Assignment 1 Report

DD2424

Mehrdad Darraji

1. **Gradient Implementation and Test**

Using the formulas from lecture, I was able to successfully compute the gradient for model W and b.

* Let be the data n the mini-batch
* Gather all and from the batch into the matrix

,

* Complete the forward pass
* Complete the backward pass
* Add he gradient for the regularization term

I computed the relative error between the numerically computed gradient value and an analytically computed gradient value with an eps value of 0.0001

The result seemed to be very well as the error between my gradient function and the ComputeGradsNumSlow was a very small. Below are different tests I ran and results:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Batch Size | 50 | 50 | 50 | 200 | 200 | 200 |
| Lambda | 0 | 0.1 | 1 | 0 | 0.1 | 1 |
| W error | 1.54e-8 | 3.68e-7 | 1.99e-7 | 1.32e-8 | 5.44e-7 | 1.20e-6 |
| b error | 7.33e-10 | 7.33e-10 | 7.33e-10 | 6.70e-10 | 6.70e-10 | 6.70e-10 |

1. **Experiments**

Through four experiments, we analyze the results of the loss functions and accuracies of the train and validation datasets, then afterwards have a visualization of the weight matrix. Finally, we see the accuracy level achieved by each experiment on the test dataset.

*Note:* in the plots, the blue line represents the training set, the red line represents the validation set, the x-axis represents the epochs, and the y-axis represents the loss or the accuracy.

* 1. **Experiment a: lambda = 0, n\_epochs = 40, n\_batch = 100, eta = .1**

The plots shown below have lines that are very jagged with high variance and a lot of noise in the weights– this is due to having a high learning rate and no regularization.

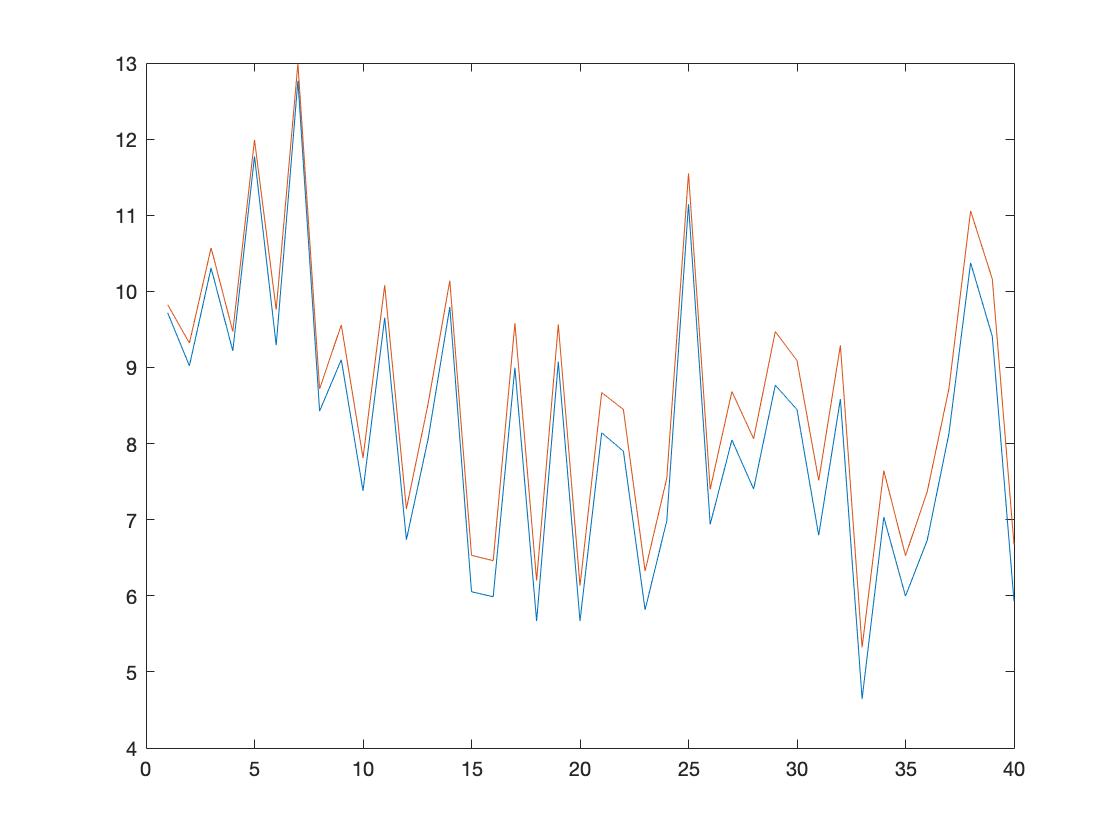


Figure 1: Plot showing loss curves for Experiment a.

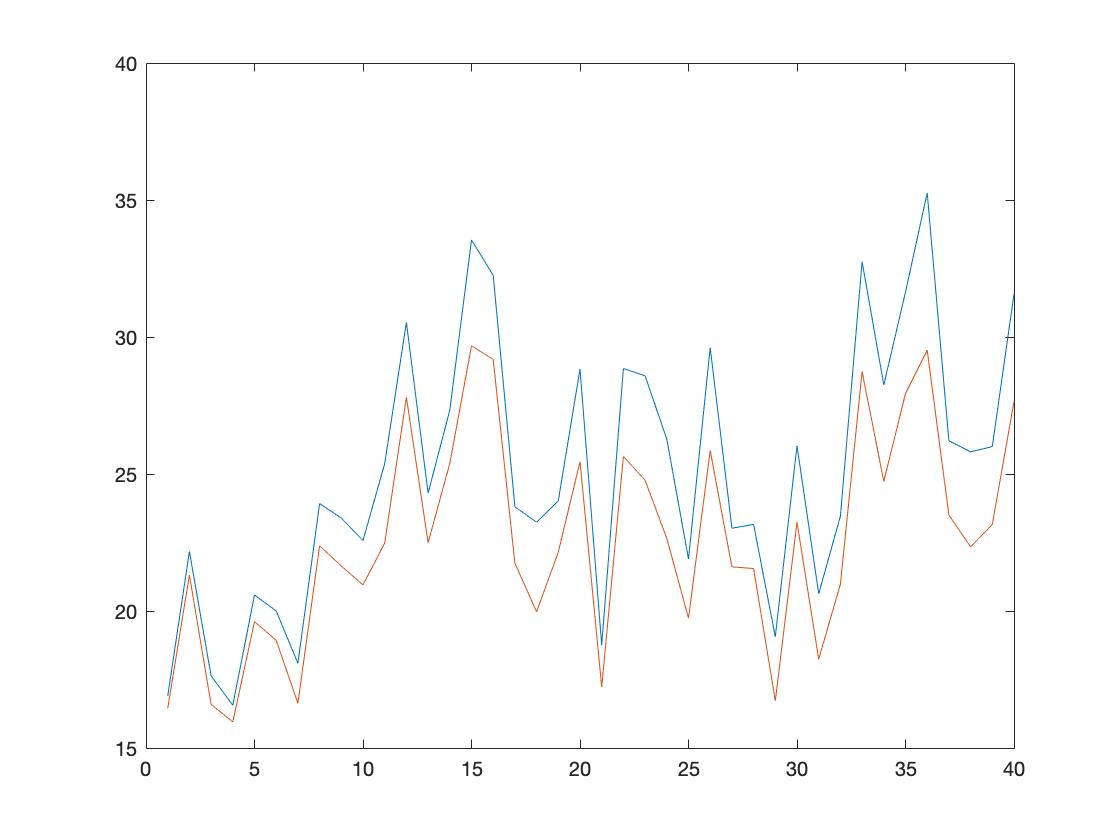


Figure 2: Plot showing the accuracy for Experiment a.



Figure 3: Visualization of the learned W matrix for Experiment a.

* 1. **Experiment b: lambda = 0, n\_epochs = 40, n\_batch = 100, eta = .01**

This experiment resulted differently than the last experiment. It has a lower learning rate – this smoothed out the curves in the plots and had a more stable weight matrix. This was an improvement from the last experiment, but the images are still a bit noisy.

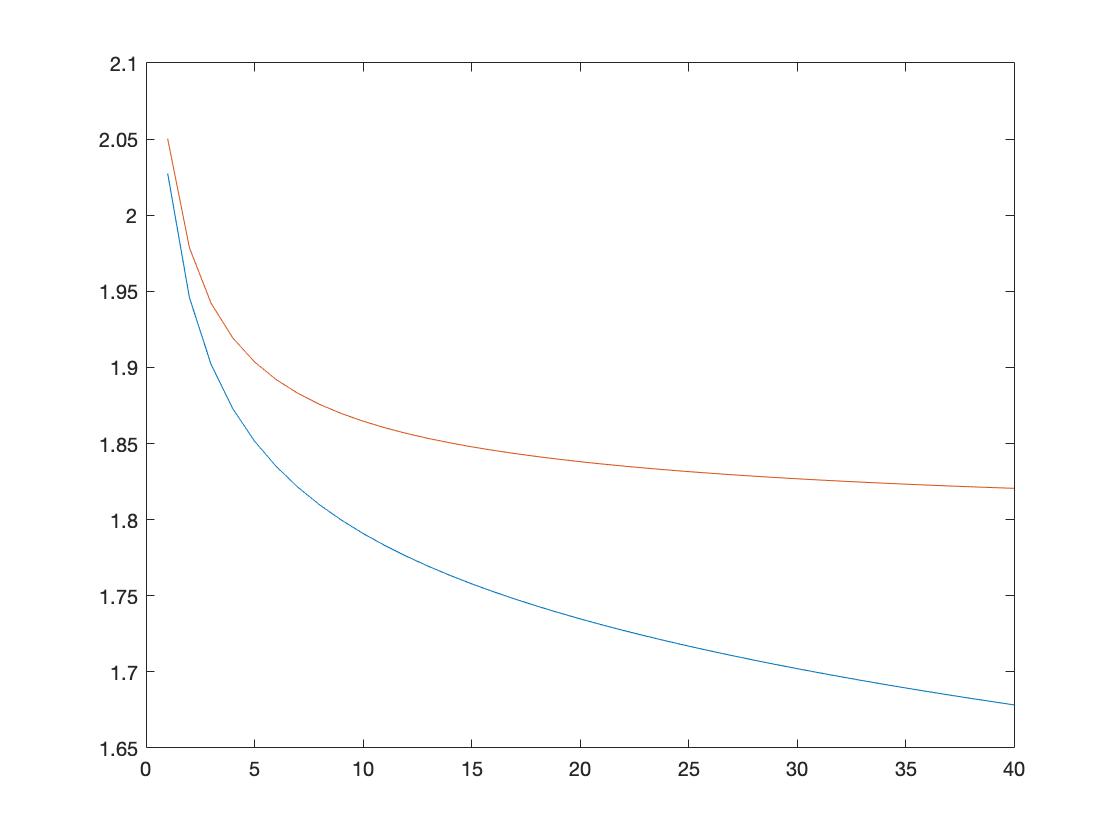
****

Figure 4: Plot showing loss curves for Experiment b.

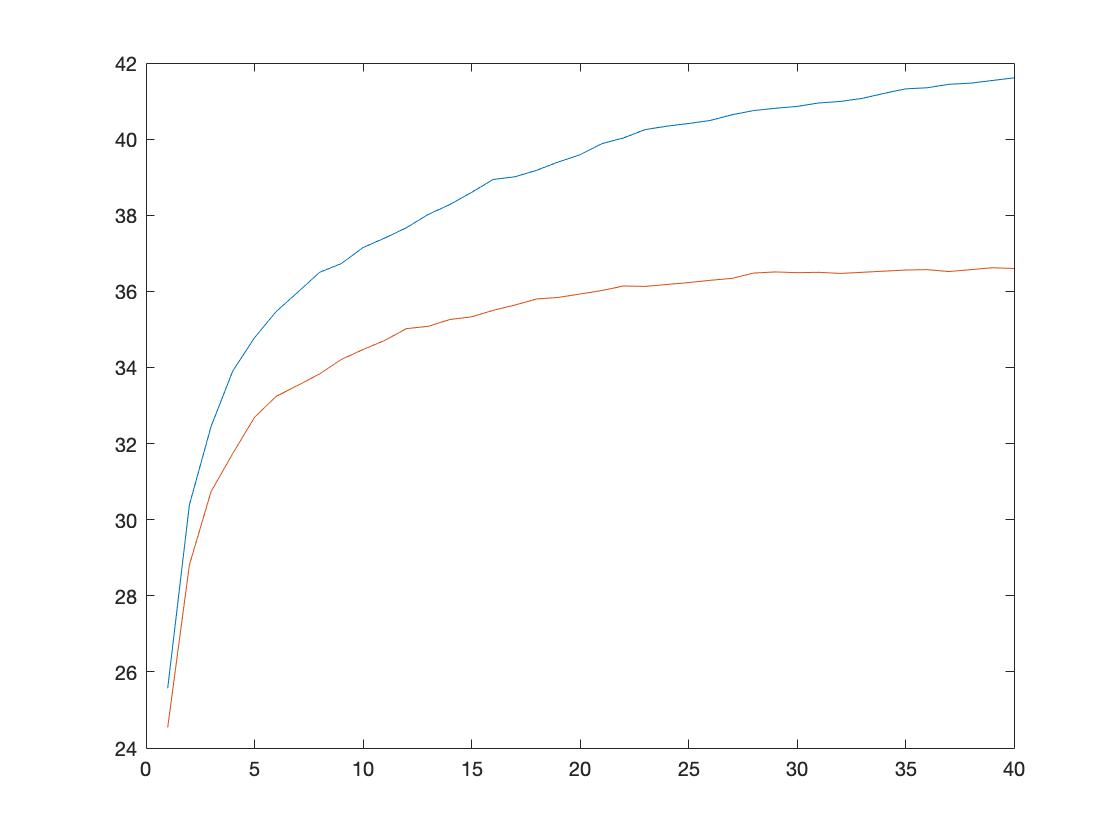
****

Figure 5: Plot showing the accuracy for Experiment b.

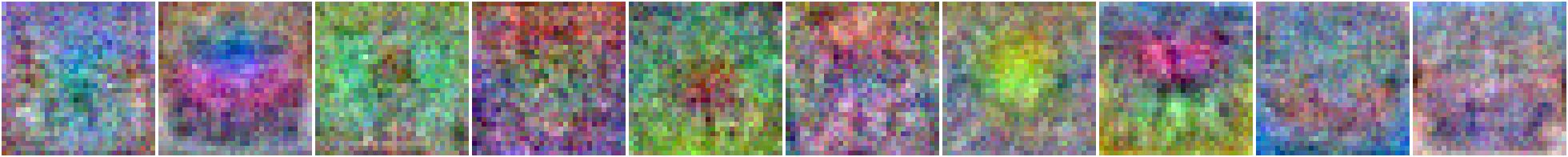
****

Figure 6: Visualization of the learned W matrix for Experiment b.

* 1. **Experiment c: lambda = .1, n\_epochs = 40, n\_batch = 100, eta = .01**

In this experiment, we decided to increase the regularization term – this resulted in a smoother weight matrix visualization and the performance became more stable than before. Another change is in the plots, where it takes less epochs for the curves to become stable.

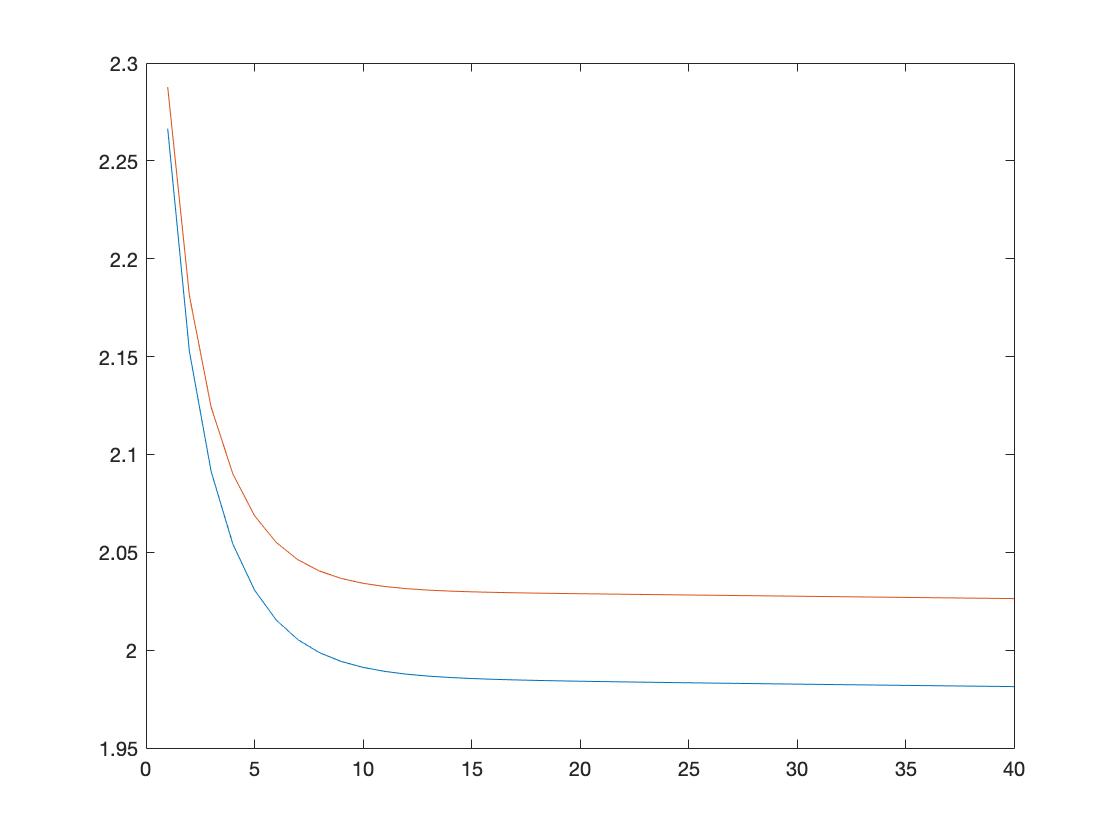
****

Figure 7: Plot showing loss curves for Experiment c.

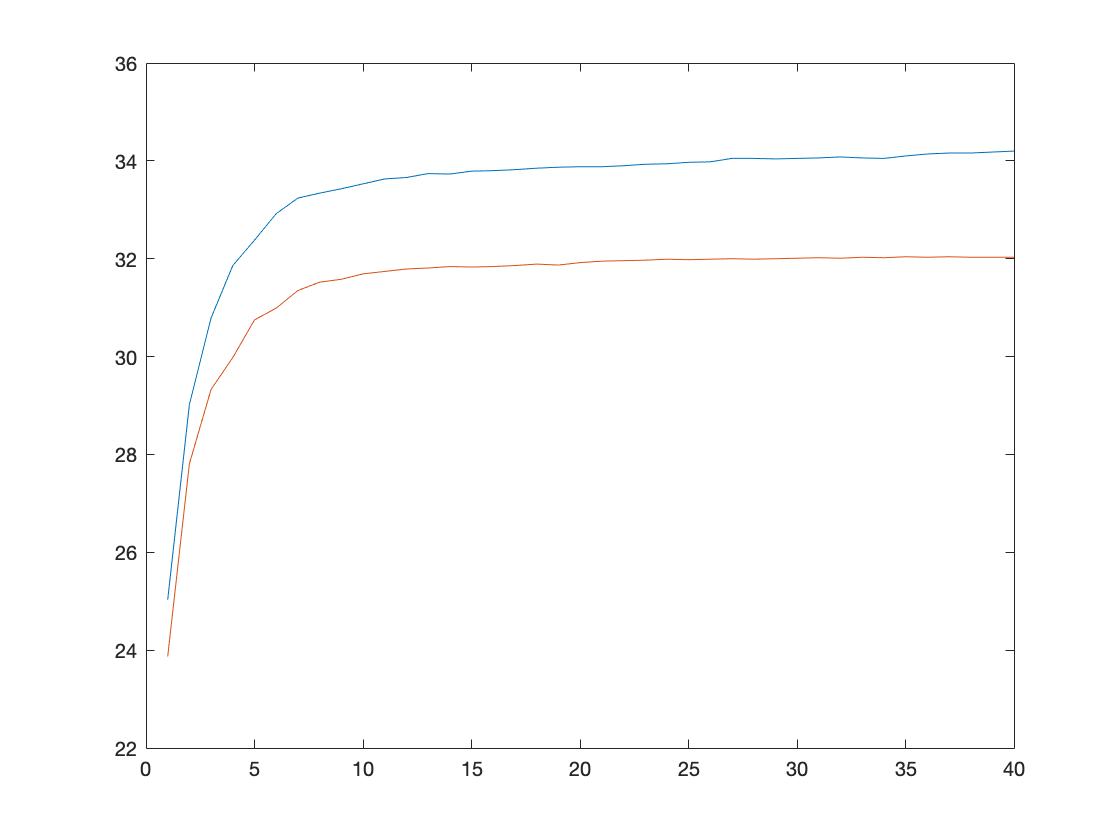
****

Figure 8: Plot showing the accuracy for Experiment c.

****

Figure 9: Visualization of the learned W matrix for Experiment c.

* 1. **Experiment d: lambda = 1, n\_epochs = 40, n\_batch = 100, eta = .01**

In the last experiment, we increased the regularization even more; however, it was too high and resulted in low performance. The loss and accuracy plots suddenly make a turn instead of smoothly stabilizing.

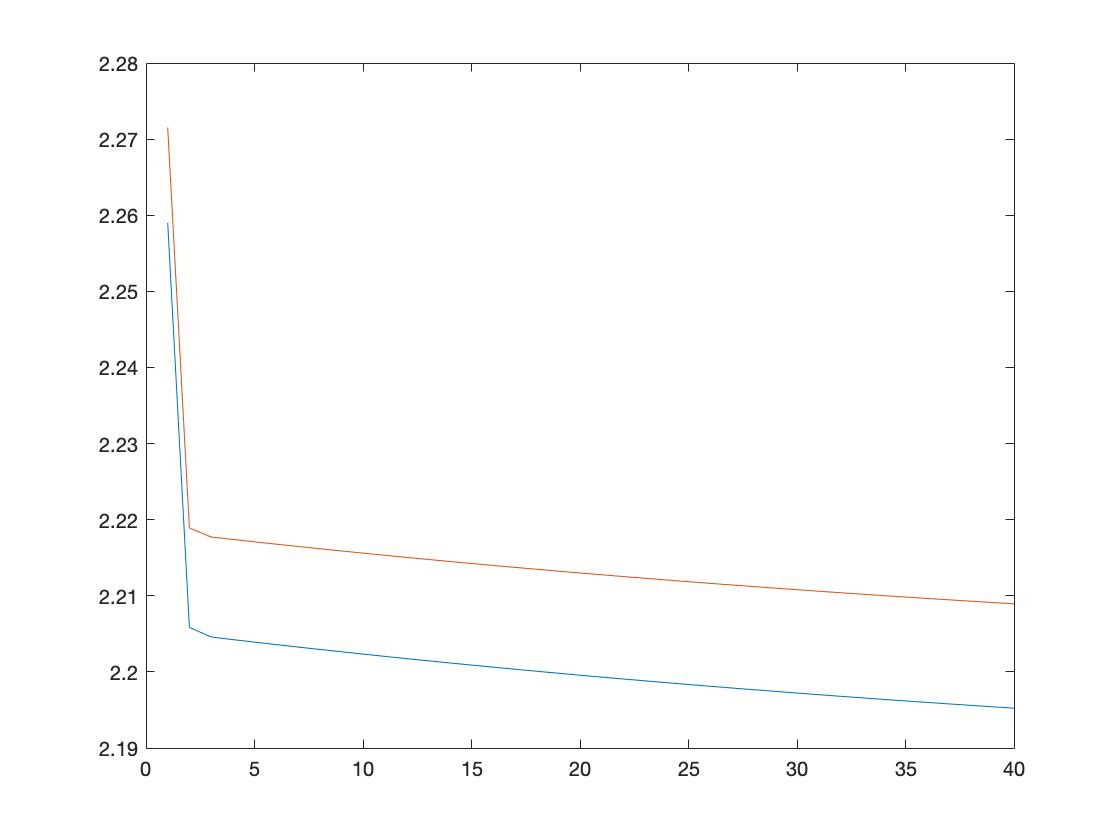
****

Figure 10: Plot showing loss curves for Experiment d.

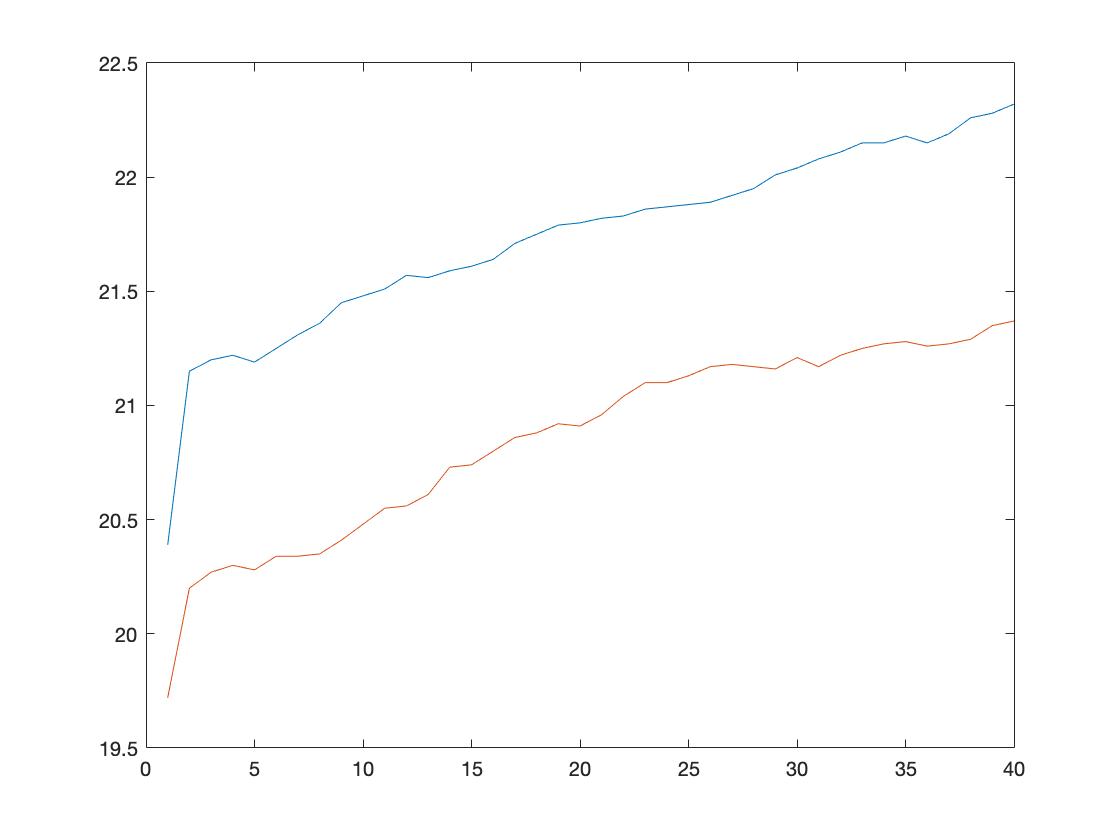
****

Figure 11: Plot showing the accuracy for Experiment d.

****

Figure 12: Visualization of the learned W matrix for Experiment d.

After each experiment, I get the final test accuracy that my network achieves after each of the training runs. Below are the results:

|  |  |
| --- | --- |
| **Experiments** | **Accuracies** |
| a | 27.58% |
| b | 36.65% |
| c | 33.37% |
| d | 21.92% |