IST 597: Foundations of Deep Learning
Homework 2: Back-Propagation of Errors & Multilayer Perceptron
Submitted by: Pakhi Agarwal

Problem 1. Back-Propagation of Errors

(a) Softmax Regression & the XOR Problem

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.

Sol. The table below shows the accuracy for different combinations of step-sizes and regularization strengths.

	Step Size ➔				
Reg ↓	**0.001**	**0.01**	**0.1**	**1.0**	**10.0**
0.001	50 %	50 %	75%	75%	50 % (nan)
0.01	50%	50 %	75%	75%	50 %
0.1	50 %	50 %	75%	75%	50 %
1.0	50 %	50 %	50 %	50%	50 % (nan)
10.0	50 %	50 %	50 %	50 % (nan)	50 % (nan)

Figure 1. Hyper parameter settings and the resultant Accuracy of Model (loss is nan)

This is the link to the file depicting the training process by recording the loss values and training accuracy → https://github.com/PakhiAgarwal/IST-597-Deep-Learning/blob/master/HW2/Deep%20learning%20Problem%201a.ipynb

As the softmax regression model we've built is a linear model, therefore it performs poorly, as a result we cannot achieve 100% accuracy using linear decision boundaries.

We achieved a best accuracy of 75% with number of epochs set to 100.

(b) Softmax Regression & Spiral problem

Please save and update your answer document with the generated plot once you have fit the softmax regression model as best you can to the data. Make sure you comment on your observations and describe any insights/steps taken in tuning the model. Do NOT forget to report your accuracy.

Sol. The figure below shows plot of Softmax regression model to the spiral data. With the best hyperparameter configuration having Step-size = 0.1 and regularization = 0.01 , we achieved an *accuracy of 56%.*

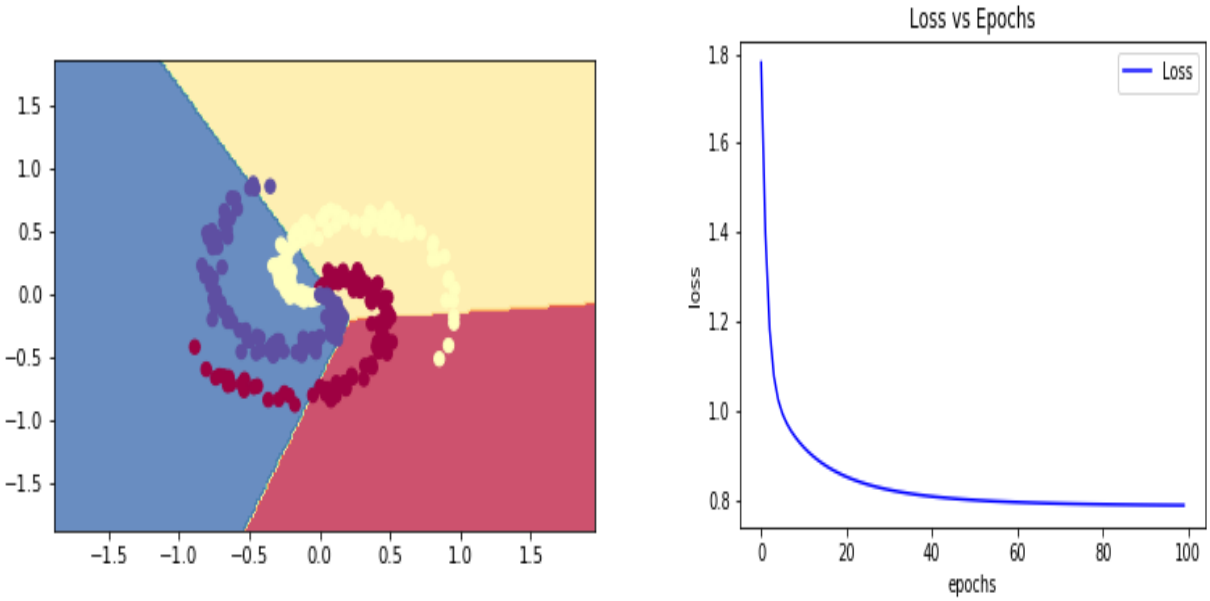


Figure 2. Decision boundary and Loss vs. Epoch for best accuracy achieved

The table below shows the different hyper-parameter settings and various observations made.

	Step Size ➔				
Reg ⬇	**0.001**	**0.01**	**0.1**	**1.0**	**10.0**
0.001	33.33%	33.33%	50%	49%	55.33 %
0.01	33.33%	33.33%	56%	49%	47.33%
0.1	33.33%	33.33%	49.67%	49%	47.67 %
1.0	33.33%	47.67%	50 .33%	50.33%	33.33 % (nan)
10.0	33.33%	33.33%	33.33%	33.33% (nan)	33.33 % (nan)

Figure 3. Hyper-parameter settings and resultant accuracies (loss is nan)

We had following observations from the parameter settings we did:

1. Having a high regularization strength makes the model constricted to loss.
2. The model diverges when the learning rate is too high.
3. As the data points are spirally distributed which makes it non-linear while the model is linear, it performs poorly. Because of this even tuning of hyper-parameters also doesn't have much effect over the accuracy of model.

(c) MLPs & the XOR Problem

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

Sol. *The best accuracy observed was 99%* with respect to all hyper-parameter settings, with 6 hidden units.

The table below (table 4) shows different settings we tried over the model and their resultant accuracies.

	Step Size ➔				
Reg ↓	**0.001**	**0.01**	**0.1**	**1.0**	**10.0**
0.001	70%	70%	82.34%	90%	88 %
0.01	70%	70%	99%	90%	88%
0.1	70%	70%	73.67%	90%	88 %
1.0	70%	70%	78 .33%	90%	70 %
10.0	70%	70%	70%	70%	70 %

Figure 4. Hyper-parameter settings and resultant accuracies

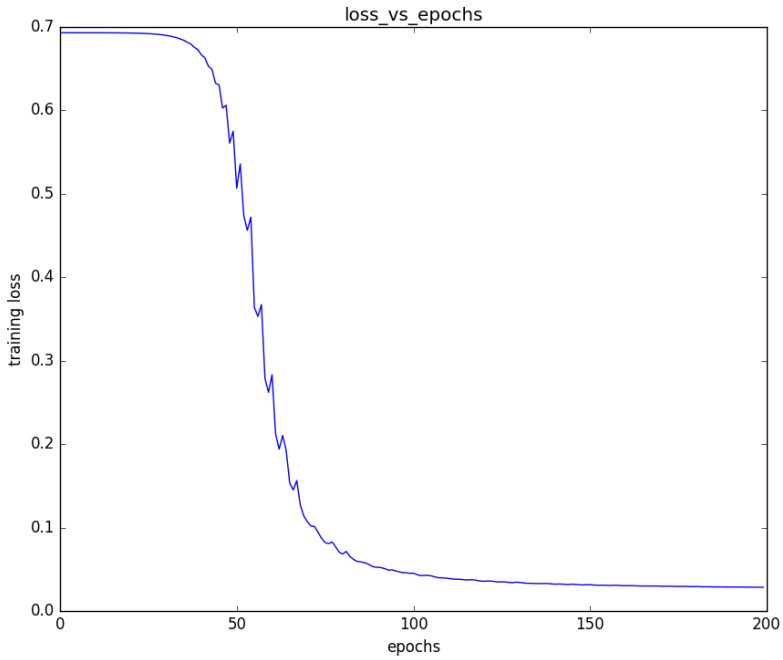


Figure 5. Loss vs. epoch

Yes, the MLP model was able to fit the XOR data. I think this is because the MLP model is non-linear which allows the model to divide the space into non-linear boundaries which inturn is required for separating XOR data.
The softmax regressor model is a linear model in comparison to MLP model, which is non-linear. So, the decision boundaries it can draw are in N-Dimension hyperplane.

(d) MLPs & the Spiral Problem
(d-1) 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.

Sol. Following were the observations by manipulating hyper-parameters and recording corresponding accuracies. n-epochs = 1500 for the observations below.

Hidden Units = 100

	Step Size ➔			
Reg ↓	**0.001**	**0.01**	**0.1**	**1.0**
0.0001	55.55%	52.23%	68%	96.66%
0.001	55.55%	52.23%	64.45%	92.23%
0.01	55.55%	50%	57.77%	78.88%
0.1	57.79%	48%	49.99%	54.44%

Hidden Units = 200

	Step Size ➔			
Reg ↓	**0.001**	**0.01**	**0.1**	**1.0**
0.0001	55.55%	50.55%	68.68%	98.88%
0.001	55.55%	50.55%	65.66%	96.66%
0.01	55.55%	50.55%	66.66%	75.55%
0.1	55.55%	47%	49.45%	50.33%

Hidden Units = 300

	Step Size ➔			
Reg ↓	**0.001**	**0.01**	**0.1**	**1.0**
0.0001	56%	49.99%	72.22%	98%
0.001	56%	49.99%	65.55%	97.77%
0.01	56%	49.99%	57.43%	79.76%
0.1	57.77%	45.55%	45.55%	55.55%

Figure 6. Different Hyper-parameter settings and resultant accuracies

The observations made from the results shown above were:

- As the number of epochs are fixed so an increase in learning rate, increases the accuracy of the model.
- An increase in regularization reduces the over-fitting nature of the model which inturn decreases the accuracy of the model.
- It could also be inferred that as the number of hidden layers increase, there is an increase in the accuracy of the model.
- An increase in number of epochs fits the model more accurately.

(d-2) 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? How did the regularization coefficient affect your model's decision boundary? Do NOT forget to report your accuracy.

Sol.

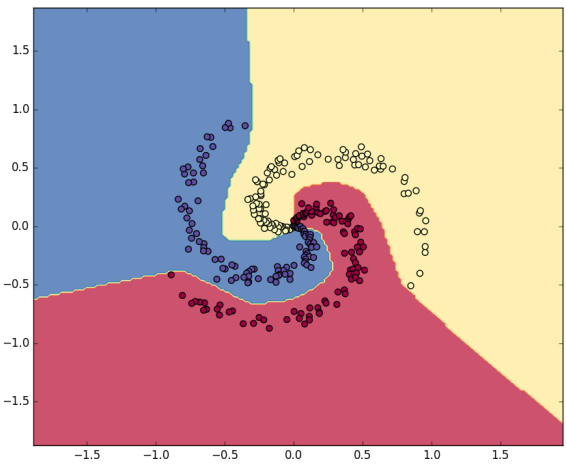


Figure 7. Decision boundary for best hyper-parameter settings

The difference between multinoulli regressor's decision boundary and MLP's decision boundary is that the MLP's decision boundary is non-linear while the multinoulli regressor's decision boundary is linear.

Yes, the MLP's decision boundary accurately captures the data distribution. Here, I observed that with the increase in regularization, the width of each of the spirals increases as the model tries to accommodate with a wide class of spirally distributed points, which in turn helps to generalize better.

The best *accuracy observed was 97%.* Here, the number of hidden layers is 200, n_epoch= 1500

	Step Size ➔			
Reg ⬇	**0.001**	**0.01**	**0.1**	**1.0**
0.0001	58.88 %	51.21 %	69.45%	97%
0.001	58.88%	51.21%	69.45%	95.44%
0.01	58.88%	51.21%	68.88%	70.77%
0.1	58.88%	50%	51.45%	49.33%

Figure 8. *Different Hyper-parameter settings and resultant accuracies*

Problem 2: Fitting MLPs to Data

(a) 1-Layer MLP for IRIS

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. Furthermore, consider the following questions:

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?

Sol.

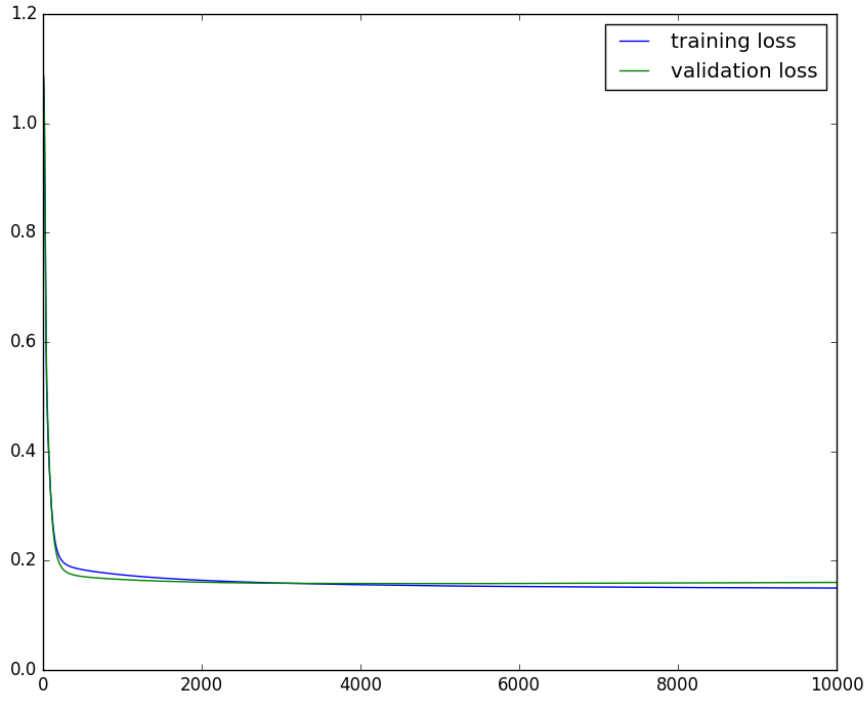


Figure 9. Loss vs epoch for best accuracy

Training accuracy = 97.98182%
Validation accuracy = 96.25%
We tried setting our model with n_epoch= 10000 and hidden layer = 100.

	Step Size (Train/Validation) ➔			
Reg ↓	**0.001**	**0.01**	**0.1**	**1.0**
0.001	97.98/95.45	97.98/96.25	95.55/94.45	33.33/31.31
0.01	97.76/95.45	96.25/95.56	93.33/97.77	33.33/31.31
0.1	95.55/92.23	95.55/94.33	95.45/90.99	33.33/31.31
1.0	35.55/30.33	35.55/32.33	33.33/31.31	33.33/31.31

Figure 10. Different Hyper-parameter settings and resultant accuracies

As per the question we tried training model with more number of epochs. We increased the number of epochs to 15000 and found that the training loss and validation loss at some point gets superimposed on each other. We also found that the difference between train loss and validation loss increases. The reason for this could be over-fitting of weights to noise in train data, which in turn performs more badly over validation data.
To control this we can-

- Try to regularize the model more strongly maybe by increasing the strength of L1/L2.
- We can also try using early stopping mechanism, in which we halt the training process if the performance of model on validation set does not improve after certain number of epochs.

(b) 2-Layer MLP for IRIS

Fit/tune your deeper MLP to the IRIS dataset and record your training/validation accuracies. Further- more, 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.

Sol.

	Step Size (Train/Validation) ➔			
Reg ↓	**0.001**	**0.01**	**0.1**	**1.0**
0.001	98.88/96.55	96.66/94.44	94.44/93.33	95.55/96.66
0.01	98.88/94.00	97.77/96.66	95.55/90.00	96.66/95.55
0.1	33.33/30.00	33.33/30.00	33.33/30.00	33.33/30.00
1.0	33.33/30.00	33.33/30.00	33.33/30.00	33.33/30.00

Figure 11. Different Hyper-parameter settings and resultant accuracies

The table above shows the different hyper-parameter values, which we tried to experiment with our model. Here, the number of epochs take were 10000 with a 100 hidden layers.

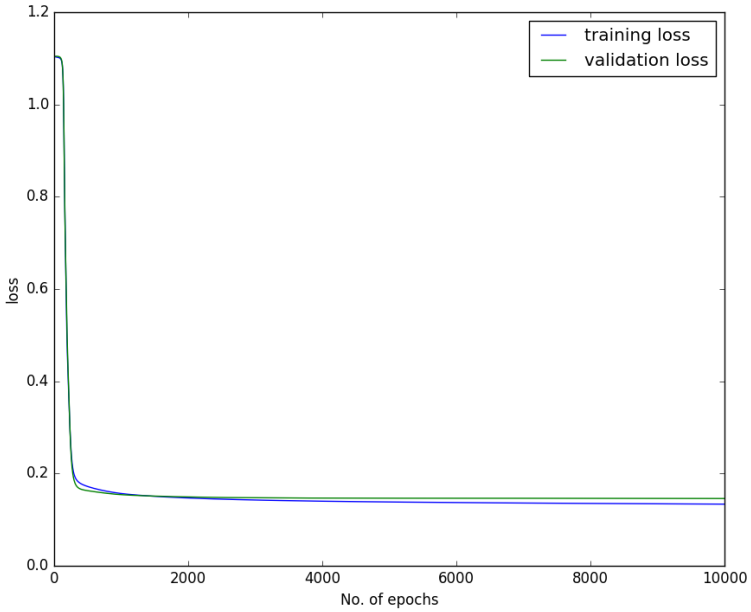
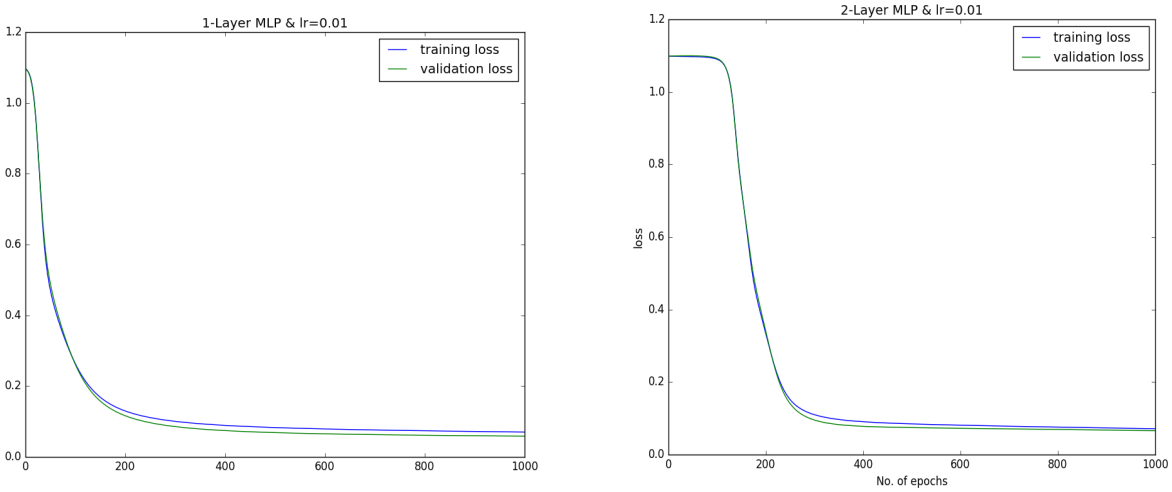


Figure 12. Loss vs. epoch

The above figure shows the loss versus epoch curve for both training and validation data.



(a) 1 layer MLP *(b) 2 layer MLP*

Figure 13. Loss vs. epoch

The above figure shows differentiation between the loss vs. epoch plots 1 layer MLP and 2 layer MLP.
According to the plot, it can be inferred that learning rate in 1 layer MLP converges faster than 2-layer MLP. The learning process in these networks is based over back propagation.
For this problem, the 2 layers in the dataset offers marginal values in terms of performance in comparison to 1 layer. Further for complex datasets, adding more layers will increase the performance of the model.