DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, IIT MADRAS

CS6910: Programming Assignment 1

Team 37

Sheth Dev Yashpal
CS17B106

Harshit Kedia
CS17B103

1 FUNCTION APPROXIMATION

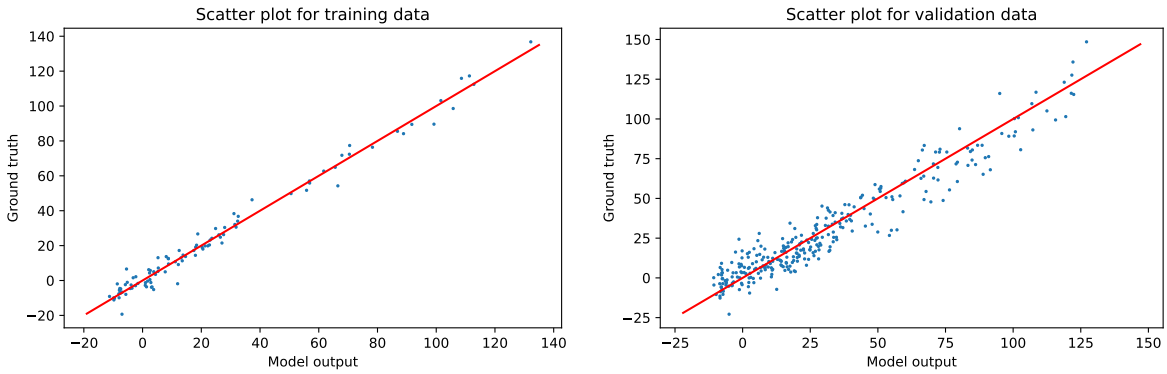

Figure 1.1: Scatter Plot of training data and validation data for the Function Approximation task.

1.1 NETWORK SPECIFICATION

The input has layer has 2 nodes and output layer has 1 node. Each of the Hidden Layers have 50 nodes each. The value of $\eta = 0.0002$ and the value of $\alpha = 0.9$ for the Generalized Delta Rule. We use *tanh* activation function in hidden layer nodes and *linear* activation for output layer.

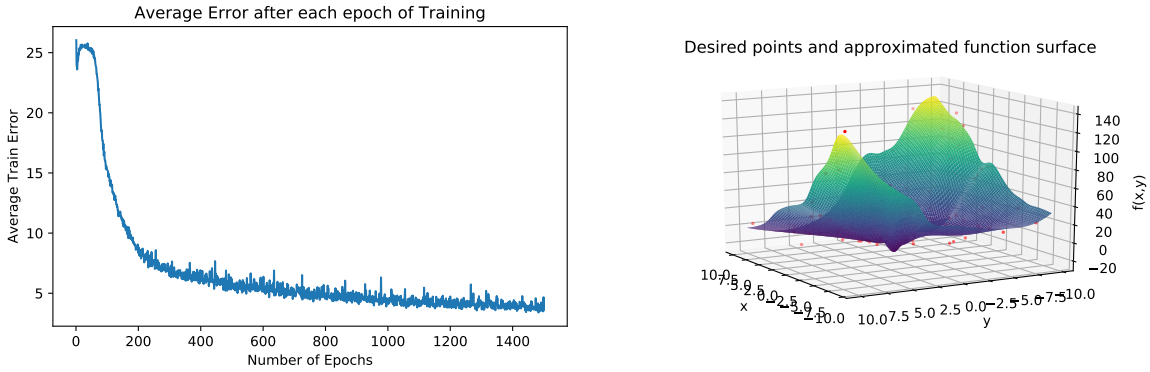

Figure 1.2: The plot of train error vs the number of epochs of training as well as the desired an estimated function outputs.

1.2 Observations:

- For the function approximation task within an average absolute error value of **3** approaching 1500 epochs. On the validation set, the error is slightly more than that and goes up to **8**.
- The issue with using less number of nodes in the hidden layers is that the training gets stuck in some local minima where the error is still quite high and therefore we are not able to get good results. Increasing the number of nodes in the hidden layer introduces additional non-linearity and parameters which solves the problem.
- Another problem while training was with the saturating activations for *tanh*. To solve this, we introduced the $\beta = 0.1$ parameter for the *tanh* activation and initialized the weights over a larger range of $[-2.5, 2.5]$. This solved the issue of saturating values as well as avoided the vanishing gradients due to low initialized weights.
- We can observe from the plots that the train error indeed starts to stabilize after a 1000 epochs and our approximated function more or less follows the trend given by the scatter of train data. The goodness of our approximation is further validated by our scatter plots where almost all the examples are close to the $x = y$ line of truth vs approximated.

2 2D Non-Linear Classification

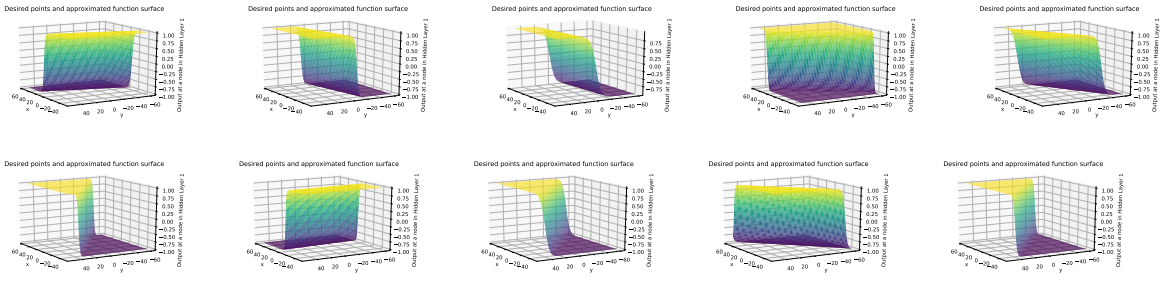

Figure 2.1: Output of nodes at Hidden Layer 1 after 1 epoch.

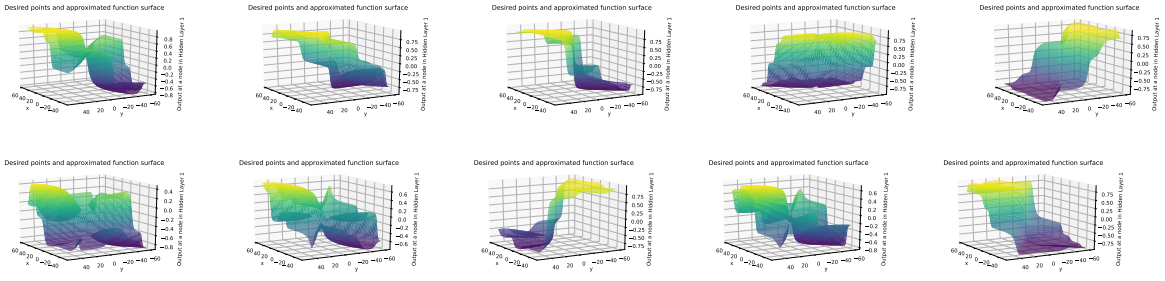

Figure 2.2: Output of nodes at Hidden Layer 2 after 1 epoch.

2.1 Network Specification

The input has layer has 2 nodes and output layer has 3 nodes. Each of the Hidden Layers have 10 nodes each. The value of $\eta = 0.002$ and the value of $\alpha = 0.1$ for the Generalized Delta Rule. We use *tanh* activation function in hidden layer nodes and *softmax* activation for output layer.

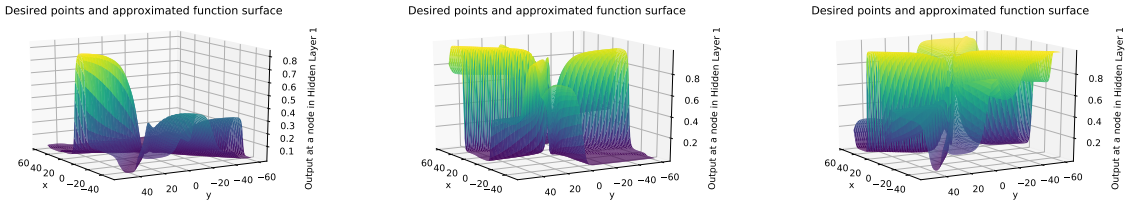

Figure 2.3: Output of nodes at Output Layer after 1 epoch.

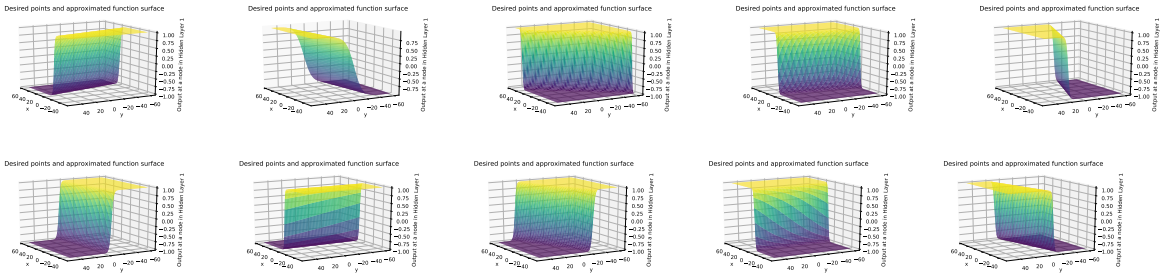

Figure 2.4: Output of nodes at Hidden Layer 1 after 50 epoch.

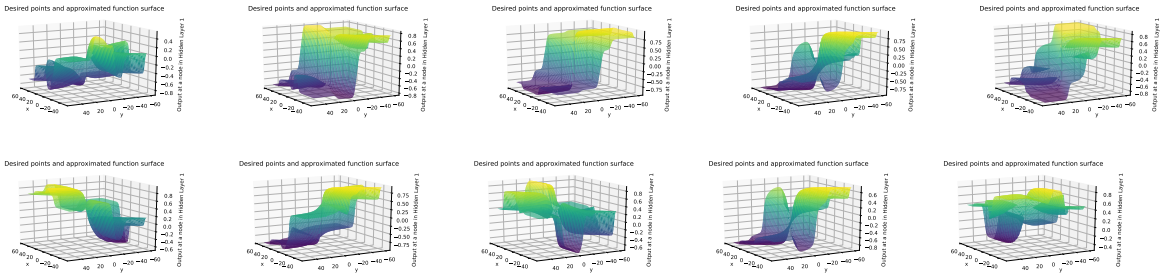

Figure 2.5: Output of nodes at Hidden Layer 2 after 50 epoch.

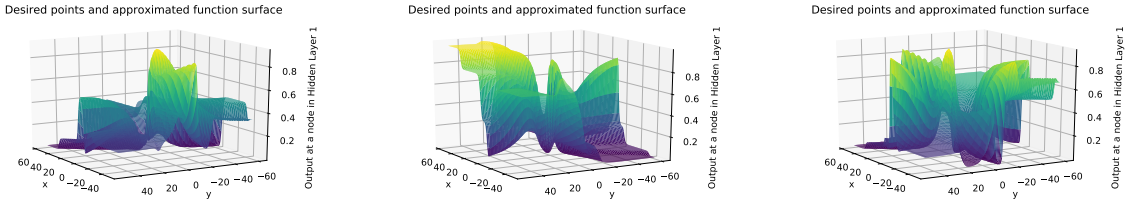

Figure 2.6: Output of nodes at Output Layer after 50 epoch.

2.2 Observations:

- The 2D Non-Linear task was particularly hard because of the fact that two of the three given classes were interleaved and were only marginally separable as evident from the decision regions and scatter plot of training data.
- Since the classes weren't quite separable, the training required a large number of epochs and therefore showing plots for the outputs of hidden layer nodes after $2, 10, 50$ epochs wasn't very insightful. Therefore, in-order to get more insight on how these functions change we have presented the plots of outputs of hidden layer nodes after $1, 50, 500 and 1000$ epochs.

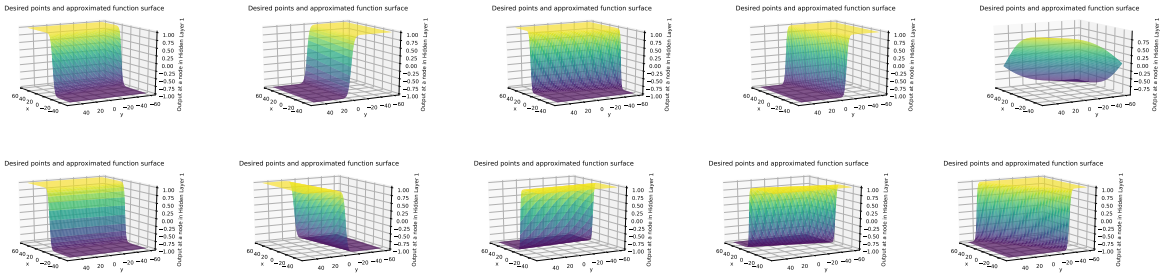

Figure 2.7: Output of nodes at Hidden Layer 1 after 500 epoch.

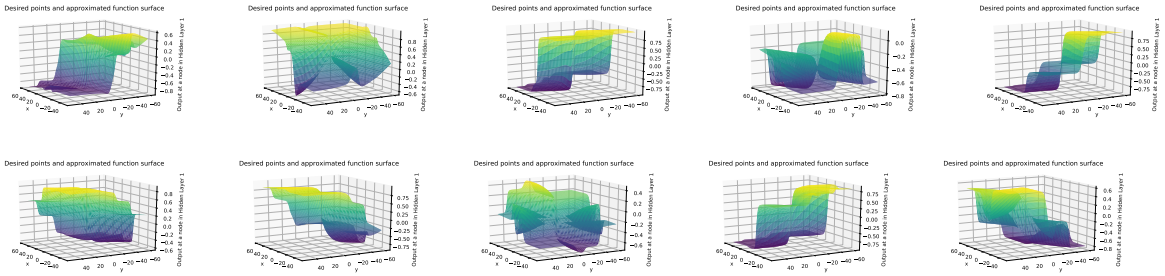

Figure 2.8: Output of nodes at Hidden Layer 2 after 500 epoch.

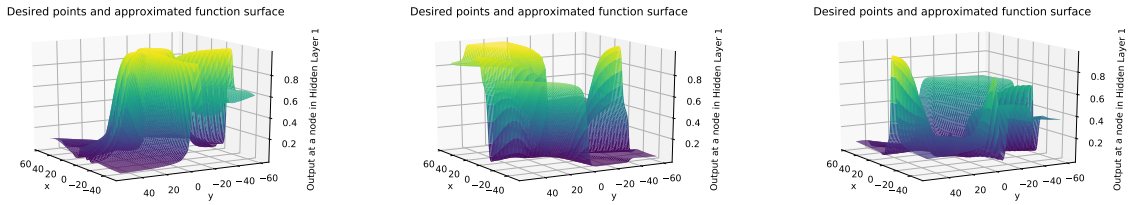

Figure 2.9: Output of nodes at Output Layer after 500 epoch.

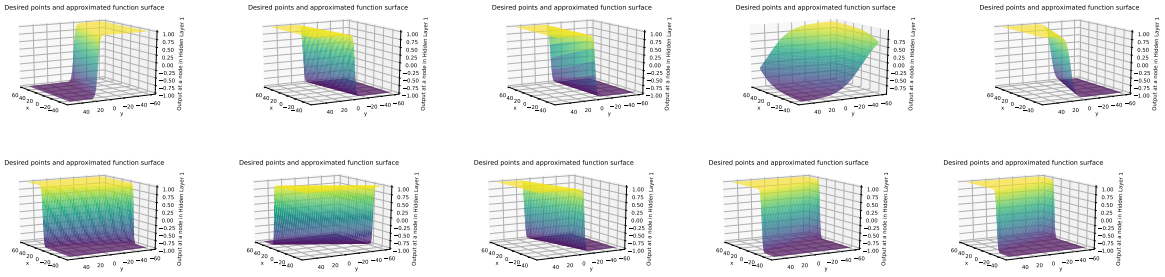

Figure 2.10: Output of nodes at Hidden Layer 1 after 1000 epoch.

- We observe from these plots that the first layer of the hidden layer only introduces one layer of non-linearity and hence the output always looks like a sheet of *tanh* activation function. The second layer is where things get interesting and we start seeing all kinds of weird shapes.
- It is interesting to see the outputs of the final output nodes after 1000 epochs (Fig 2.12). We can indeed see the bulge in the first plot in the centre which corresponds to the orange-red region from the Decision Region plot. The other two plots are a bit messed up due to the conflicting nature of the two classes.

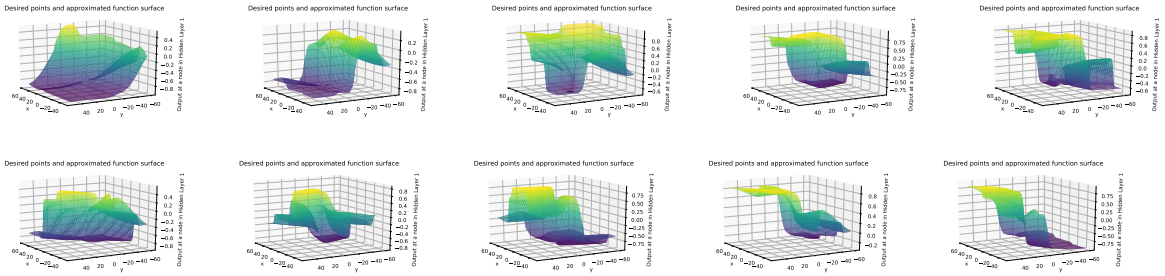

Figure 2.11: Output of nodes at Hidden Layer 2 after 1000 epoch.

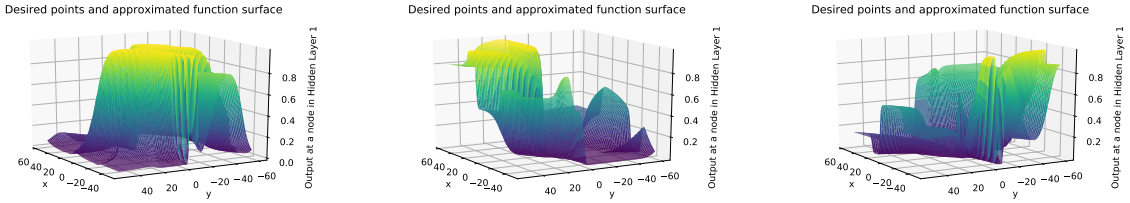

Figure 2.12: Output of nodes at Output Layer after 1000 epoch.

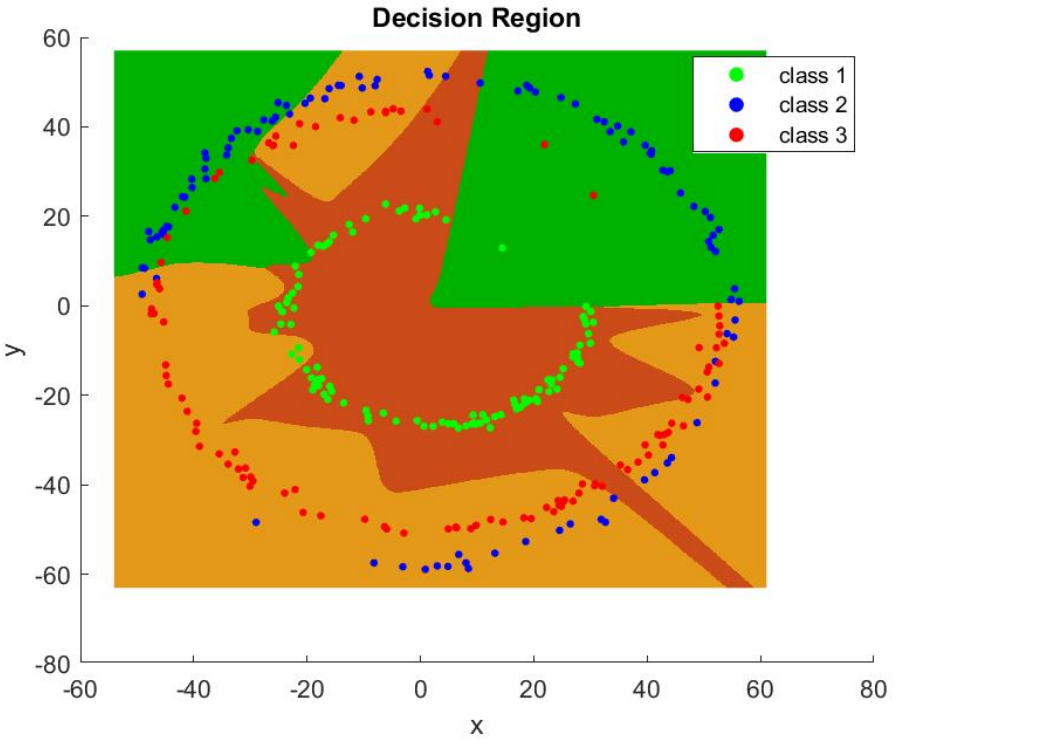

Figure 2.13: Decision regions after training and scatter plot of training data for the 2D Non-Linear classification task.

3 Image Classification

3.1 Network Specification

The input has layer has 32 nodes and output layer has 5 nodes. Hidden Layer 1 has 20 nodes while Hidden Layer 2 has 10 nodes. The parameters are set as $\eta = 0.01$, $\alpha = 0.1$ and $(\rho 1, \rho 2) = (0.9, 0.99)$. We

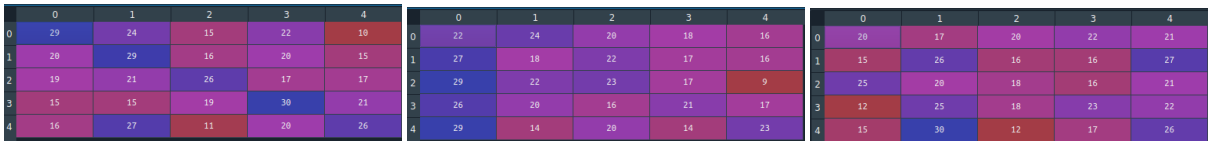

Figure 3.1: Confusion Matrices for evaluation on the Validation Data using normal Delta Rule, Generalized Delta Rule and ADAM respectively. Here indices 0, 1, 2, 3, 4 represent image class for **bird, cat, deer, ship and truck** respectively.

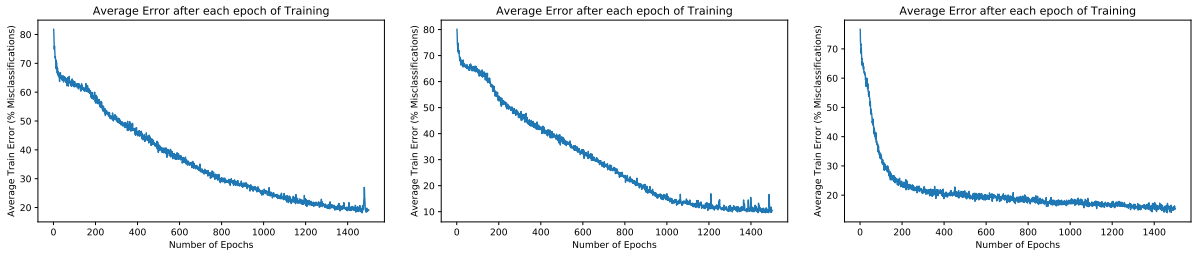

Figure 3.2: Error vs Epoch graph for Training Data using normal Delta Rule, Generalized Delta Rule and ADAM respectively.

use *tanh* activation function in hidden layer nodes and *softmax* activation for output layer.

3.2 Observations:

- In the Error vs Epoch plots for the three weight update policies, we can clearly see that although the final absolute error is similar, but ADAM method reaches near the saturation value very fast, with about one-fourth epochs compared to other two.
- Since the image feature vectors extracted were of 512 dimensions, we used Principle Component Analysis (PCA) on the entire training data to transform it to a reduced dimension space, 32 dimensions in our case. This is done so that the number of parameters to be estimated is on par with training data available and also the size of the network doesn't blow up drastically.
- If the number of nodes on Hidden layers is increased, the Train data classification accuracy increases, but then the results on validation data go even worse than a random guess due to over-fitting on the training data.
- There is no particular insight that we can derive from the validation results as for all the three methods the results are near random. The best accuracy is obtained is for the normal Delta rule after 1500 epochs which is at **28%**.