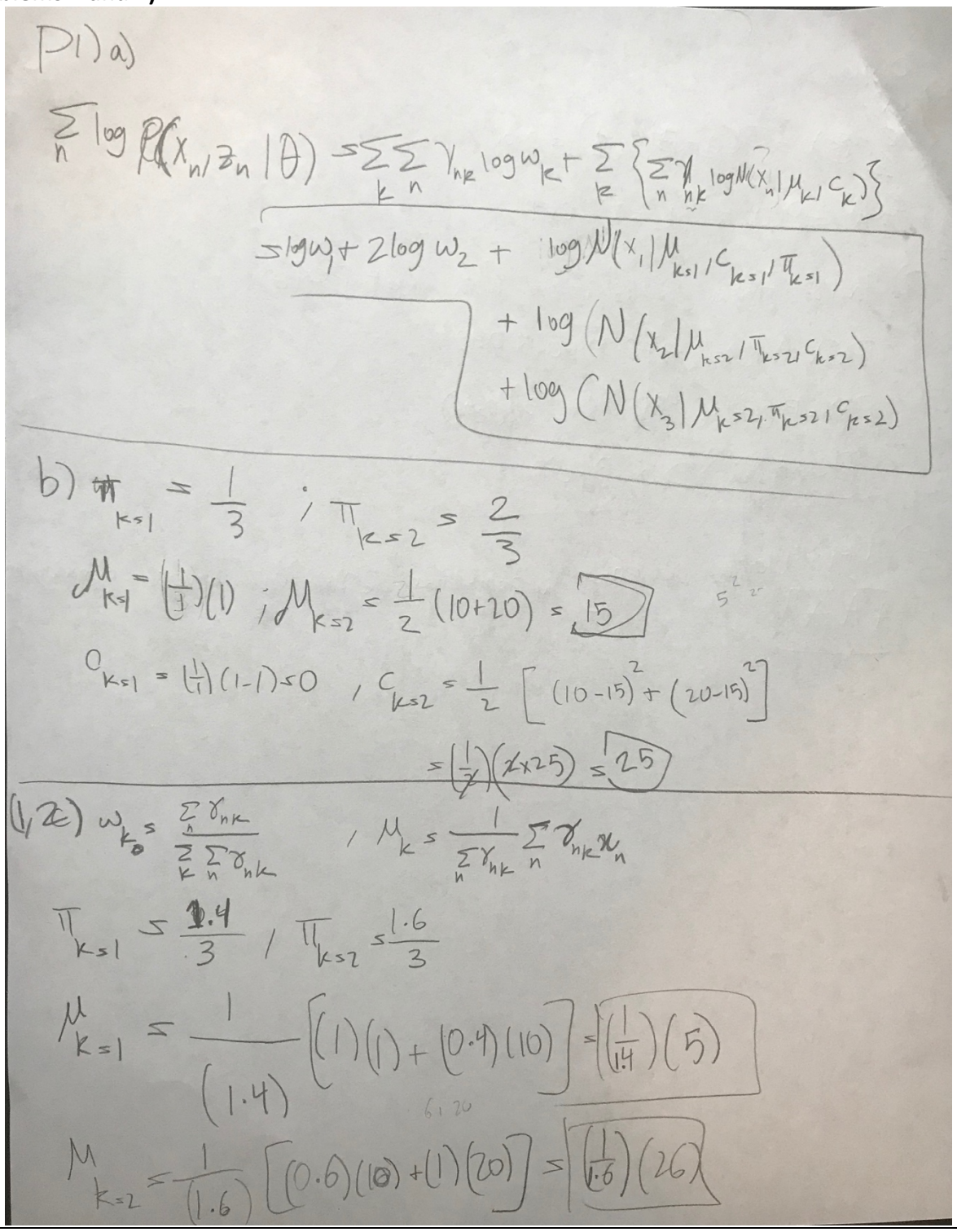
Milad Nourian
ECE M146
HW5
Professor Diggavi

Poblems 1 and 2)



Problem 2)

a)

$e_1(B) = P(O_t = B \mid q_t = 1) = 0.01$

$e_2(A) = P(O_t = A \mid q_t = 2) = 0.41$

b) The most frequent symbol is one with largest $P(X \mid \theta)$

$P(X = A \mid \theta) = (0.49)\,0.99 + 0.51 \cdot 0.49$

$P(X = B \mid \theta) = (0.49)(0.01) + (0.51)(0.51)$

$P(X = A \mid \theta) > P(X = B \mid \theta)$

---

c) It is clear by calculating the probabilities the symbol with highest probabil

AAA

P3)a)

The data is split in two part using sklearn.train_test_split ()

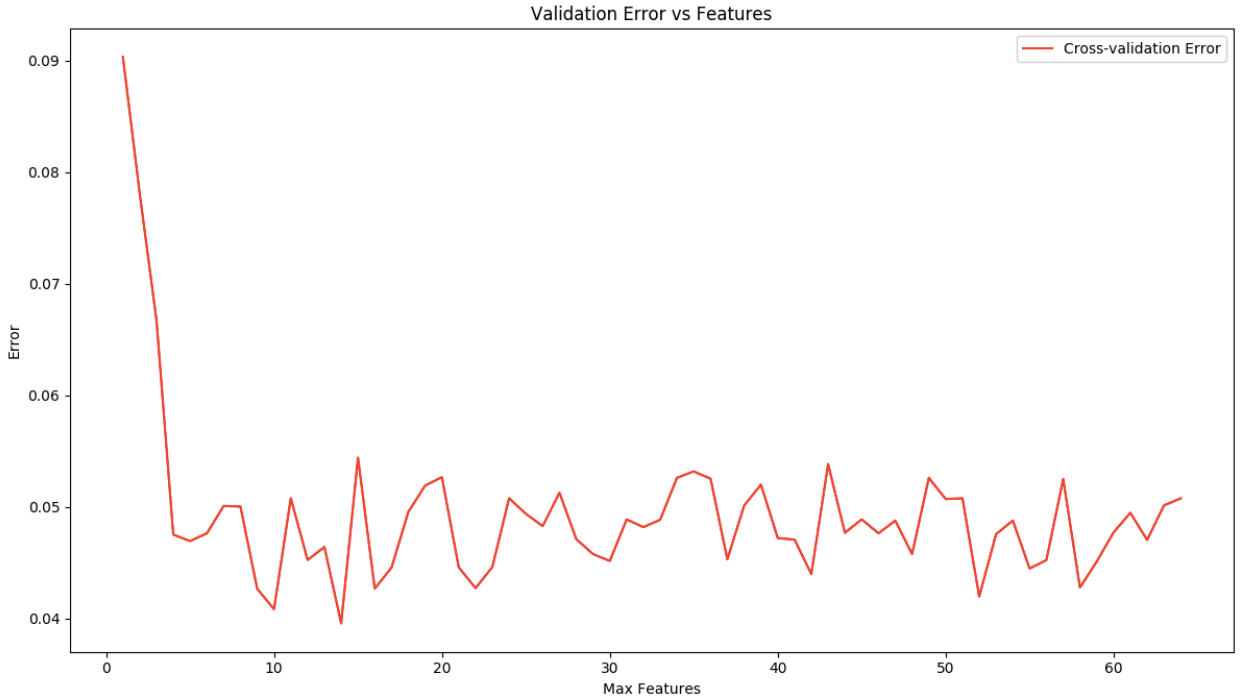
P3)b)

Using the cross_validation function, using sklearn, the error found was **0.14495188989709984**.

P3)c)

Using the bagging classifier from sklean, the CV score (from cross_val_score) is found to be **0.050778474279623564**. We notice that using the bagging that uses sampling from the original data set to construct a classifier works much better than simply just using a weak learner such as a decision tree as the classifier.

P3)d)

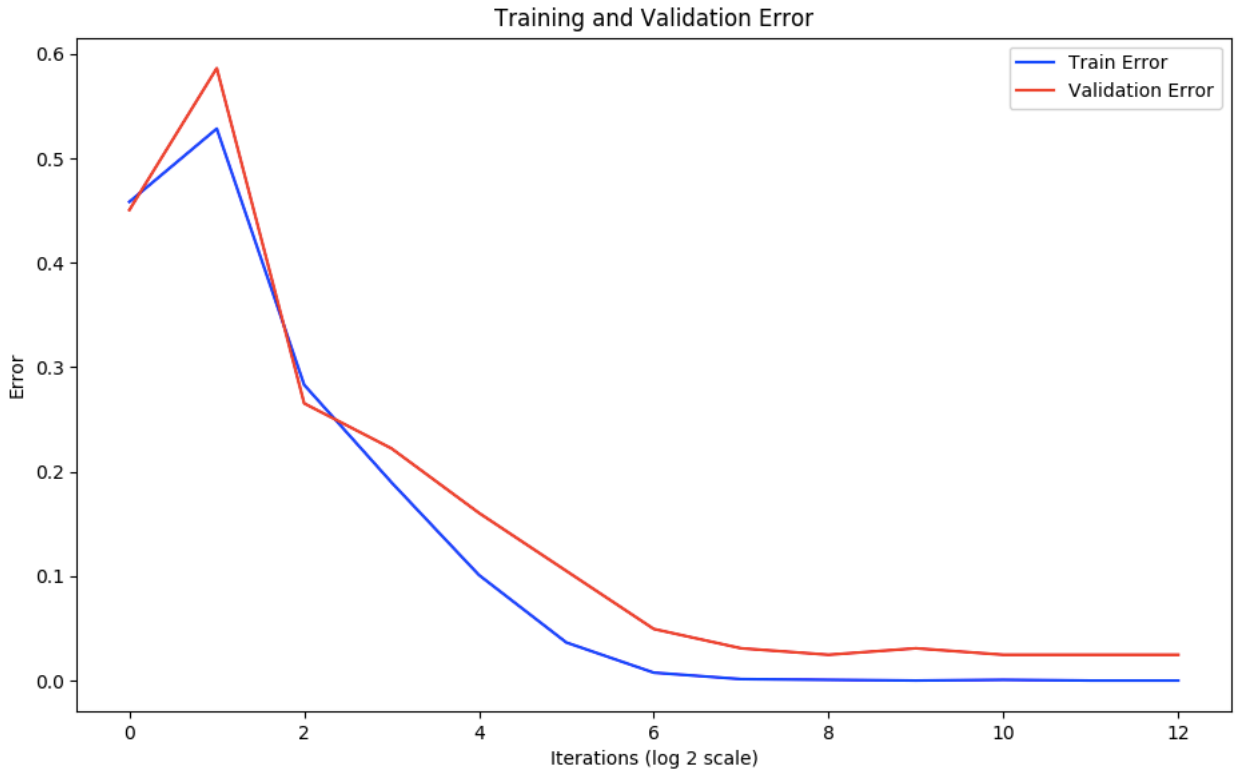


Random Forest classifier is a subclass of bagging classifier where we set the max_features for each node. This means that we set the max_features to be the number of samples that we **randomly** consider on each nodes and then split on one of those nodes (the best one), in this way, we introduce randomness into the decision tree classifier and reduce the variance.

By looping over the hyperparameter max_features we found that best value for the number of features to sample from randomly is **14**, and the cross validation error for random forest is **0.03957504383281463** better than the error we saw with just the bagging algorithm.

P3)e)

The graph for error vs number of iterations is shown below. As shown in the plot, the minimum value of error for the AdaBoostAlgorithm is found to be **0.024691358024691357** and the optimal number of iterations is **256**.



P3)f)

Using the best hyperparameters we found for num_features for the random forest algorithm and best number of estimators we found for the AdaBoost algorithm, we obtained the following results:

('Bagging test error', 0.06666666666666665)

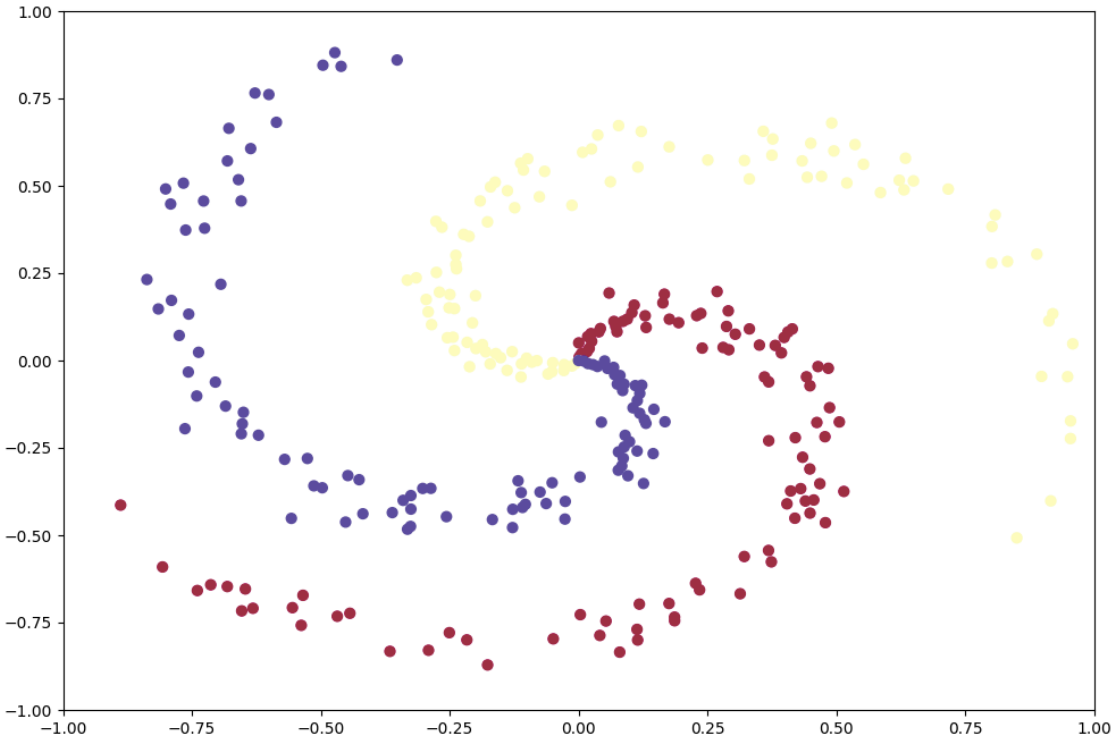
('Random forest test error', 0.05555555555555558)

('Boosted tree test error', 0.03888888888888886)

Clearly boosted algorithm (with a tuned number of estimators) outperforms the other 2 classification algorithm.

P4)a)

The visualization of the data is shown below. As seen from plots, the dimensionality of the data is 2 (2_D), number of classes is 3, and data is not linearly separable which means in order to obtain better results, we can use a non-linear classifier such a neural network.



P4)b)

Let $s = \begin{bmatrix} s0 \\ s1 \end{bmatrix}$, now find $\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{s}) = \frac{\boldsymbol{e^{s0}}}{\boldsymbol{e^{s0}+e^{s1}}} = \frac{\boldsymbol{1}}{\boldsymbol{1+e^{s1-s0}}} = \frac{\boldsymbol{1}}{\boldsymbol{1+e^{-s}}}$ where we set s= s1 – s0. As seen softmax for 2D case is equivalent to sigmoid function.

P4)c)

We know

$$L_{CE} = -\sum_{k=1}^{K} t_k \log\, y_k$$

and now we take the partial derivative:

$$L = -[t_k log(y_k)]$$

as there would be only one term where the indicator function is non-zero.

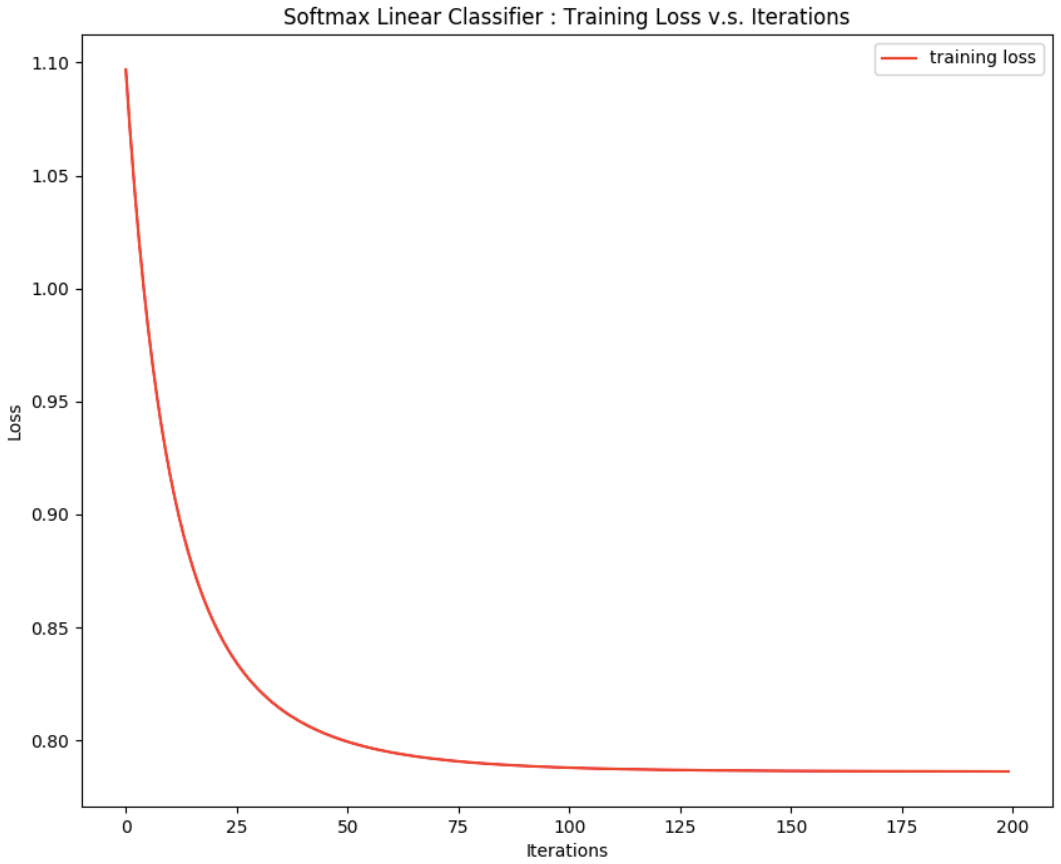
$$\frac{\partial L}{\partial s_k} = -t_k \frac{1}{y_k} \frac{\partial}{\partial s_k} y_k = -t_k \left(\frac{1}{y_k}\right)(y_k)(1 - y_k)$$

and we eventually get

$$\frac{\partial L}{\partial s_k} = y_k - t_k$$

as $y_k * t_k = 1$. Therefore, we have found the partial derivative and gradient of cross-entropy loss for a neural network.

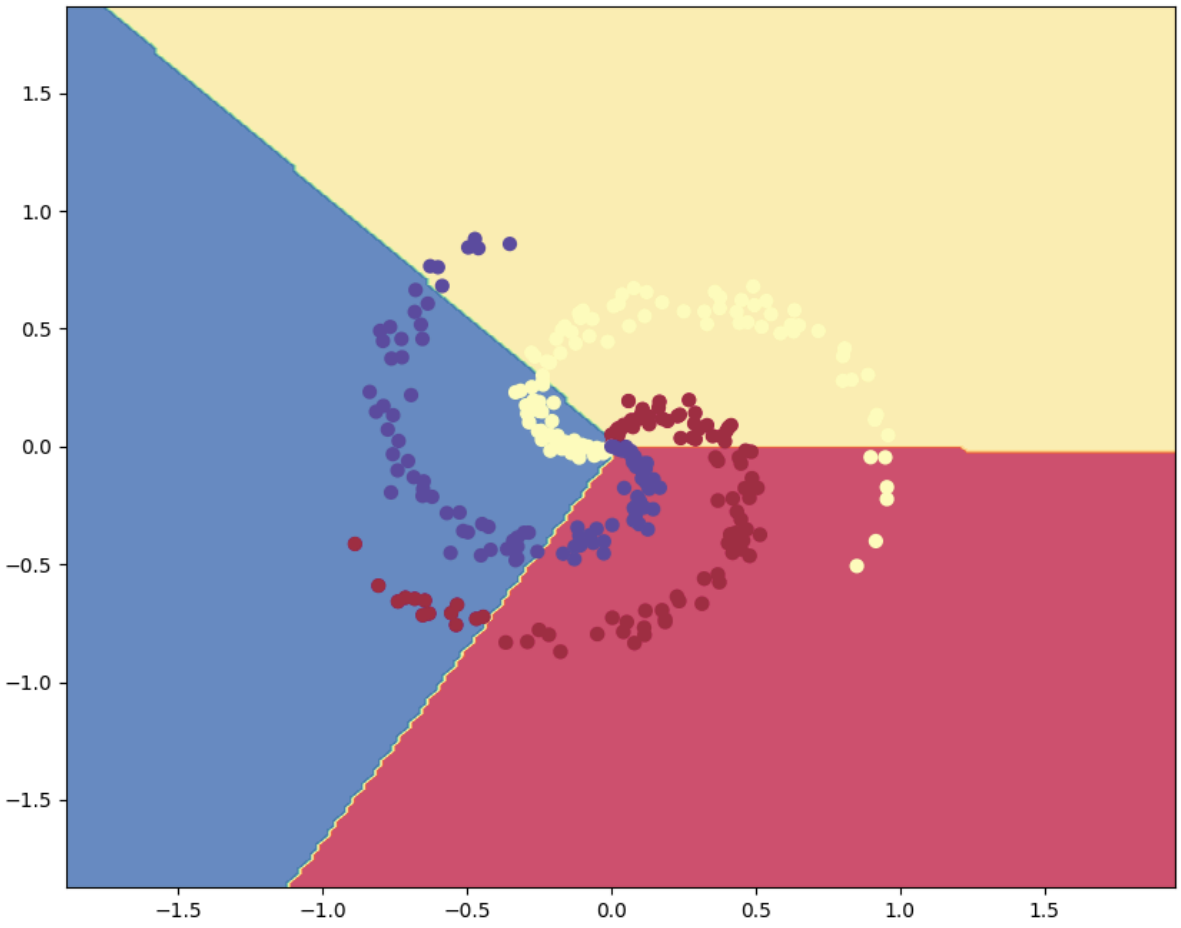
The learning curve (training error vs. number of iterations) is shown below. It is evident that as the number of iterations increase, the training error decreases. This also makes sense intuitivly as with more iterations, we train the model better by using more iterations of the gradient descent to decrease the loss function. Also, the decision boundry of the Neural network is shown below. We can see that a shallow neural network produced a non-linear decision boundry. We can increase the accuracy by increasing the depth of the neural network. From the result we can see that the error for the softmax model is **0.49.**



$$\frac{\partial L}{\partial s_k} = -t_k \frac{1}{y_k} \frac{\partial}{\partial s_k} y_k = -t_k \left(\frac{1}{y_k}\right)(y_k)(1 - y_k)$$

$$\frac{\partial L}{\partial s_k} = -t_k \frac{1}{y_k} \frac{\partial}{\partial s_k} y_k = -t_k \left(\frac{1}{y_k}\right)(y_k)(1 - y_k)$$

=---dc



P4)e)

The learning curve of the deeper neural network (with 2 layers) is shown below. Again, we can see that the training error reduces with the number of iteration. There are some spikes in the graph, and that can be due to a wrong direction of movement in the gradient descent but in general the learning curve is downward. Also, we can se

Accuracy is **0.98** for the case of neural network with 2 layers, which is less than 0.49 for the softmax case. We can see that the neural network classifies the points almost perfectly and with a very low error so neural network outperforms the softmax classifier and the deeper the Neural network, we would expect a higher performance as long as we can avoid overfitting.

P4)f)

The training accuracy for the 2 layer neural network is 0.98 (0.02 error) and the training accuracy for the linear model used in part is 0.49 (0.51 error). Evidently, we can see that the neural network performs a much better job at classifying a non-linear distribution. We see that our shallow neural network with only 2 layers perform a much better job than a linear model,

and we can improve performance even on more complex data sets by increasing the number of layers in the neural network.