Assignment 1

Max Junestrand

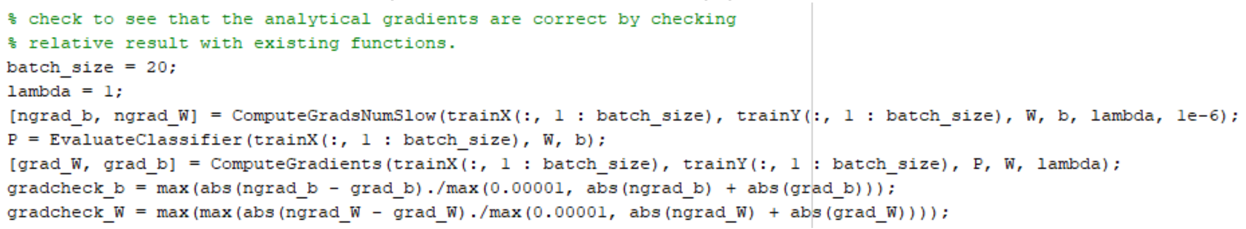
March 31, 2021

1 Figures, plots and comments

Each subsection covers plot, figures for different values for lambda (regularization) and eta (the learning rate). Each test ran 40 epochs, meaning that it went over the data 40 times, and the size of each of those batches was 100.

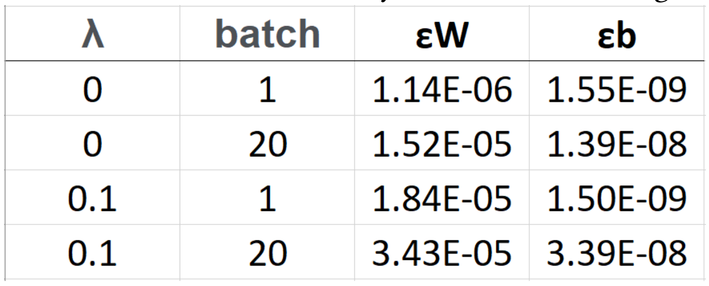
I implemented the code based on the lecture notes and suggestions from the assignment specification. I also correctly computed the gradient analytically, which I verified with this code:

Figure 1: code for checking gradients



```
% check to see that the analytical gradients are correct by checking
% relative result with existing functions.
batch_size = 20;
lambda = 1;
[ngrad_b, ngrad_W] = ComputeGradsNumSlow(trainX(:, 1 : batch_size), trainY(:, 1 : batch_size), W, b, lambda, 1e-6);
P = EvaluateClassifier(trainX(:, 1 : batch_size), W, b);
[grad_W, grad_b] = ComputeGradients(trainX(:, 1 : batch_size), trainY(:, 1 : batch_size), P, W, lambda);
gradcheck_b = max(abs(ngrad_b - grad_b)./max(0.00001, abs(ngrad_b) + abs(grad_b)));
gradcheck_W = max(max(abs(ngrad_W - grad_W)./max(0.00001, abs(ngrad_W) + abs(grad_W))));
```

Figure 2: differences between analytical and numerical gradients



λ	batch	εW	εb
0	1	1.14E-06	1.55E-09
0	20	1.52E-05	1.39E-08
0.1	1	1.84E-05	1.50E-09
0.1	20	3.43E-05	3.39E-08

As seen in the table above, the differences are sufficiently small for us to use them and continue with the assignment. I used the more reliable method comparing the relative error as between analytical and numerically computed gradients as described in the assignment.

General comments will be at the end.

1.1 Lambda = 0 Eta = 0.1

Figure 3: Loss plots for lambda = 0, eta = 0.1

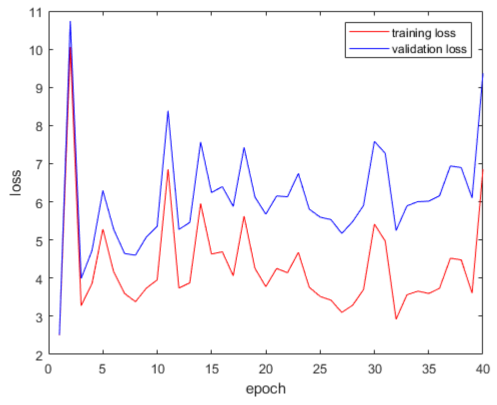
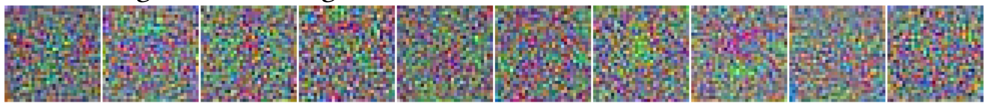


Figure 4: W images for the classes for lambda = 0, eta = 0.1



Training accuracy: 44.51

Test accuracy: 28.25

1.2 Lambda = 0 Eta = 0.001

Figure 5: Loss plots for lambda = 0, eta = 0.001

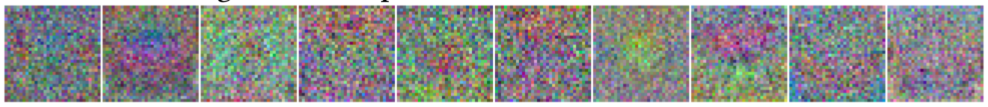
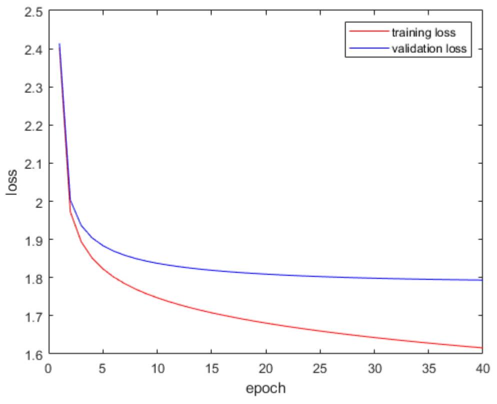


Figure 6: W images for the classes for lambda = 0, eta = 0.001



Training accuracy: 45.42

Test accuracy: 38.58

1.3 Lambda = 0.1 Eta = 0.001

Figure 7: Loss plots for lambda = 0.1, eta = 0.001

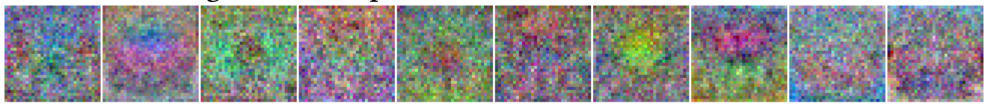
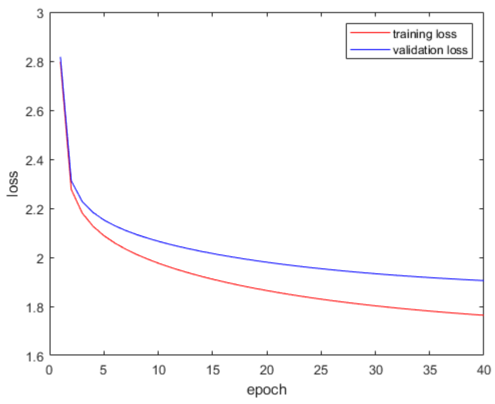


Figure 8: W images for the classes for lambda = 0.1, eta = 0.001



Training accuracy: 44.65

Test accuracy: 39.24

1.4 Lambda = 1 Eta = 0.001

Figure 9: Loss plots for lambda = 1, eta = 0.001

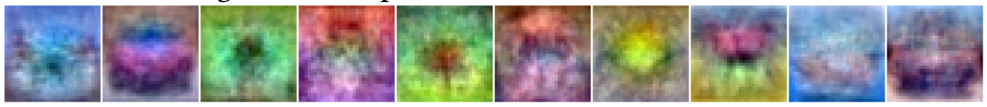
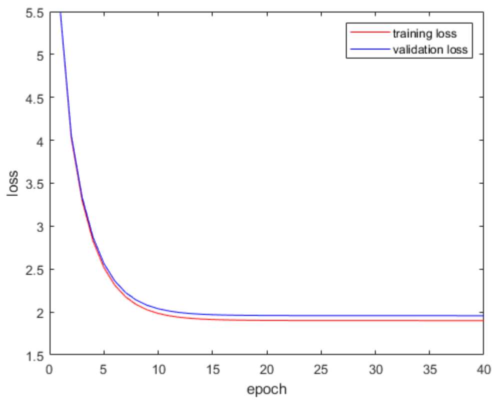


Figure 10: W images for the classes for lambda = 1, eta = 0.001

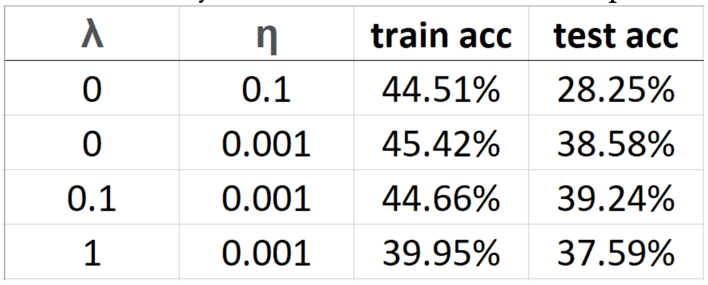


Training accuracy: 39.95

Test accuracy: 37.59

2 Summary

Figure 11: Accuracy for train and test on different parameters



λ	η	train acc	test acc
0	0.1	44.51%	28.25%
0	0.001	45.42%	38.58%
0.1	0.001	44.66%	39.24%
1	0.001	39.95%	37.59%

When we decrease the learning rate (eta) test accuracy increases and the loss function converge slower, which reduces oscillation or and the risk of missing local minimas.

When we increase the regularization (lambda) we restrict the network and generalize it. When we do this, from 0 to 0.1 we can observe an increase in test accuracy. When we however increase this to much we end up with a network that generalizes too much and cannot represent the data in a meaningful way.

I'm surprised the last test didn't provide better accuracy however, as the weights (when viewed as images) seem to have a much clearer picture and show more detail.