Deep Learning in Data Science

Assignment 1

Isak Gamnes Sneltvedt

April 4th, 2022

DD2424

Assignment 1

Gradient comparison

For the gradient comparison I checked the absolute differences between the numerical and analytical computed vectors. The function I wrote checks if the value is less than 1^{-7} which I chose after reading the Standford's course Convolutional Neural Networks for Visual Recognition (https://cs231n.github.io/neural-networks-3/gradcheck). There it says that if the absolute difference is less than 1^{-7}, you can be happy.

Loss and Cost function plots

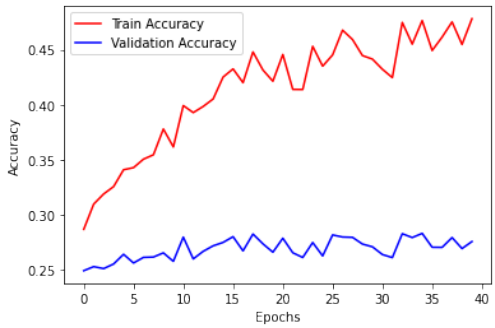

Figure 1: Showing accuracy with lamba = 0, eta = 0.1

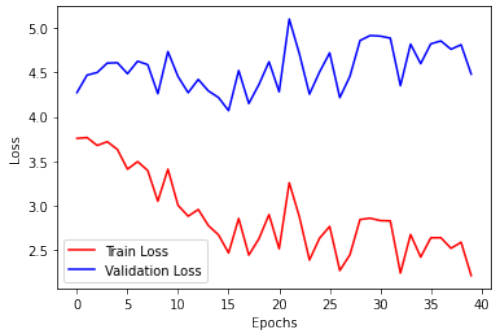

Figure 2: Showing loss with lamba = 0, eta = 0.1

In these plots we are able to see the difference between training with a high and low learning rate. When we have a eta = .1 we can see that the accuracy (figure 1) "jumps" back and forth. This is due to the high learning rate. The eta makes it so that the weight and bias are adjusted by quite a lot in each update. Compared to the plot with eta = 0.001, we can

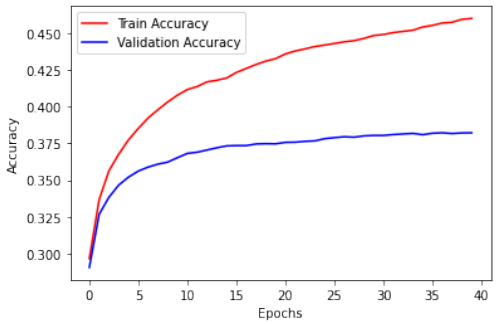

Figure 3: Showing accuracy with lamba = 0, eta = 0.001

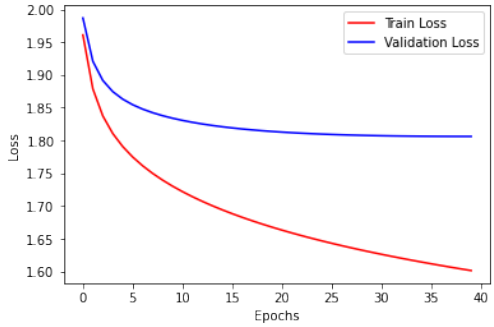

Figure 4: Showing loss with lamba = 0, eta = 0.001

see that the accuracy (figure 3) is way smoother. However, the performance in both cases is way better for training than for the validation. This is a classic example of overfitting. The overfitting can be avoided using regularization, which we do not do in this case due to the lambda value being set to 0.

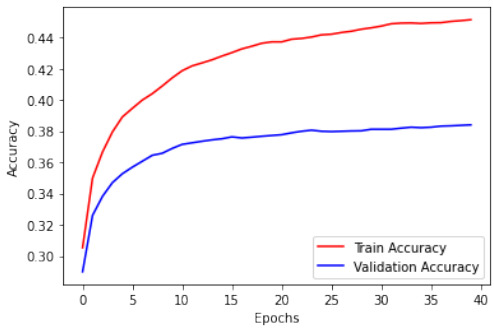

Figure 5: Showing accuracy with lamba = 0.1, eta = 0.001

This show how the performance has increased after adding the regularization. We can see that the loss is low for both the test and validation data. There is still a small gap between training and validation accuracy, but it is greatly reduced compared to the ones seen in

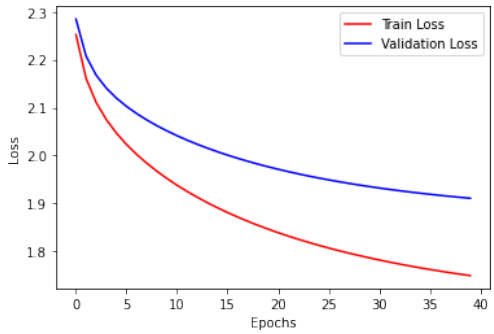

Figure 6: Showing loss with lamba = 0.1, eta = 0.001

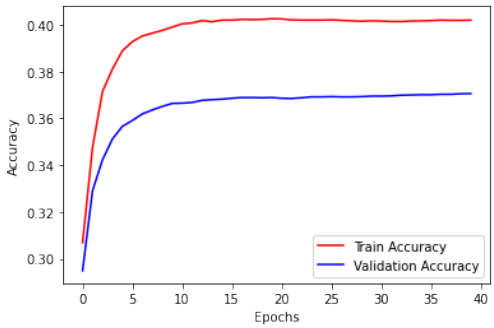

Figure 7: Showing accuracy with lamba = 1, eta = 0.001

figure 1 and figure 3

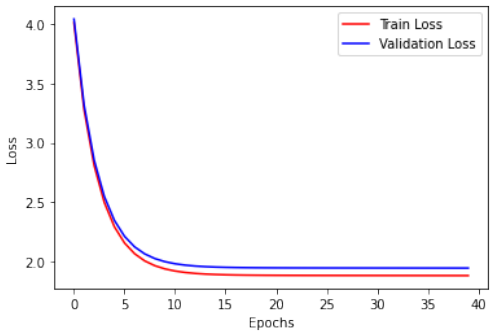

Figure 8: Showing loss with lamba = 1, eta = 0.001

Weight montages

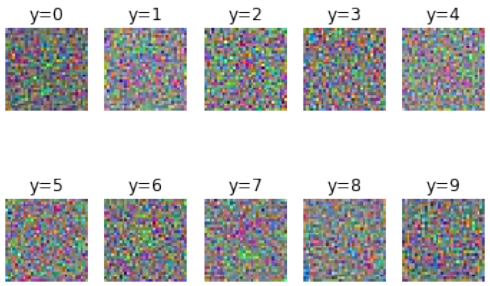

Figure 9: Showing weight with lamba = 0, eta = 0.1

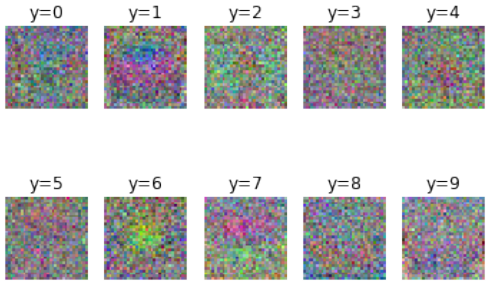

Figure 10: Showing weight with lamba = 0, eta = 0.001

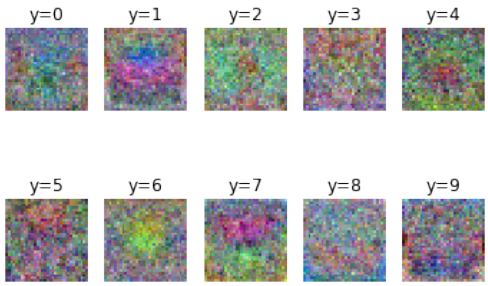

Figure 11: Showing weight with lamba = 0.1, eta = 0.001

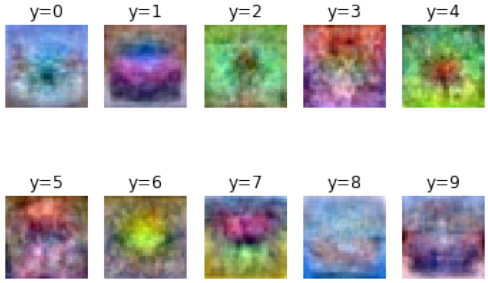

Figure 12: Showing weight with lamba = 1, eta = 0.001

Final test accuracy

After running the gradient descent algorithm using a good value for eta, as well and adding regularization, my model managed to get an accuracy of around 37%.