Assignment 2

Max Junestrand

April 15, 2021

1 Figures, plots and comments

Each subsection covers a question of the assignment.

1.1 Gradient Checking

I used extremely similar code from last week to check the analytical gradients with the numerical, the only real differences is that we check the gradients for both W1, W2 and b1 and b2, compared to last week when we only checked W and b. The relative error rate between the analytical and numerical gradients with lambda = 0 was

Figure 1: relative error rate for gradients

W1	1.22E-06
W2	2.50E-06
b1	1.17E-06
b2	2.92E-06

We conclude that the error rate is very low and we continue with the assignment.

1.2 Figures for default values

In figure 2, 3, and 4 you find training curves (cost, loss, accuracy) for one cycle of training. And in figure 5, 6, and 7 you find training curves for three cycles of training. They are reasonably replicated against the benchmark that we should recreate imo. In the first figures we can see that after around 6 epochs the increase in accuracy and decrease in loss and cost have flatten, yet when the eta changes we keep getting better results. This is due to the cyclic nature of the eta going up and down. It is desirable that it looks this way because it allows our network to explore wider search space and can thus find a better local optima. In the last figures the same argument is even clearer, and here we reach a little better accuracy, perhaps it would be even

better for more cycles as it seems to have an upwards trend (?), and a little lower cost and loss as well, for both the training and validation set.

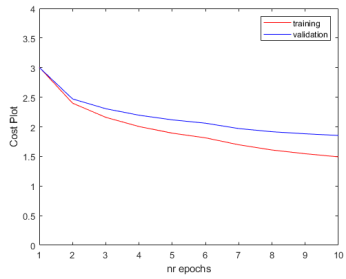

Figure 2: Cost Plot

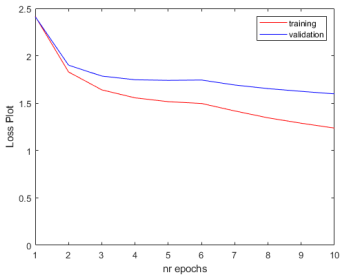

Figure 3: Loss Plot

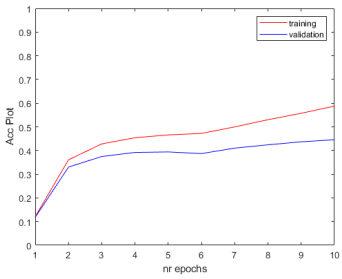

Figure 4: Accuracy Plot

The below figures run for 3 cycles.

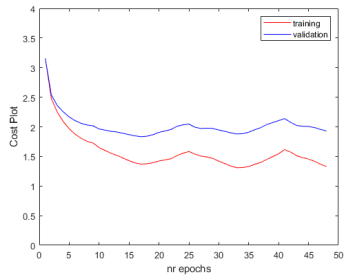

Figure 5: Cost Plot

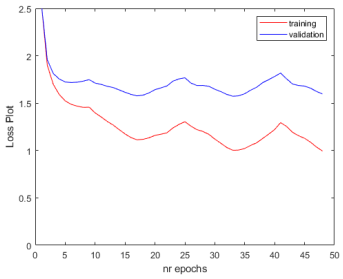

Figure 6: Loss Plot

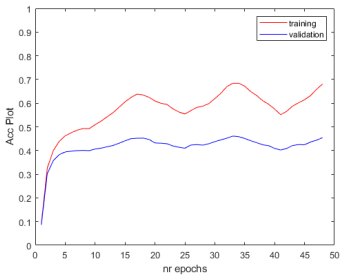

Figure 7: Accuracy Plot

1.3 Coarse Search for Lambda

The coarse search for Lambda begins with "testing" different lambdas by generating different values between 10e-5 and 10e-1. I generated 8 different values for lambda and noted their accuracy on the validation test. The validation set was 10% of the original available training data (out of 50.000).

The table for the different generated values of Lambda and their corresponding accuracy is found below.

Figure 8: Coarse search for a good lambda

				Rough Tuning				
Lambda	0.094341	0.000199	**0.003178**	**0.002134571**	**0.004148**	0.099224	3.08E-05	0.000479
Accuracy	0.4006	0.5108	**0.5142**	**0.5156**	**0.526**	0.3994	0.5062	0.5088

We can note that the bold values of lambda and their accuracy on the validation set are the three best. Meaning that lambda = 0.003177645, 0.002134571, and 0.004148018 were the best. With an accuracy between 51.4 and 52.6%. We will thus use these values to continue our search for a better Lambda in the next step.

1.4 Fine Search for Lambda

Now that we've concluded that we should search for lambda values around 0.002 and 0.004, that's what we do. Here I again choose to search for 8 values of lambda, but this time between 0.001 and 0.01 or (10e-3 and 10e-2). Because all of our best Lambdas were in that interval and I wanted to search a bit below and a bit over them. Again we also use 5000 examples as validation and 45.000 as training. Below is the table for the generated lambdas and their accuracy on the validation set.

Figure 9: Fine search for a good lambda

				Fine Tuning				
Lambda	0.001223	**0.001013**	0.002414	0.002812579	0.001103	0.001561	0.004579	0.00787
Accuracy	0.518	**0.5212**	0.511	0.5044	0.5198	0.5092	0.521	0.52

Again, the bold value is the best, and lambda = 0.001013229 is thus our best found lambda here. What can be noted is that it's accuracy 52.12% actually is a bit lower than some of the values we found in the coarse search, but since this is mostly for practise we will run with this value. Given a real scenario, we might also want to generate more than 8 lambdas, or to enforce some kind of demand on the randomness so that we generate lambdas with more difference.

1.5 Best Lambda

The fine search for lambda showed that lambda = 0.001013229 gave the "best" result on the validation set. So that Lambda was chosen to continue the experiment with. Here we train on all data except for 1000 examples which we use as validation set.

This network's performance was 52.26% on the validation data.

Figure 10: Loss plot for best lambda