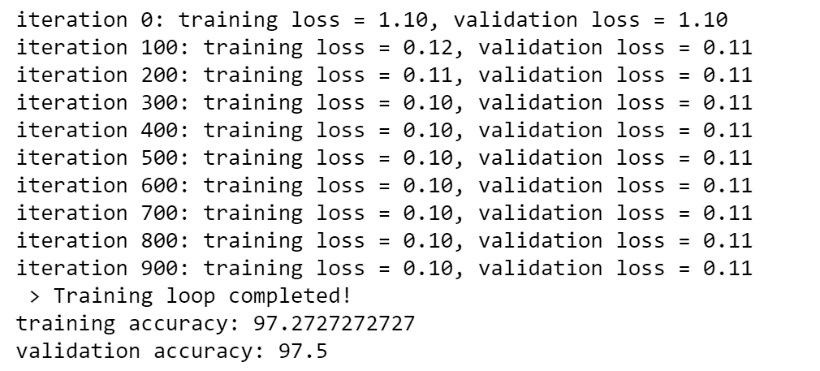
**(a)**

**1. Record your accuracy for both your training and development sets and track your loss as a function of epoch. Create a plot with both of these curves superimposed.**

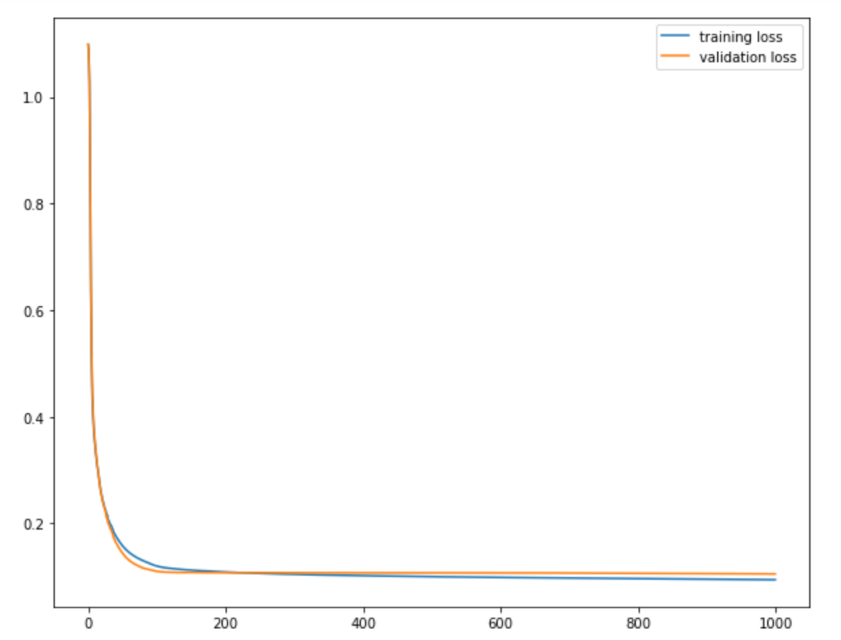
The following hyper-parameters are eventually tuned:



The following is the error tracking:



The following is the error plot (both training and validation).



**2. What ultimately happens as you train your MLP for more epochs?**

Eventually, we can see from the plot that the training error starts to be lower than the validation error.

**3. What phenomenon are you observing and why does this happen?**

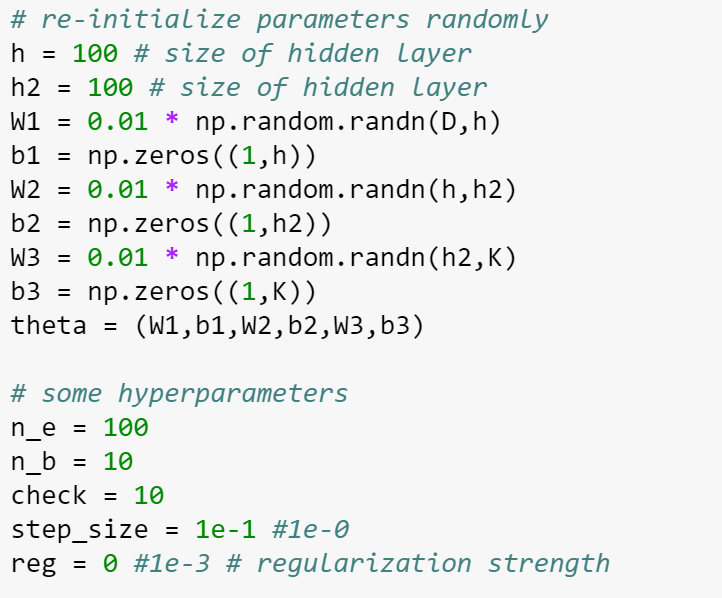
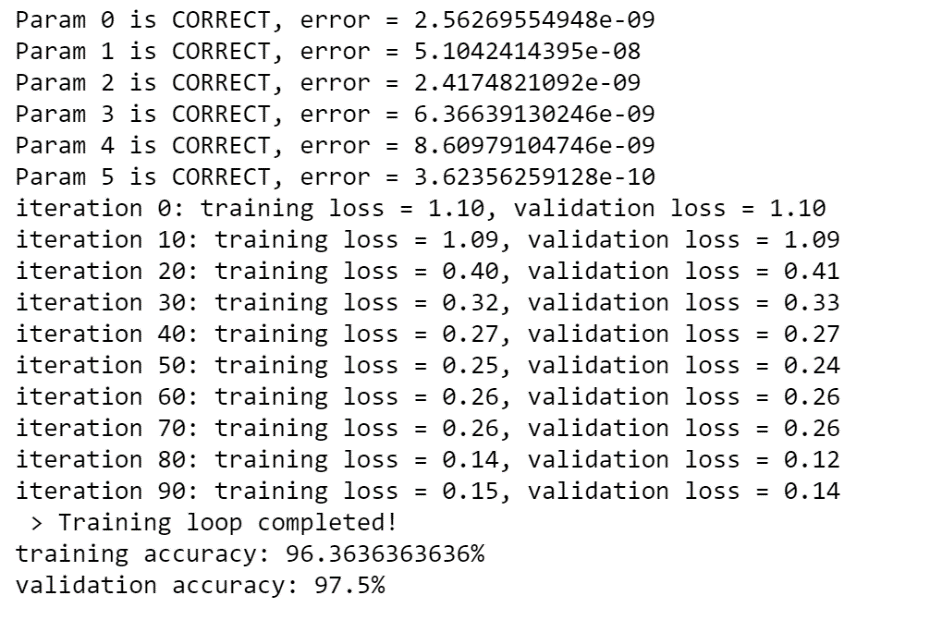
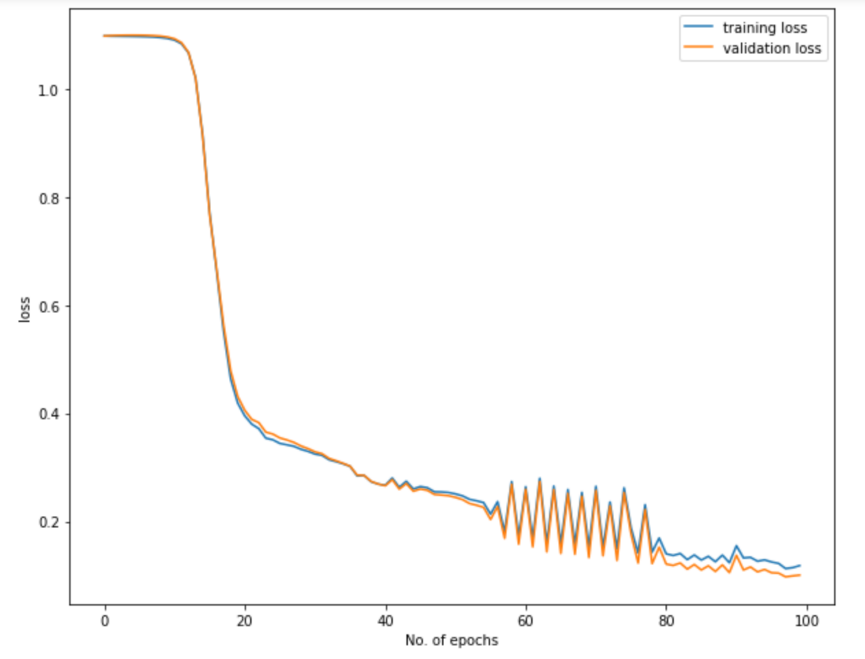
Overfitting phenomenon. This is due to the model capacity is high enough to allow overfitting.

**4. What are two ways you can control for this?**

One simple way is to “early-stop” the training process. We can follow the plot above to determine when to stop. The other way is to reduce the size of hidden layer or regularization coefficient so that the model capacity is reduced to avoid overfitting.

**(b)**

**1. Fit/tune your deeper MLP to the IRIS dataset and record your training/validation accuracies. Create plots of your loss-versus-epoch curves as you did in Problem #1a.**

The following parameters are tuned and used:  
  
The following is the error record in the training process:  
  
Look at the following for the plot of loss-versus-epoch curves:  


**2. Write down any thoughts/comments of your tuning process and the differences observed between training a single and two-hidden layer MLP.**

It’s easy to see that the error curve behaves more stably for the 1-hidden layer case comparing to the 2-hidden-layer case. The reason could be that the loss function of the 2-hidden-layer case is much more complicated (further from being convex) than the 1-hidden-layer case. Therefore, there could be more fluctuations in the training process.