

IST 597: Foundation of Deep Learning

HW 2: Back-Propagation of Errors and Multilayer Perceptrons

Nikhil Sunil Nandoskar, nxn59@psu.edu

Problem 1a) Softmax Regression and the XOR Problem

Q 1) Make sure the program returns “CORRECT” for each parameter matrix/vector before focusing on tuning your model on the XOR data.

Ans: Yes the program returns “CORRECT” for each parameter matrix/vector The snapshot of the terminal output is attached below.

```
Param 0 is CORRECT, error = 2.351276687932414e-08
Param 1 is CORRECT, error = 1.9663919034049805e-08
iteration 0: loss 0.693149
iteration 10: loss 0.693148
iteration 20: loss 0.693148
iteration 30: loss 0.693148
iteration 40: loss 0.693148
iteration 50: loss 0.693148
iteration 60: loss 0.693147
iteration 70: loss 0.693147
iteration 80: loss 0.693147
iteration 90: loss 0.693147
training accuracy: 0.75
```

Figure 1: Displaying the result of ComputeNumgrad

Q2) 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.

Ans: The first thing to understand here is XOR dataset is non-linear and we are building a linear model using "Softmax" function. The model will misclassify one point of the dataset for the best hyperparameters given to it. The best-expected accuracy, in this case, we can predict is "75%". I have tried various combinations of step-size (learning-rate) and regularisation to get the best accuracy.

Step-Size	Regularisation	Accuracy (%)
0.01	0.01	50
0.01	0.1	50
0.1	0.1	50
1	0.001	75
1	0.01	75
1	1	75

Table 1: The number of epochs kept here is 100 for all the cases above

Best Parameters Chosen:

Accuracy: 75%

Step-size: 1

Regularisation: 0.001

Problem 1b) Softmax Regression and the Spiral Problem

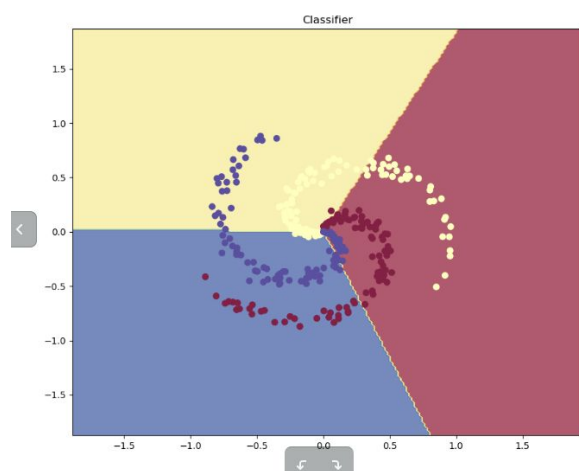
Q 1) Make sure you comment on your observations and describe any insights/steps taken in tuning the model. Do NOT forget to report your accuracy.

Ans: Following table shows the hyperparameters setting and its corresponding accuracy

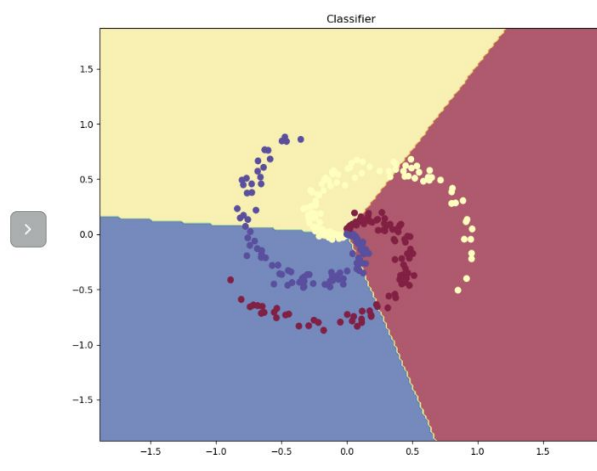
Number	Step-Size	Regularisation	Accuracy (%)
1	0.0002	0.01	54
2	0.001	0.001	53
3	0.1	0.1	49
4	1	1	49
5	10	10	33
6	0.001	10	51

Table 2: The number of epochs kept here is 500 for all the cases above

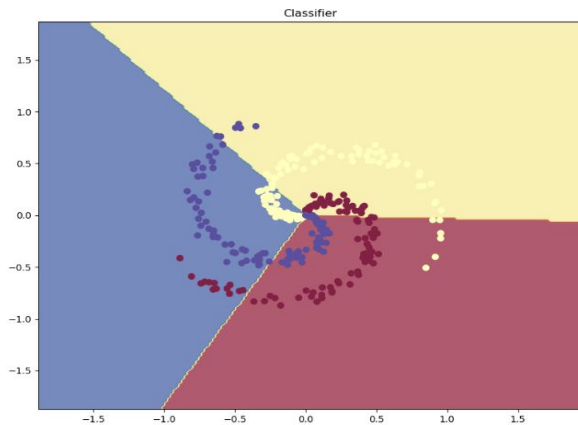
For the readings given in Table 2 above the corresponding plots obtained are as below:



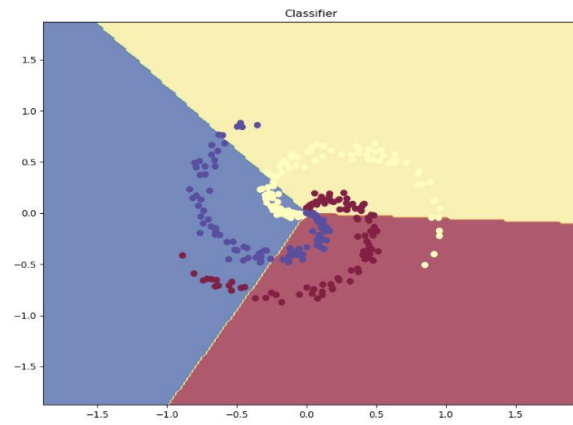
(1)



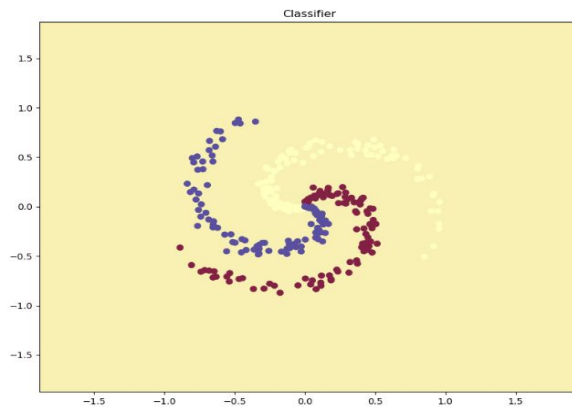
(2)



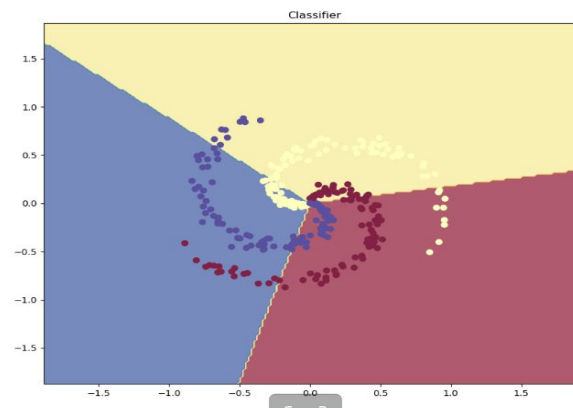
(3)



(4)



(5)



(6)

Figure 2: Plots of all the hyperparameter settings for 1b

After carefully observing Table 2 and Figure 2, the best accuracy I'm choosing is:

Accuracy: 49%

Step-size: 0.1

Regularisation: 0.1

The reason behind choosing accuracy to be 49% over 54% is that the decision boundaries for accuracy 49% are better than for 54%. It aligns pretty well with all the three classes.

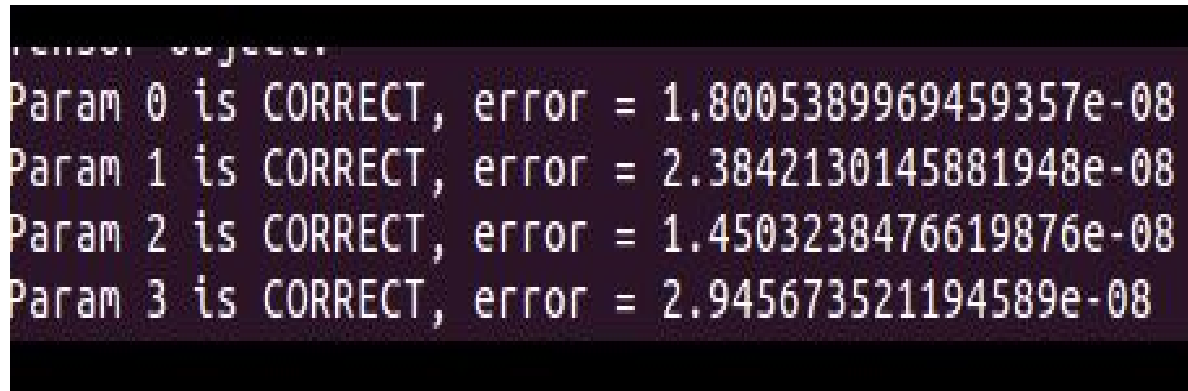
The observations of the hyperparameter tuning process are as follows:

- 1) As we increase regularisation and step-size accuracy decreases. It is evident from Figure 2 (5), it classifies only one class currently. Thus, accuracy is 33%.
- 2) The spiral dataset is non-linear and we trying to fit a linear model to it. Thus, the accuracy is low in this case. We require at least one hidden layer with non-linear activation function to fit the model as per the data.

Problem 1c) MLP and the XOR Problem

Q1) Report your accuracy as well as record your loss as a function of epochs (present its evolution during training either through a plot or a screenshot of the program output that should appear in the terminal)

Ans: The first step was to check the compute numerical gradient same like part 1a. Following snippet shows the terminal output:



```
Param 0 is CORRECT, error = 1.8005389969459357e-08  
Param 1 is CORRECT, error = 2.3842130145881948e-08  
Param 2 is CORRECT, error = 1.4503238476619876e-08  
Param 3 is CORRECT, error = 2.945673521194589e-08
```

Figure 3: Displaying the result of ComputeNumgrad

After observing various hyperparameters the best result obtained is as follows:

Best Parameters Observed:

Accuracy: 100%

Step-size: 0.1

Regularisation: 0.01

Initial loss: 0.693150

Loss after first 10 epochs: 0.657458

Final loss: 0.118058

Size of Hidden Layer: 6

Number of Epochs: 500

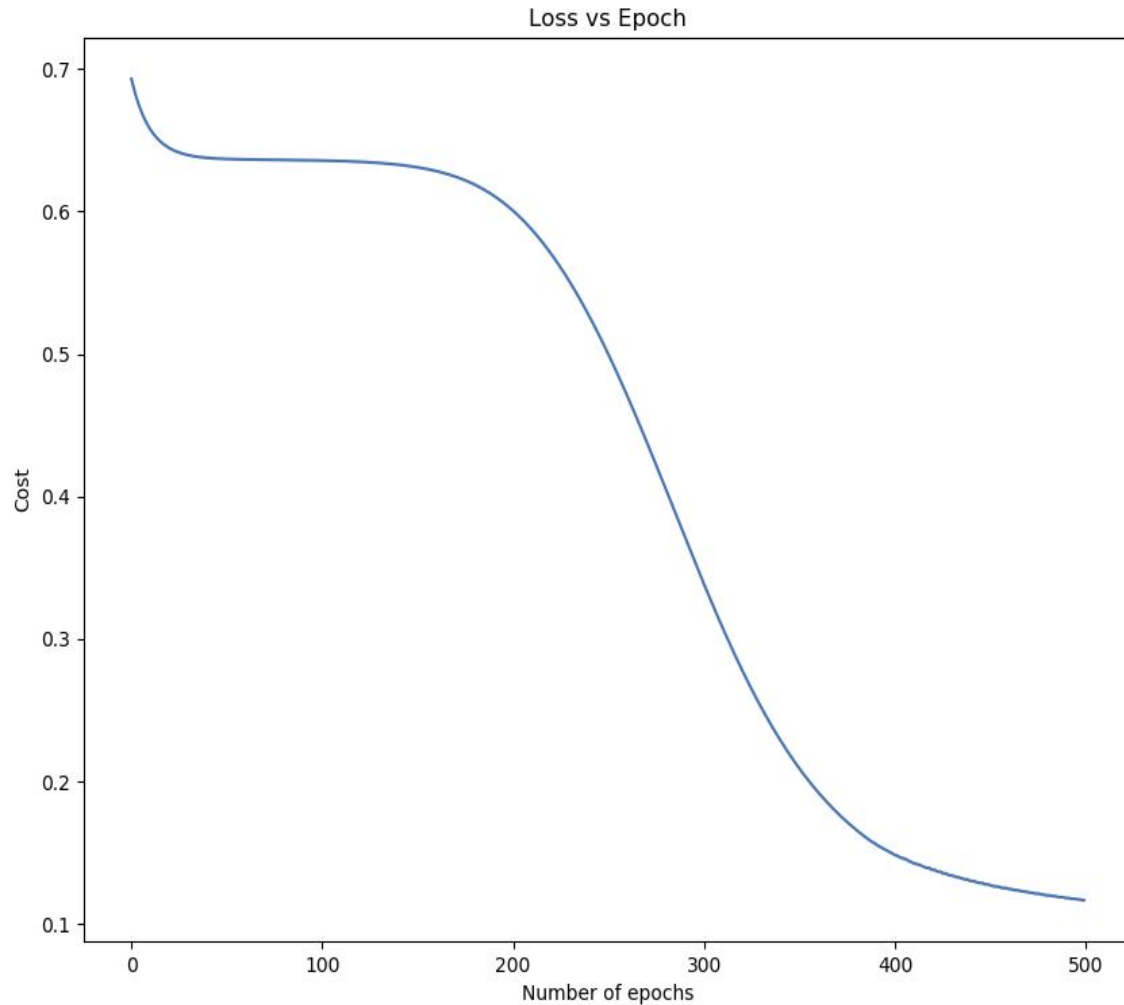


Figure 4: Loss as a function of epochs

Q2) Was your MLP model better able to fit the XOR data? Why would this be the case, when compared to your softmax regressor?

Ans: As it is clearly evident from the mentioned result above, the MLP model perfectly fits the XOR data. This is because of the hidden layer having non-linear activation function. Due to the nonlinearity added by the “RELU” activation function of the hidden layer the model is able to perfectly partition into the non-linear decision boundaries. Whereas, in case of “Softmax” activation function gives us a linear model it is able to separate the decision Boundaries in 2-Dimensional space.

Problem 1d) MLPs and the Spiral Problem

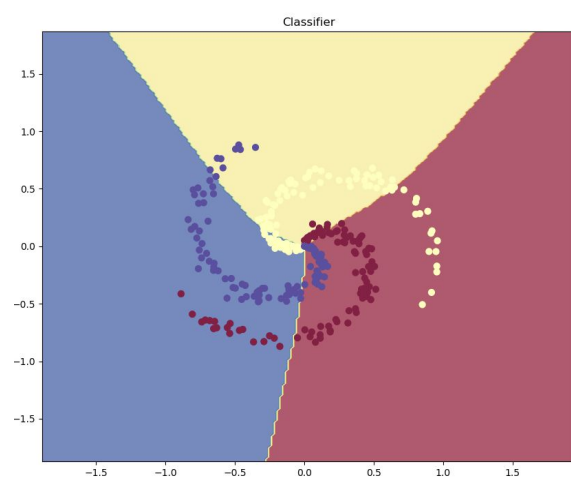
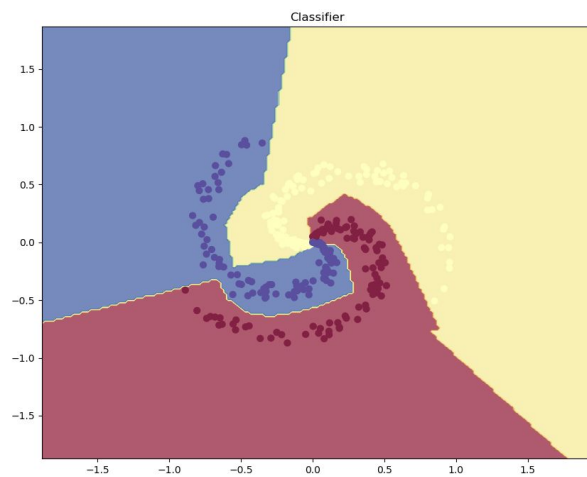
Q1) Make sure you document what you did to tune the MLP, including what settings of the hyper-parameters you explored (such as learning rate/step-size, number of hidden units, number of epochs, value of regularization coefficient, etc.). Do not forget to write your observations and thoughts/insights.

Ans: Following table shows the hyperparameters setting and its corresponding accuracy

Number	Step-Size	Regularisation	Accuracy (%)
1	1	0.0001	99
2	0.001	0.001	58
3	1	0.001	97
4	0.001	0.001	52
5	1	0.01	72
6	1	0.1	51

Table 3: The number of epochs kept here is 2000 for all the cases above

For the readings given in Table 3 above the corresponding plots obtained are as below:



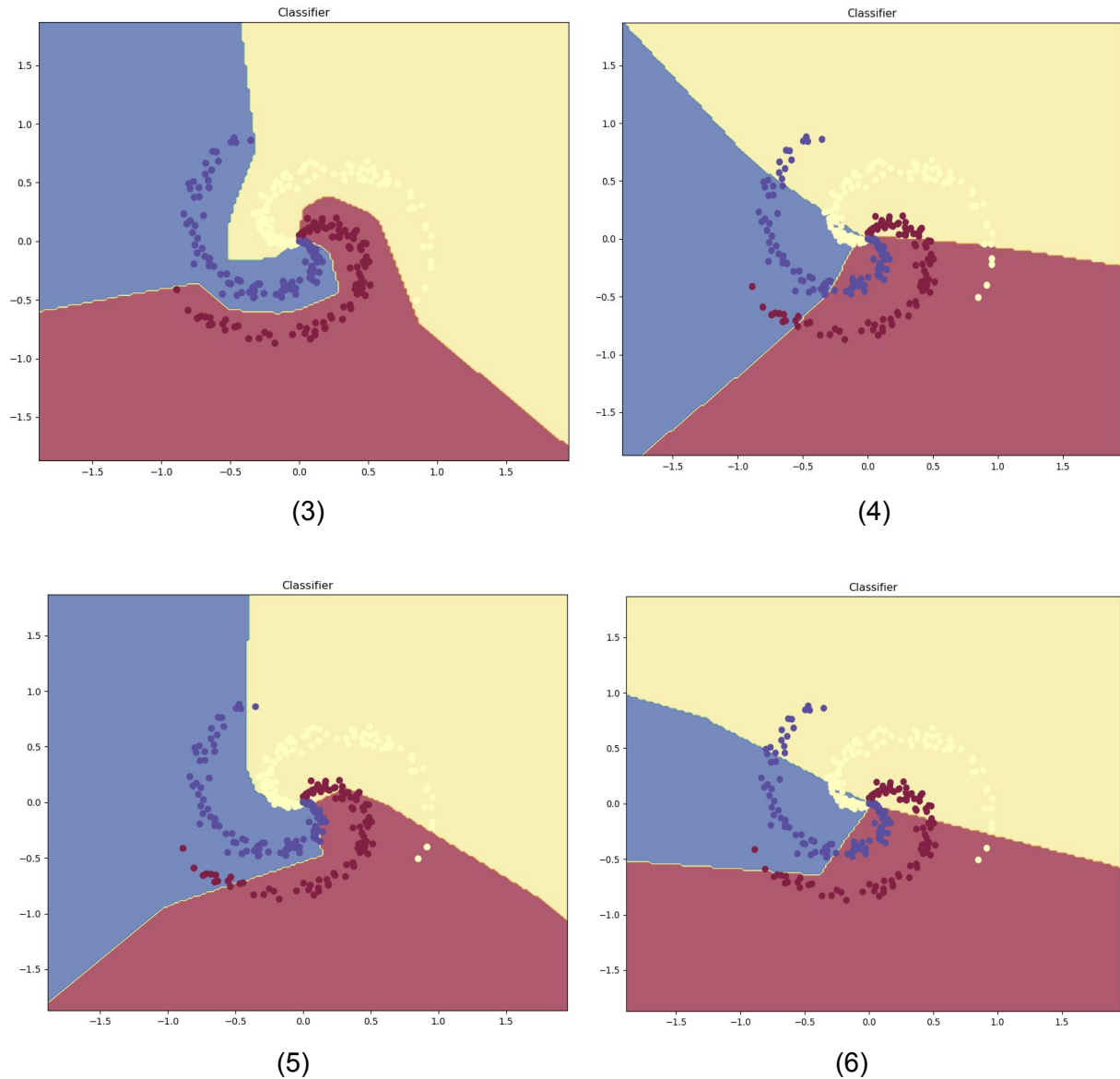


Figure 5: The above plots have number of epochs 2000 and size of layer to be 100

The observations of the hyperparameter tuning process are as follows:

- 1) As step-size and regularisation increases the accuracy is less. This is justified by the concept as step size increases it moves quickly towards local minima and the loss is also high due to higher regularisation.
- 2) In comparison with 1b, the MLP model tries to fit all the data points to its respective class (Figure 5 (1)). Just to check whether our model is overfitting we need to try it on a validation set.
- 3) Due to “RELU” activation function on the Hidden layer our MLP model tries to learn the data with more precision and the nonlinearity makes it to curve along with the spiral data.

Best Parameters Observed:

Accuracy: 99%

Step-size: 1

Regularisation: 0.0001

Size of Hidden Layer: 100

Number of Epochs: 2000

Q2) Generate the decision boundary plot of your tuned MLP and paste it into your answer document. Comment (in your answer document) on the decision boundary plot: What is different between the MLP's decision boundary and the multinoulli regressor's decision Boundary? Does the MLP decision boundary accurately capture the data distribution?

Ans: Following is the plot of the best tuned MLP and multinoulli regressor model for the spiral dataset.

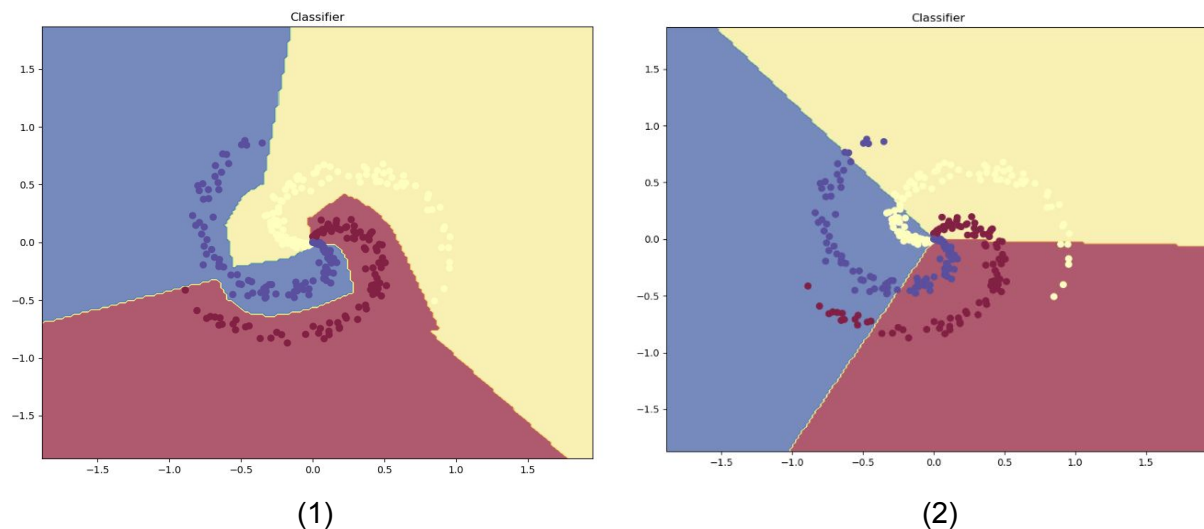


Figure 6: (1) is for tuned MLP model, (2) is for tuned multinoulli regressor model

As it is clearly evident from the figures above the decision boundary plot for the MLP model is better than multinoulli regressor model. Figure 6(2) shows the boundary plotted by a linear model thus we don't observe any curves but just slopes of straight lines. Whereas, in Figure 6(1)

we can see the decision boundary to curve along with the data points classifying them into respective classes. It will be very interesting to see when we add at least one data point for validation how the model behaves! Can it classify it correctly? Or is there any overfitting in the MLP model selected? If so we can go with a different model hyperparameter having high regularization, accuracy will be little less though but it will take care of high bias and high variance problem.

The MLP decision boundary accurately captures the data distribution.

Q3) How did the regularization coefficient affect your model's decision boundary? Do NOT forget to report your accuracy.

Ans: Following graphs are for decision boundary of various regularisation parameter and keeping step-size to be 1

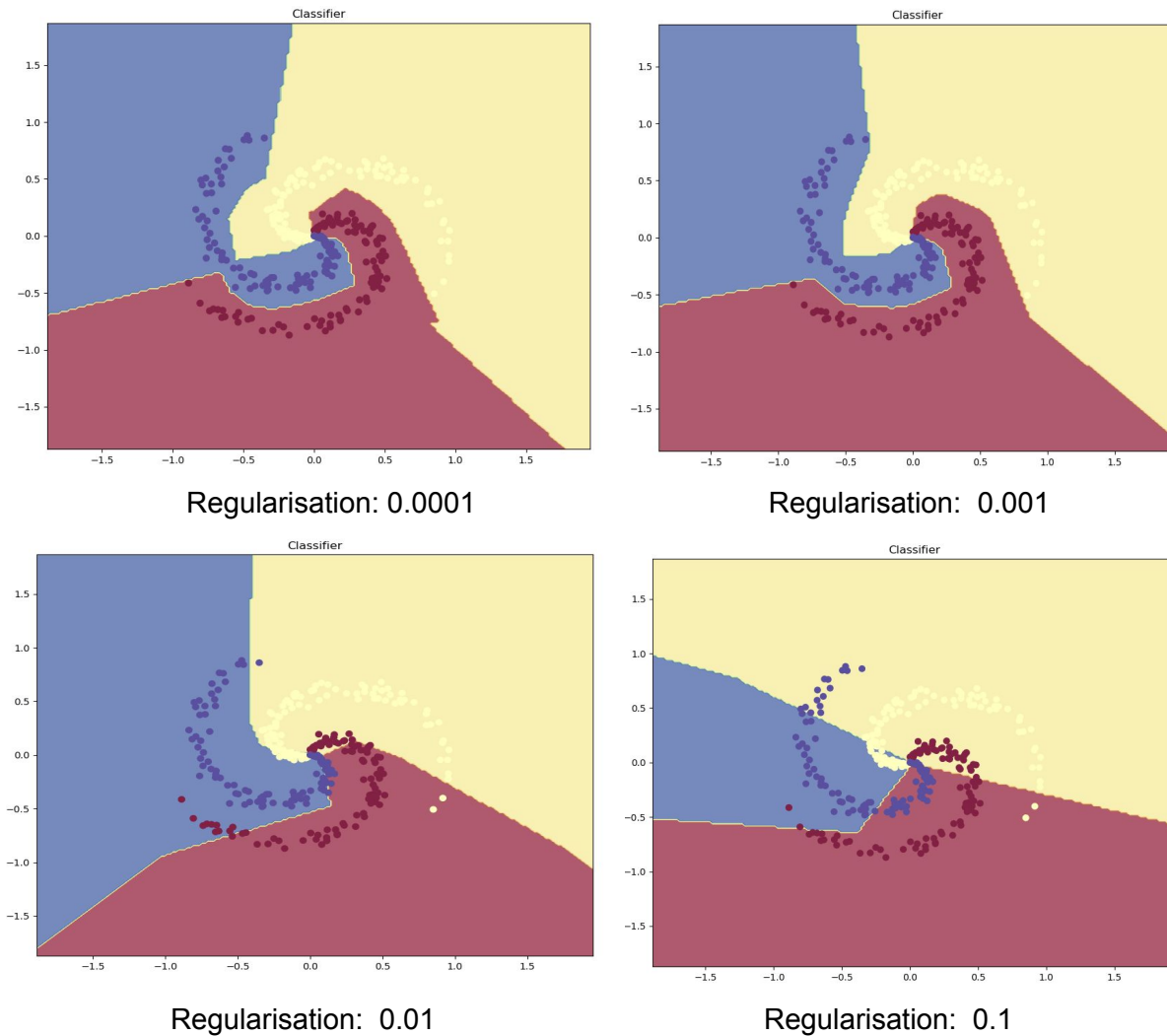


Figure 7: For various regularisation parameter keeping number of epochs 2000 and size of layer to be 100

It can be seen from above figures that as for lesser regularisation the model tries to learn and fit all the data points to its respective class. It follows all the data points classifying them. But as regularisation increases the model becomes less curvy and doesn't follow all the data points.

Best Parameters Observed:

Accuracy: 99%

Step-size: 1

Regularisation: 0.0001

Size of Hidden Layer: 100

Problem 2a) 1-Layer MLP for IRIS

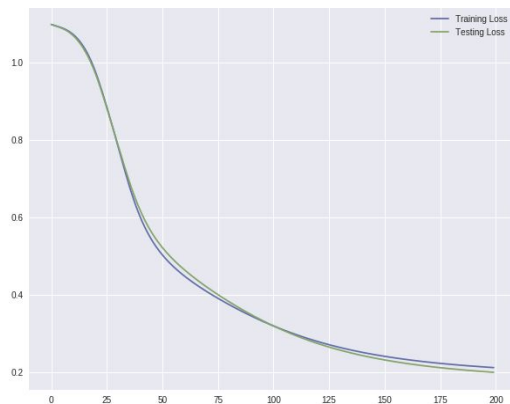
Q1) Besides recording your accuracy for both your training and development sets, track your loss as a function of epoch. Create a plot with both of these curves superimposed.

Ans: Following table shows the hyperparameters setting and its corresponding accuracies

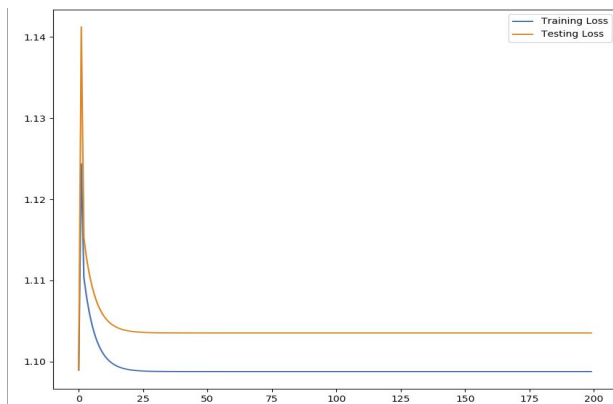
Number	Step-Size	Regularis ation	Train Accuracy (%)	Training Loss	Test Accuracy (%)	Test Loss
1	0.01	0.0001	97.27	0.21	97.5	0.20
2	1	0.001	35	1.09	30	1.10
3	0.01	0.01	97.27	0.07	97.5	0.06

Table 4: The number of epochs kept here is 200, except for the last case where number of epochs are 500

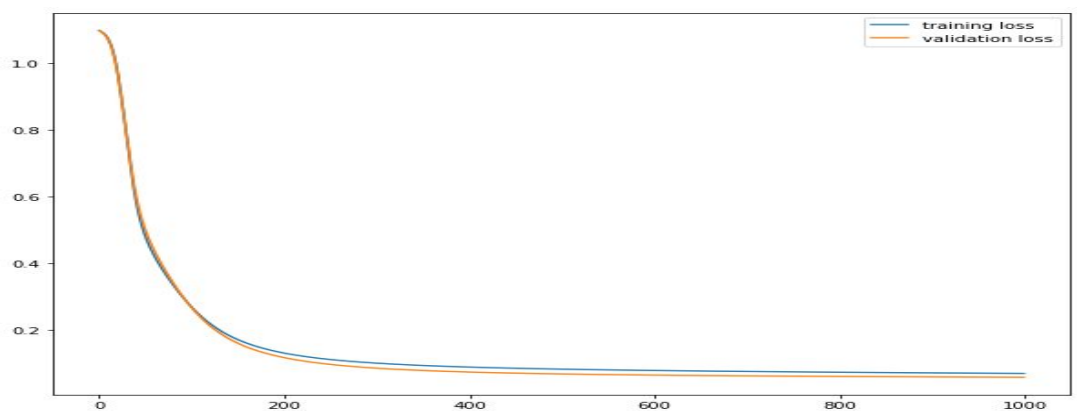
For the readings given in Table 4 above the corresponding plots obtained are as below:



(1)



(2)



(3)

Figure 8: Loss vs Number of epochs for various hyperparameters settings

Best Parameters:

Training Accuracy: 97.27%

Validation Accuracy: 97.5%

Step-size: 0.010

Regularisation: 0.0001

Size of Hidden Layer: 100

Number of Epochs: 200

Q2) What ultimately happens as you train your MLP for more epochs? What phenomenon are you observing and why does this happen? What are two ways you can control for this?

Ans: I have increased the number of epochs by the value 800. Following is the plot of Loss vs number of epochs.

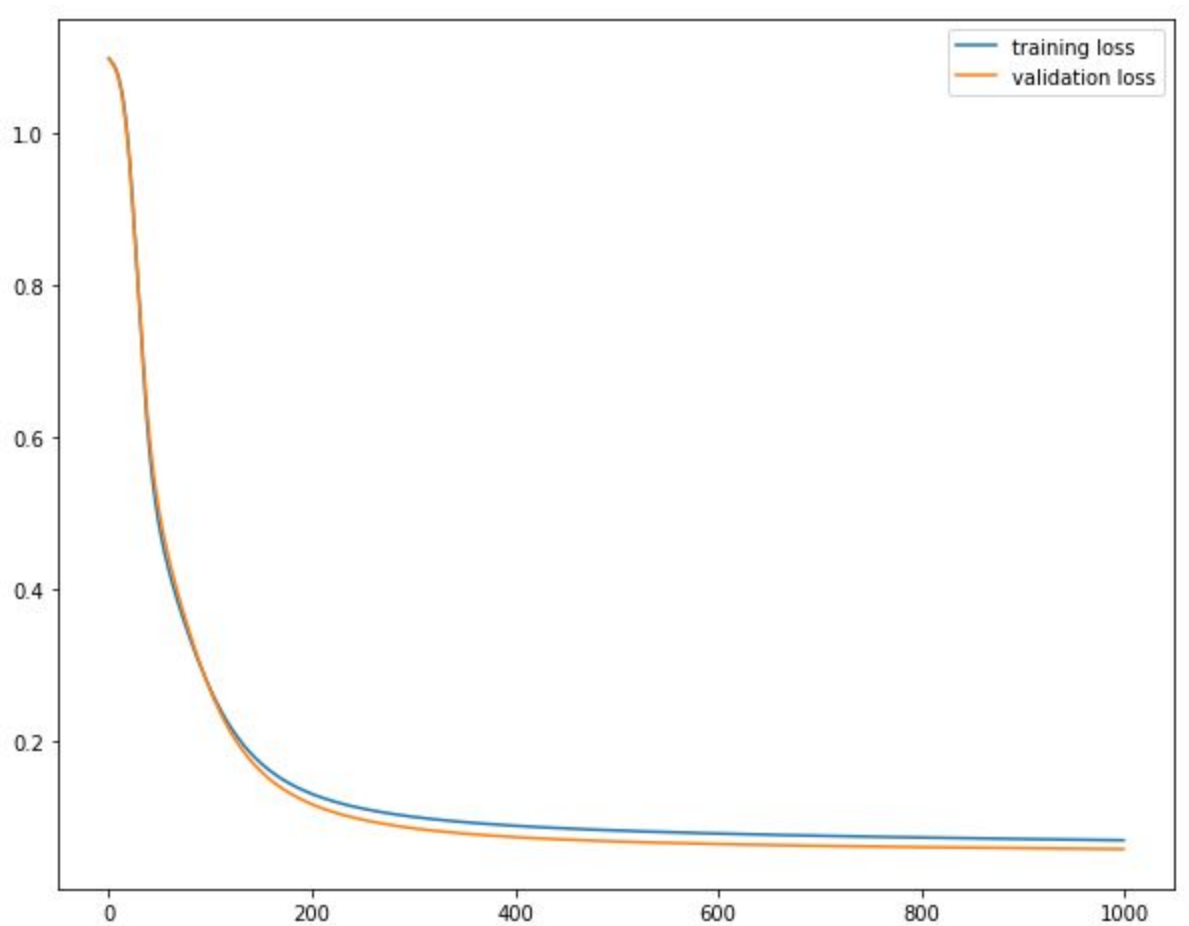


Figure 9: The number of epochs are 1000, step-size: 0.01, regularisation: 0.01

Observations:

As we increase the number of epochs we see a gap/difference between the Training and Validation loss. This is evident from Table 4, the difference is of 0.01 unit. This is so because as we train with a lot number of epochs the model tends to overfit the data, thus it performs little poorly on testing data. This can be solved by penalising our model with regularisation techniques. The choice of regularisation viz Elastic Net, L1, L2 is ours and we try all three methods and see which performs better. We can also make use of “Early stopping” method which we used in HW 1.

Problem 2b) 2-Layer MLP for IRIS

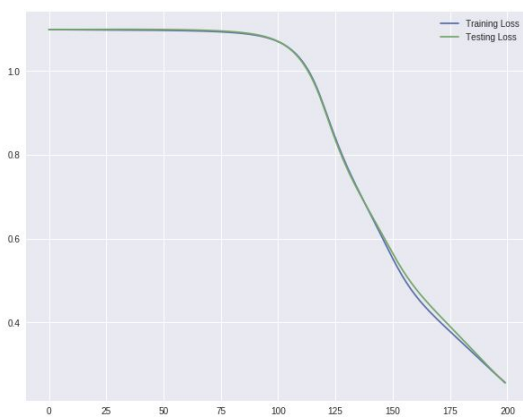
Q1) Fit/tune your deeper MLP to the IRIS dataset and record your training/validation accuracies.

Ans: Following table shows the hyperparameters setting and its corresponding accuracy

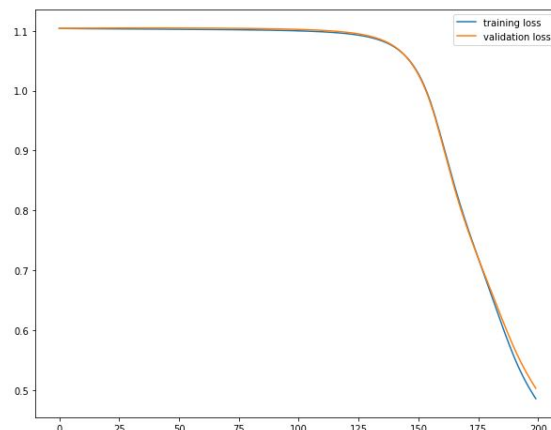
Number	Step-Size	Regularis ation	Train Accuracy (%)	Training Loss	Test Accuracy (%)	Test Loss
1	0.01	0.0001	97	0.30	98	0.30
2	0.01	0.01	80	0.55	82.5	0.57

Table 5: Number of epochs are 500 and size of both the hidden layers is 100

For the readings given in Table 5 above the corresponding plots obtained are as below:



(1)



(2)

Figure 10: Loss vs Number of epochs

Best Parameters:

Training Accuracy: 97%

Validation Accuracy: 98%

Step-size: 0.010

Regularisation: 0.0001

Size of Hidden Layers: 100

Number of Epochs: 200

Q2) Furthermore, create plots of your loss-versus-epoch curves as you did in Problem #1a. Put all of this in your answer document and write down any thoughts/comments of your tuning process and the differences observed between training a single and two-hidden layer MLP.

Ans: Following graphs are of 1 Layer MLP and 2 Layer MLP's Loss vs Epochs

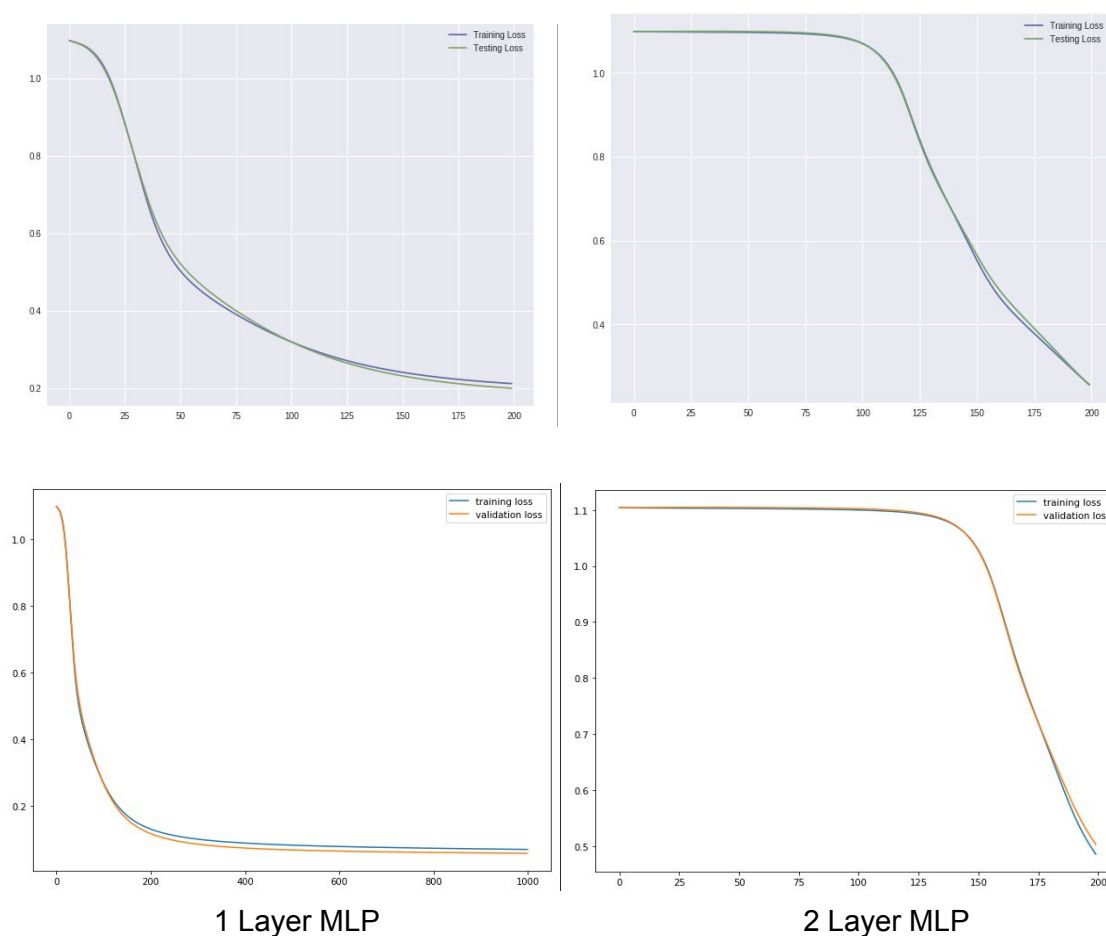


Figure 11: Left side is plots of 1 Layer MLP and right side of 2 Layer MLP

Observations:

From the graph we can conclude that for 2 Layer MLP the loss starts to fall after a considerable number of epochs, unlike 1 Layer MLP where the loss starts to fall quickly. Such behavior depicts that deep neural networks take a long time to converge. Also during

back propagation the change in derivative parameters is very less leading to vanishing gradient problems. This can be solve by using a nonlinear activation function like "RELU" and increasing the number of neurons in the hidden units as it will give the model more free parameters to use while converging. Deep neural networks performs very good when the data is highly complex. SInce, the IRIS data is simple we don't see much difference in the performance of 1 Layer MLP and 2 Layer MLPs. However, a considerable amount of difference can be seen when Deep Neural Networks are used on complex data sets like CIFAR100 etc.