Foundations of Deep Learning - Assignment 2 - Answer Document

Submitted by : Anand Gopalakrishnan (aug440@psu.edu)

Problem 1 - Backpropagation of Errors

1) a) Softmax Regression & the XOR problem

Q) 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:

Gradient check result:

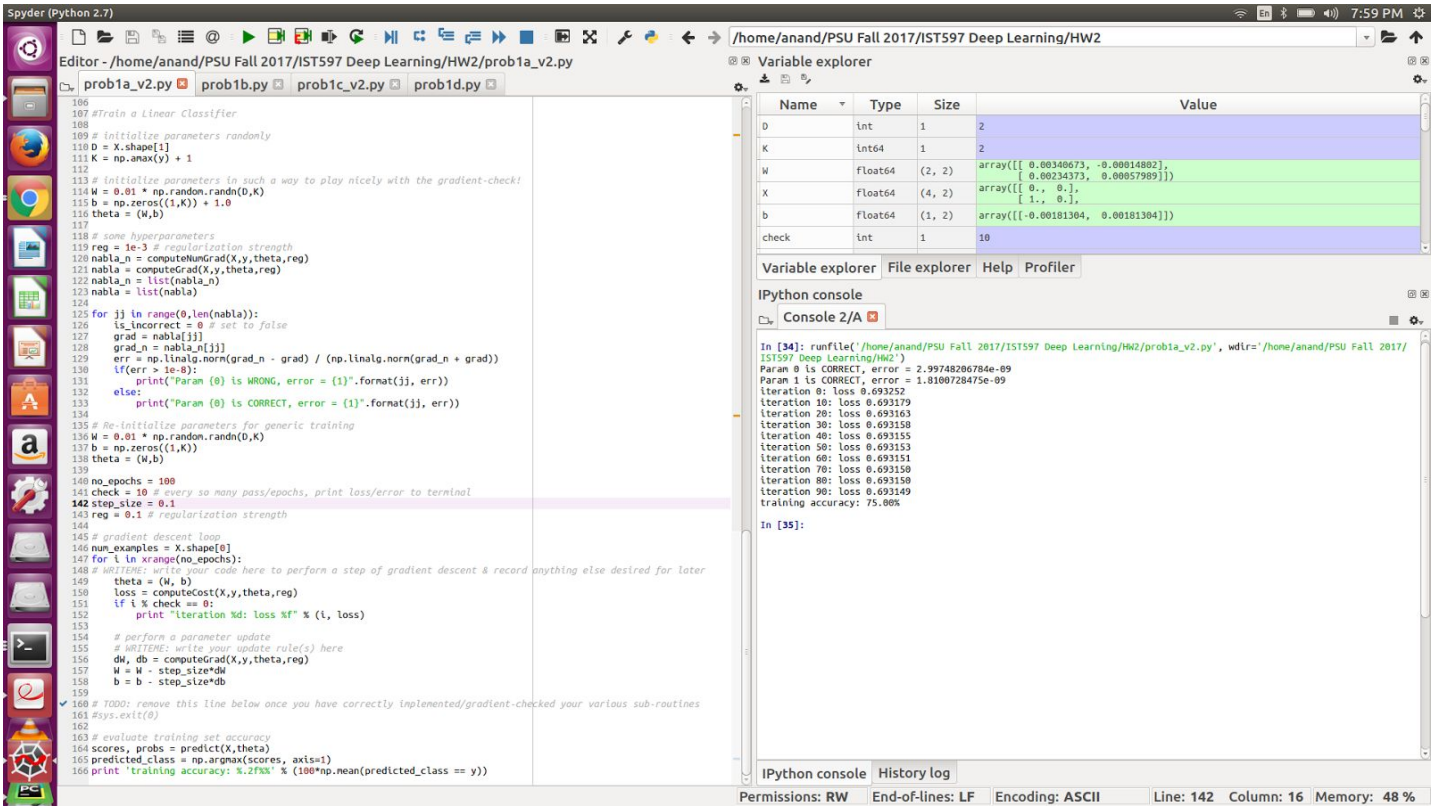


Figure 1. Gradient check results

The following table (Figure 2) shows the accuracy (and in some cases the loss values) for different combinations of step-sizes & regularization strength

	Step Sizes ----->				
Regularization	0.001	0.01	0.1	1.0	10.0
0.001	Accuracy = 25%	25%	50%	75%	50%
0.01	25%	25%	50%	75%	50%
0.1	25%	50%	75%	75%	50%
1.0	25%	50%	50%	50%	Acc = 50% loss=nan
10.0	25%	50%	50%	Acc = 50% loss = nan	Acc= 50% Loss = nan

Figure 2. Hyperparameter settings tested for 1) a)

The softmax regression model (with a single layer) we've built is a linear model and therefore it performs poorly, as the XOR dataset is a non-linear and therefore we CANNOT achieve 100% accuracy using a linear decision boundaries as given by the softmax model.

Best Accuracy achieved = 75% with learning rate = 0.1 ; regularization = 0.1; n_epochs = 100 (fixed)

1) b) Softmax Regression & the Spiral problem

Q) 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.

Ans: The following figure shows the plot of Softmax regression model to the spiral data with the best hyperparameter configuration: step-size = 0.01 ; regularization = 1.0;

Best Accuracy achieved = 50%**

The following figure (Figure 3) shows the decision boundary & loss_vs_epochs curve for the best accuracy settings:

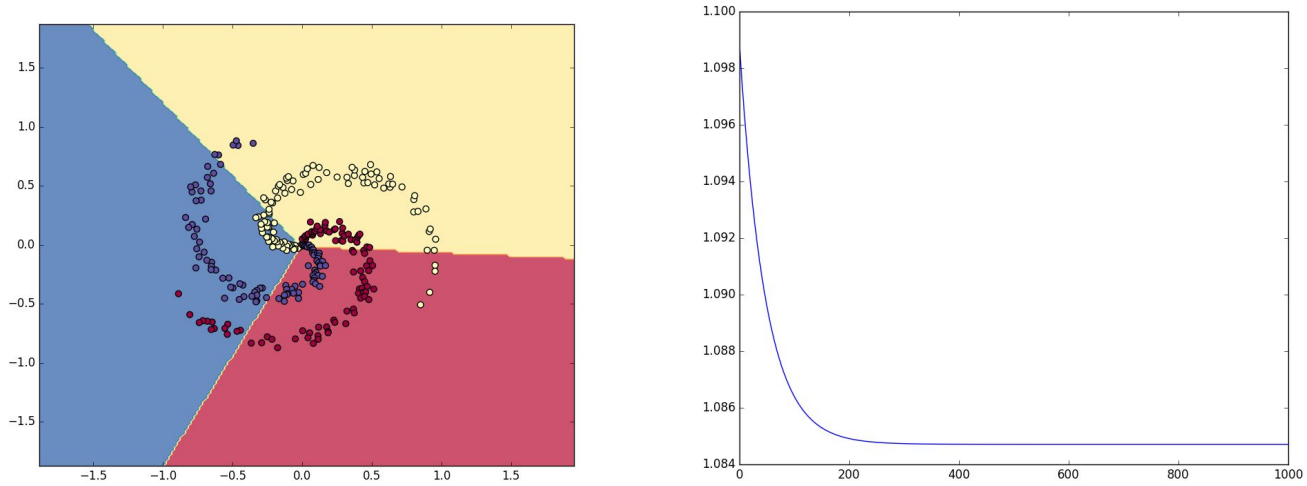


Figure 3. Decision Boundary & Loss_vs_Epochs plots for best accuracy achieved in 1) b)

** Hyperparameter settings achieving 59% accuracy not used as the decision boundary in that case is skewed towards one class of data points and does very poorly on the other 2 classes as shown in figure below:

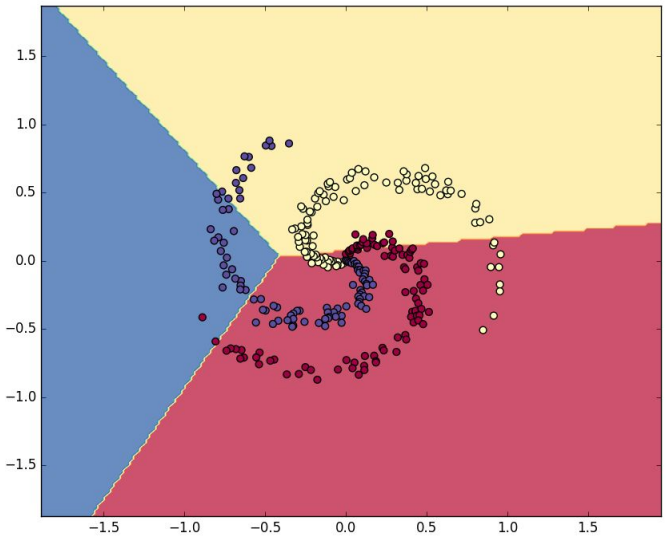


Figure 4. Skewed Decision Boundary for 1) b)

The following table (Figure 5) shows the accuracy (and in some cases the loss values) for different combinations of step-sizes & regularization strength

	Step Sizes ----->				
Regularization	0.001	0.01	0.1	1.0	10.0
0.001	Accuracy = 49%	49%	49%	49%	59%
0.01	49%	48%	49%	49%	47%
0.1	49%	49%	49%	49%	48%
1.0	49%	50%	50%	50%	Acc = 33% loss=nan
10.0	49%	50%	50%	Acc = 33% loss = nan	Acc= 33% Loss = nan

Figure 5. Hyperparameter settings tested for 1) b)

The observations of hyperparameter tuning process are as follows:

1) If the learning rate is too high (i.e. 10.0) the learning diverges
2) If the regularization strength is too high the model is highly constricted and loss
3) Since the model is linear whereas data points are spirally distributed (nonlinear), it performs poorly as it lacks the nonlinearity required to accurately capture data distribution. Also Tuning of hyperparameters does very little to help improve the accuracy.

1) c) MLPs & the XOR problem

Q) 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: Best Accuracy observed wrt to all hyperparameter settings experimented (as shown in Tables above) is for hidden units = 6; learning rate = 1.0; regularization strength= (0.001-0.1); **Best Accuracy = 100%**

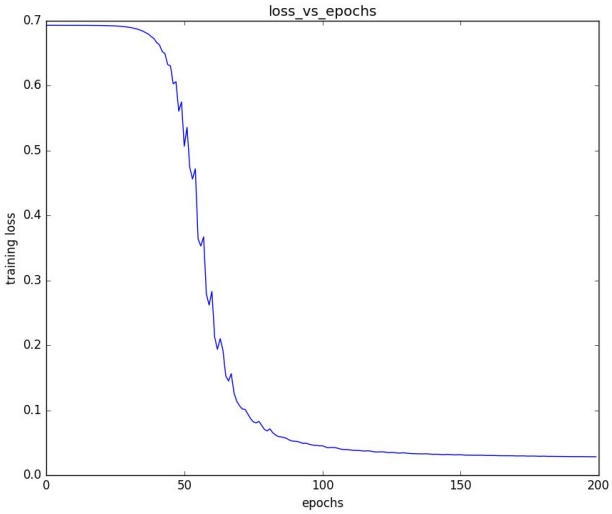


Figure 6. Loss_vs_epochs for 1) c)

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

Ans: MLP model was able to fit the XOR data perfectly, this is because the MLP model is a nonlinear model (nonlinear ReLU activation function) with a single hidden layer and this allows the model to partition the space into nonlinear decision boundaries which is required to separate XOR data.

The softmax regressor model on the other hand, is a linear model that is capable of linear separability. In other words, the decision boundaries it can draw are lines in 2-Dimension or planes/hyperplanes in N-Dimensions.

1) d) MLPs & the Spiral problem

Q) 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: The series of tables (in Figure 7) show the accuracies observed for various hyperparameter settings tested.

With hidden units = 100

	Step Sizes ----->			
Regularization	0.001	0.01	0.1	1.0
0.0001	Acc = 54.67%	50.33	65%	98.67%
0.001	54.67%	50.33%	63.67%	96.67%
0.01	54.67%	50.00%	56.33%	77.33%
0.1	55.67%	49%	48%	51.67%

With hidden units = 200

	Step Sizes ----->			
Regularization	0.001	0.01	0.1	1.0
0.0001	Acc = 54.67%	50.67%	67.67%	99.00%
0.001	54.67%	50.67%	65.67%	97.33%
0.01	54.67%	50.33%	56.33%	77.33%
0.1	54.33%	48.00%	47.33%	51.33%

With hidden units = 400

	Step Sizes ----->			
Regularization	0.001	0.01	0.1	1.0
0.0001	Acc = 56.00%	50.33%	70.33%	99.00%
0.001	56.00%	50.33%	68.33%	97.67%
0.01	56.00%	50.33%	57.00%	76.00%
0.1	56.67%	49.33%	48.67%	51.33%

Figure 7. Hyperparameter settings tested for 1) d)

n_epochs = 2000 for accuracies reported in tables in Figure 7

The trends observed in hyperparameter tuning process are as follows:

1) Since I've fixed the number of epochs in the experiments, as learning rate increases for all models, the accuracy increases (loss decreases) as it takes larger steps towards the minima.
2) For a model of given number of hidden units and learning rate, as regularization increases, the overfitting nature of the model to the training data reduces (accuracy decreases as model now fits training data noise lesser). However, a better way to estimate accuracies of hyperparameters would be on a separate validation dataset.
3) The model accuracy increases with increase in number of hidden units (with accuracy plateauing for hidden units = 400). This is because greater number of hidden units offers the model greater number of free parameters to use to better fit the nonlinear data distribution. However, beyond a certain point greater model complexity offers marginal/no improvements.

Q) 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?

Ans: The following figure (Figure 8) shows the decision boundary for tuned MLP (hyperparameter settings as reported below)

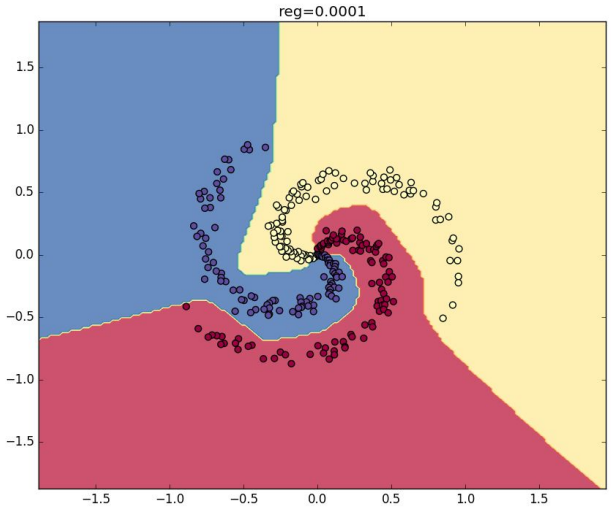


Figure 8. Decision Boundary for best hyperparameter settings for 1) d)
The MLP's decision boundary is nonlinear whereas the multinoulli regressor's decision boundary is linear.

Q) Does the MLP decision boundary accurately capture the data distribution? How did the regularization coefficient affect your model's decision boundary?

Ans: MLP decision boundary accurately captures the data distribution (spiral shape). As we regularize more strongly, the width of each of the spirals increases as it tries to accommodate a wider class of spirally distributed points to help it generalize better.

The following figures show how the shape of decision boundary changes with regularization (Hyperparameter settings: n_epochs = 2000; hidden_units = 400; learning_rate = 1.0 kept fixed)

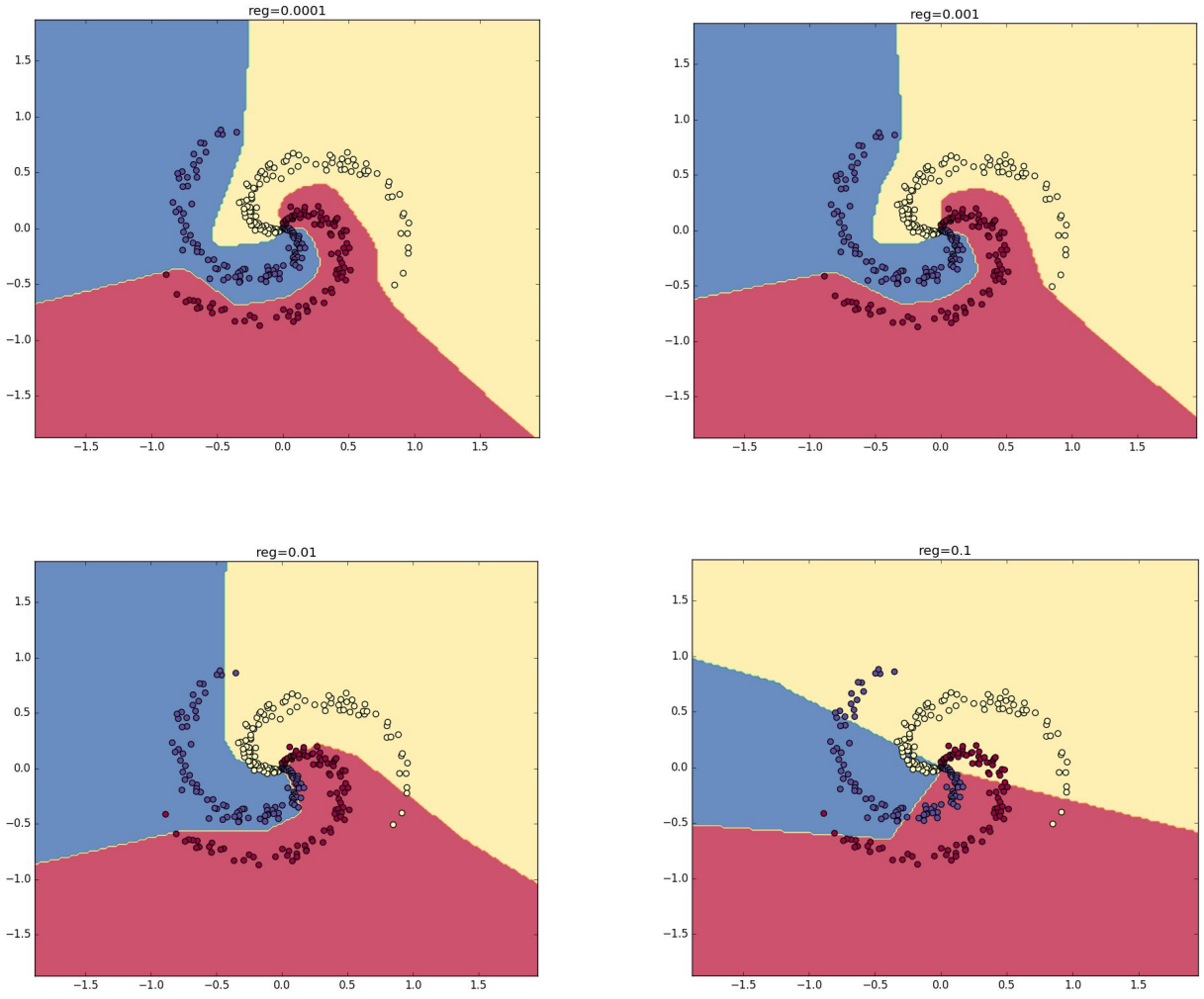


Figure 9. Shows the effect of regularization on shape of decision boundary

Q) Do NOT forget to report your accuracy.

Ans: Best Accuracy observed w.r.t to all hyperparameter settings experimented (as shown in Tables above) is for hidden units = 200; learning rate = 1.0; regularization strength= 0.0001; n_epochs=2000

Best Accuracy = 99.00% (best model with least model complexity chosen from all parameters tested as shown in series of Tables above)

Problem 2 - Fitting MLPs to Data

2) a) 1-Layer MLP for IRIS data

Q) 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: Train data Accuracy: 98.1818% Validation data Accuracy: 97.5% (model with least difference between training and validation accuracies chosen as it is representative of lesser overfitting)
For learning rate = 0.01; regularization strength = 0.01; n_epochs = 10000; hidden layer = 100 units

The following figure (Figure 10) shows the plot of loss as a function of epoch for both train and validation data superimposed (hyperparameter settings same as above)

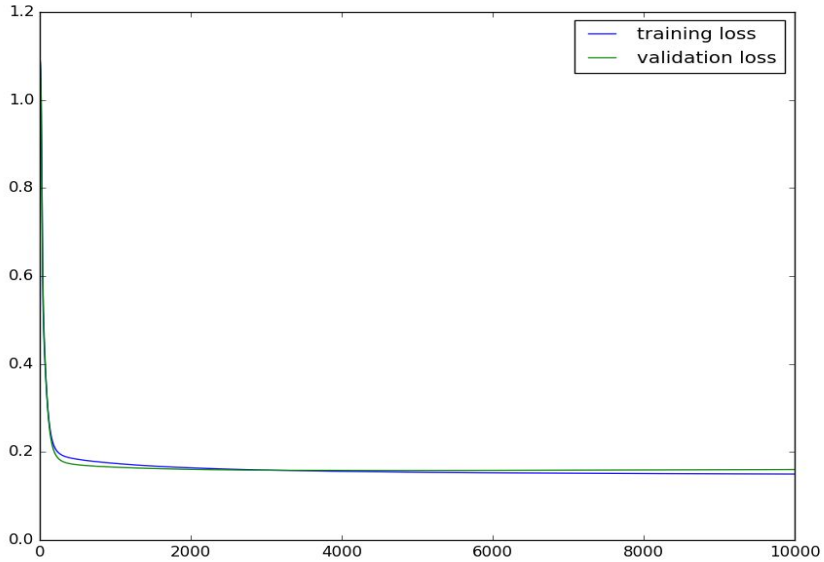


Figure 10. Shows loss vs epochs for best accuracy observed for 2) a)

The following Table (Figure 11) shows the hyperparameter values (learning rates and regularization strength) experimented for tuning the MLP model. Number of epochs & hidden units fixed at 10000 and 100 respectively. (Values in cell are training accuracy/validation accuracy)

	Step Sizes ----->			
Regularization	0.001	0.01	0.1	1.0
0.001	99.09/97.5	99.09/97.5	97.27/97.5	34.54/30.00
0.01	99.09/97.5	98.18/97.5	96.36/97.5	34.54/30.00
0.1	98.18/97.5	96.36/95.00	95.45/90.00	34.54/30.00
1.0	34.54/30.00	34.54/30.00	34.54/30.00	34.54/30.00

Figure 11. Hyperparameter settings tested for 2) a)

Q) 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?

Ans: The following figure (Figure 12) shows train data loss vs epochs and validation data loss vs epochs superimposed for more epochs (i.e. 20000)

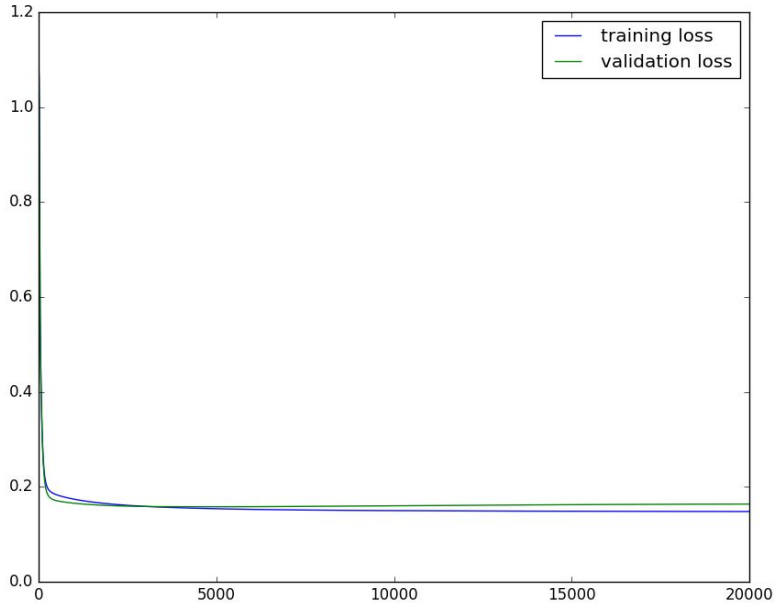


Figure 12. Shows loss vs epochs plots for train/validation data for more epochs

As we train the MLP for more epochs and plot the train loss and validation loss vs epochs we observe that the difference between the train loss and validation loss tends to widen (with train loss always being lower). This is because the model is now starting to overfit it's weights to the noise in the train data and is therefore performing worse on the validation data (loss of generalization ability).
We can alleviate this by more strongly regularizing the model i.e by increasing the strength of L1/L2 regularization terms or use a combination of L1/L2/gradient clipping instead of just one. Another strategy to prevent the model from overfitting to training data, is to use an early stopping mechanism, where we halt the training process if the model's performance on validation set does NOT improve after a certain number of epochs.

2) b) 2-Layer MLP for IRIS data

Q) Fit/tune your deeper MLP to the IRIS dataset and record your training/validation accuracies. Furthermore, create plots of your loss-versus-epoch curves as you did in Problem #1a. Put all of this in your answer document.

Ans: The following Table (Figure 13) shows the hyperparameter values (learning rates and regularization strength) experimented for tuning the MLP model. Number of epochs & hidden units fixed at 10000 and 100 respectively. Values in cell are training accuracy/validation accuracy

	Step Sizes ----->			
Regularization	0.001	0.01	0.1	1.0
0.001	99.09/97.5	97.27/95.0	96.36/97.5	34.54/30.00
0.01	99.09/95.00	98.18/97.5	97.27/95.0	34.54/30.00
0.1	34.54/30.00	34.54/30.00	34.54/30.00	34.54/30.00
1.0	34.54/30.00	34.54/30.00	34.54/30.00	34.54/30.00

Figure 13. Shows hyperparameter settings tested for 2) b)

The following figure (Figure 14) shows the loss-versus-epochs curves for both training and validation data

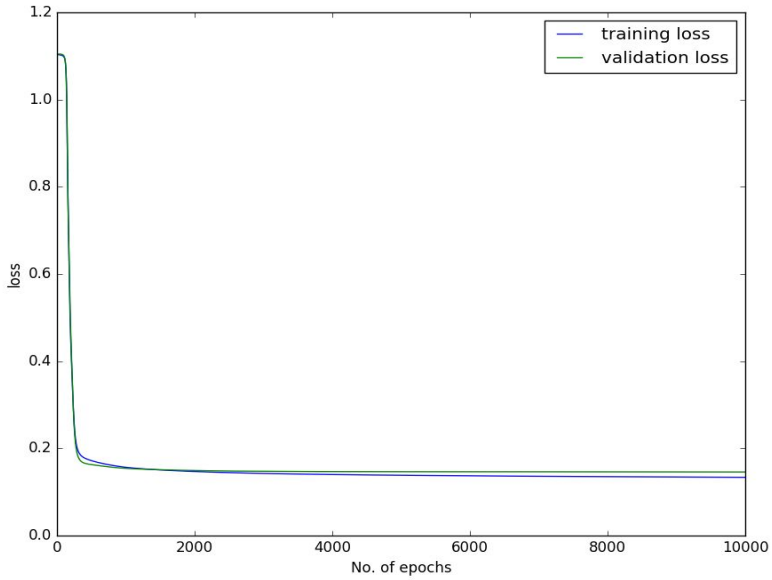
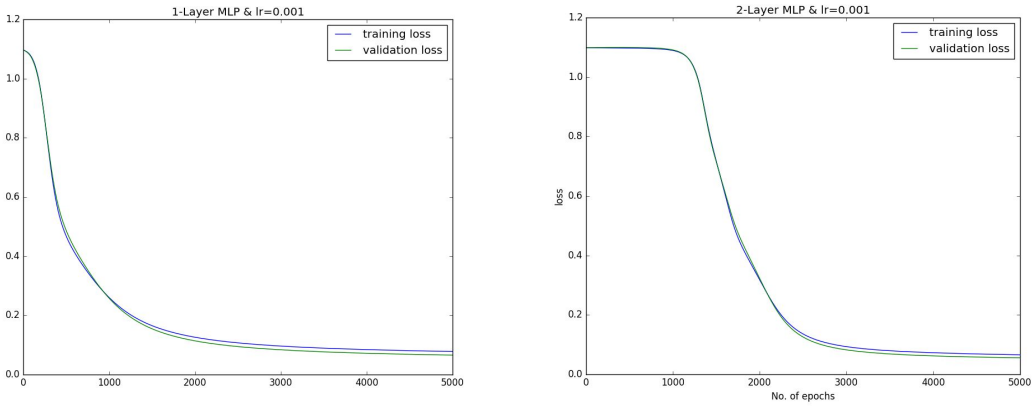


Figure 14. Shows loss vs epochs curves training and validation data for 2) b)

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

Ans: The following figure (Figure 15) shows a series of loss_vs_epochs curves for different learning rates for 1-layer and 2-layer MLPs with regularization and all other parameters of learning kept the same.



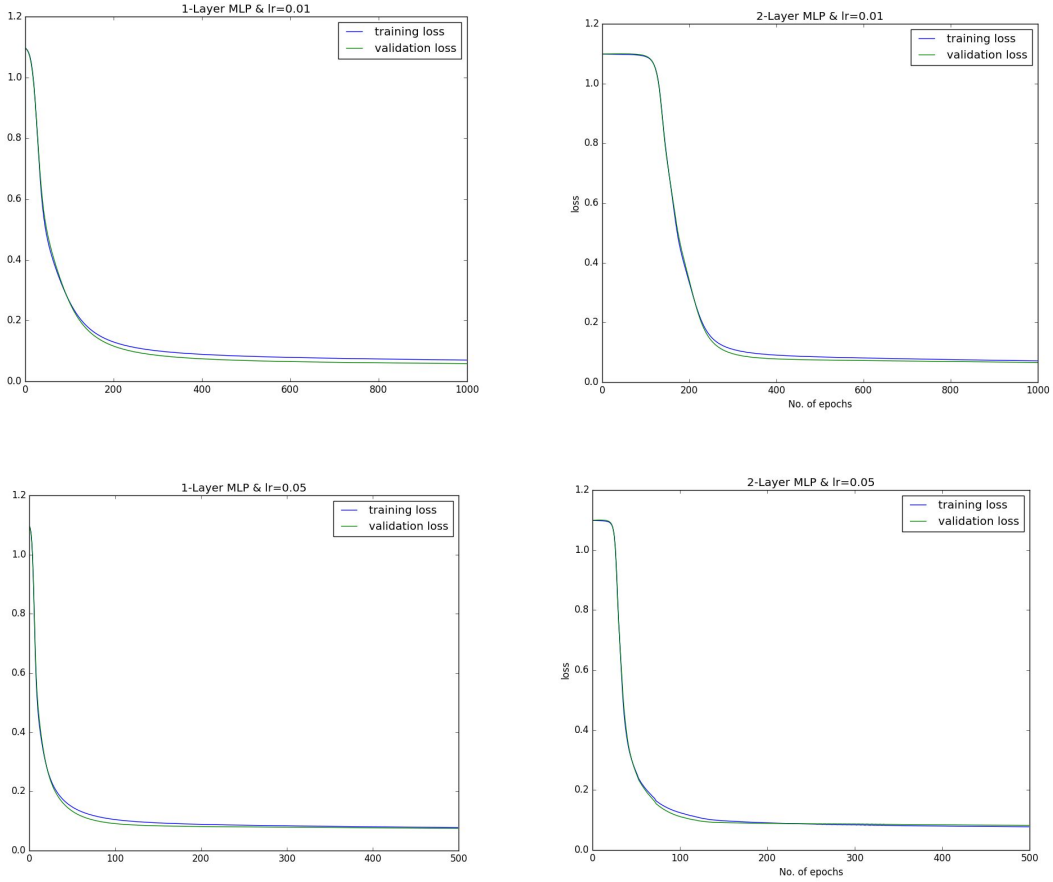


Figure 15. loss vs epochs comparison between 1-layer (left-panel) & 2-layer (right-panel) MLP

A common theme between the figures in the left-panel and right-panel of Figure 15, is that the curves for the left-panel plateau much quicker than in right-panel. This tells us that the learning process in the 1-layer MLP converges much faster than that in case of the 2-layer MLP. The workhorse of the learning process in these networks is backpropagation and the weights get updated (learning process continues) proportional to the error gradients they receive. Hence in deeper networks, we experience the vanishing gradient problem wherein the gradients are progressively diminished as they flow back to the lower layers leading to longer time (more epochs) needed to reach convergence.

However, deeper networks have the advantage that they can model more complex data distributions using a composition of nonlinear functions. In this problem, for the IRIS dataset a 2-layer offers marginal/no value in terms of performance compared 1-layer as the dataset is simple and features are low-dimensional. For very high-dimensional, large and complex datasets, increasing the layers in the networks will offer tremendous boost in performance.